FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

SirTom - Structural information retaining Triangle-based Occlusion Management

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Media Informatics and Visual Computing

by

Stefan Dietrich
Registration Number 1026167

to the Faculty of Informatics

at the TU Wien

Advisor: Dipl.-Ing. Dr.techn. Ivan Viola
Assistance: Dipl.-Ing. Johannes Sorger

Stefan Dietrich

Ivan Viola

Erklärung zur Verfassung der Arbeit

Stefan Dietrich
Sankt-Johann-Gasse 1-5/2/9, 1050 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 18.9 2017

Stefan Dietrich

Acknowledgements

I want to thank Ivan Viola and Johannes Sorger for their support and valueable input. Additionally I want to thank Andrew Berry for his feedback on the project and Matthias Glinzner for providing the implementation of his project. Finally I want to thank my family, friends and colleagues for their years of support and cooperation. I specially want to mention my mother, who I probably troubled more than necessary during my years of study.

Abstract

As cutaway occlusion management techniques are always bound to a certain degree of information loss and other occlusion management solutions do not always yield good results, specifically in regards to dense molecular assemblies like the ones displayed in CellVIEW. This makes it desirable to devise a strategy which optimizes cutaway techniques to retain more information. SirTom focuses on retaining as much information about structures inside such assemblies as possible by creating a cutaway strategy that opts to give them a mesh-like appearance. This is done by creating a mesh and then deciding via probability which molecules are more likely to retain this information in a pseudo-random fashion. SirTom grants the possibility to gain insight with less removed particles and is such a helpful algorithm for visualizing occluding surface structures.

Contents

CHAPTER 1

Introduction

Technology allows us to get a grasp of microscopic assemblies, among them cells. Understanding cells as a basic component of life is important not only in Biology but also Medicine. To deepen our knowledge and to communicate new discoveries, new visualizations had to be drawn by hand in the past.

To ease the production of mesoscale visualizations multiple computational solutions were conceived among these cellPACK [JAAA+15]. These provide biologists with easily interchangeable prefabs which can be altered with less work than drawing everything anew. The densely packed scenes created with tools like cellPACK then get rendered in CellVIEW[MAPV15].

The testing dataset is a densely populated 3D-Model of a cell that was created with aforementioned cellPACK. These models are assembled from protein molecules which in turn are defined down to an atomic level. This means that in our dataset every molecule of the cell and its surroundings are defined by their position, orientation and a value that defines which of the prefabs to pick from.

Typically points of interest in cell visualizations are concealed by other molecules such as blood plasma or the lipid membrane. Managing these occluding elements is a significant part of visualization and a proper management grants an observer better insight to detailed, previously hidden parts of the cell. The encapsulating nature of the membrane causes it to fully occlude all proteins inside it. When cutting away enough elements of the liquid membrane in a random fashion in order to get a view inside it we lose defining information about itself.

This causes the desire to view points of interest while retaining existing information as much as possible. For this purpose we have varying tools available, each with their own advantages. However, structural information is only kept marginally near the cut-off, not at all or the overall result is not satisfying. Managing occluding objects via transparency is not suited for dense structures. This is due to even a tiny amount of opacity leading

to visual clutter and eventually full opaqueness once enough instances overlap, removing the possibility to analyze points of interest.

The visibility equalizer(VisEQ)[MMS+16] is a tool designed to work particularly well for cells - it creates a screen door effect to open a window revealing elements hidden behind occluding layers. Regardless, it comes with a major drawback: Structural information inside the molecular assembly is lost. This constraint is caused by VisEQ only considering the clipping object as a tool to manage visibility, breaking down the membrane into individual phospholipids, failing to recognize the role of other lipids in the compound.

In order to convey more structural information an extension to VisEQ is needed. Taking from standard 3D graphics modeling tools, problems caused by occlusion are removed by representing structures as wire-frames, not only saving computational resources but also granting us a look inside. In order to achieve those looks we need to create a discrete representation of the surface, a mesh. SirTom then provides an algorithm that uses such a mesh to remove phospholipids in visualizations. This is done in a structured manner so that the remaining elements have a wire-frame-like appearance.

CHAPTER 2

Related Work

As this work encompasses topics in regards to 3D modeling and mesh creation as well as visualization techniques both will be covered in this chapter. Integral parts to understanding this work are cellPACK, CellVIEW and VisEQ as the suggested technique is built upon these technologies.

2.1 Visualization

In order to visualize a mesoscale structure, such as a cell, we have to look at two different things, occlusion management and how to visualize parts of a cell - the molecules.

Visualization as an extension of illustration has existed through most of human history, the earliest known materials are dating as far back to 6200 BC. These early instances of visualization mostly focused on mapping celestial bodies and geographic properties, but some instances of geometric representation existed. Over time these visualization developed to be more complex, describing perspective principles and Euler solving the problem of addressing three dimensional space.[Fri09]

As the first drawings where occlusion became a problem came into being it was an artistic or technical decision on which parts are important to be shown, and by extension, which occluding parts are to be omitted. During the industrial revolution and henceforth technical drawings became important in order to convey standardized procedures and conventions. This also caused the artists to develop a set toolkit, which they used to create drawing even people without proper topical knowledge could understand. [VG05]

Multiple applicable techniques to manage occlusion were devised over time and a collection of these described by Viola[VG05]. For simplicity we will split them into three groups of which only one will be discussed closer: Explosion-Based, Opacity-Based and Cutaway-Based. Explosion-Based occlusion management techniques rely on moving or splitting apart overlaying structures to reveal inlying detail. Opacity-Based methods focus on

altering the transparency of occluding objects or areas thereof to grant a view onto relevant information. Finally Cutaway-Based techniques select parts of or whole structures and remove them to create a visualization of the relevant matter.

Cutaway techniques rely on geometric base forms to be inserted into the object, structure or assembly to define a space which should be cut away. Alternating the use of geodesic curves to create cutaway definitions[LRA+07] is possible. The biggest drawback of cutaway based methods is that by removing parts of the visualized object it is always accompanied by a loss of information. The advantage of it is the relative ease of use. Given the understanding of which part is most interesting to the observer there is no additional fine tuning required once the form-giver for the cutaway is in place.

Going back in time, when the understanding of microscopic structures such as molecules increased, the need to visualize them arose. This caused the first visual representations to be created. An early molecule representation similar to CellVIEW was being crafted by John Dalton [Dal08]. Going from there Hofmann created the first three dimensional stick and ball models of molecules, which where later adapted to better use the third dimension. These models consisted of colored balls representing the atoms which were attached to each other with sticks representing the molecular bond of the atoms.

Linus Pauling and later Robert Corey permanently shaped our understanding on how to visually represent molecules with their space-filling molecular model, which is used in the CellVIEW framework. The size is scaled after Van der Waals radii. The angles between atoms in the model are determined by their angles in their crystalline form in organic compounds [Pau31] [CP53].

In the late 80s and early 90s of the 21st century these techniques of visualizing molecules carried over to computational rendering. A plethora of static [FHJL88][Eva93] and later dynamic [HDS96] visualization engines for molecules sprung up. These developments eventually led to cellPACK [JAAA+15], a tool derived from autoPACK, providing a model editing and creation toolkit specifically geared towards the visualization of cells satisfactory to the packing problem. Whereas tools like MegaMol[GKM+15], a state-of-the-art particle visualization tool[MAPV15], is able to render 100 million atoms at ten frames per second. There was no true large scale solution up until the creation of CellVIEW, which is able to render 15 billion atoms at 60 frames per second[MAPV15].

But not only is it important to render the molecules themselves but also to properly allow for in depth analysis of the rendered structures. For this purpose earlier discussed occlusion management techniques are used in order to replicate the visual fidelity of Goodsells works[SG11]. This influence is already visible in the groundwork for SirTom, the Visual Equalizer(VisEQ) by Le Muzic et al. [MMS+16].

2.2 3D Model

Since we are working with a dataset that doesn't hold any direct representation of surfaces [MAPV15] we need to find a way to replicate the original volume used to define the surface.

Glinzners[Gli16] solution is a quintessential building block for SirTom. This section will give an overview over relevant works to deepen understanding towards Glinzners work, mainly triangulation methods, followed by possible methods that can provide the algorithm with a base mesh and further content in regards to mesh optimization.

A simple way to triangulate points in space is Delaunay Triangulation for which multiple algorithms exist[LS80]. Chew[Che89] describes a method to populate 2D areas with as equilateral as possible triangles with the help of Delaunay Triangulation by adding additional points inside large-radius Delaunay circles. Rebay[Reb93] uses Delaunay Triangulation to create meshes in an efficient way by calculating connections and positions simultaneously. The resulting mesh is arbitrarily shaped.

As mentioned, not having a full representation of the surface and volume[Baj82] a way to recreate these is necessary. As we can create meshes from point clouds this is a solvable problem[LTW04] with multiple different approaches. The used test data has rather well defined point clouds which allows a simple sample-and-connect approach. More complex surfaces may need more reliable methods of point cloud simplification as suggested by Alexa et al.[ABCO$^+$03]. Pauly et al. [PGK02] provides an expensive direct to mesh point cloud simplification method[MD03]. A method estimating surface normals is described by Mitra et al.[MN03] in which local properties of sample points are used to assign normals to created meshes. It is important to point out that solutions which create convex meshes from point clouds are not sufficient in giving a representation for SirTom. For oriented point sets Kazhdan et al.[KH13] proposes a generalization of the screened Poisson equation to recreate surfaces, only using a sample from the initial set.

Meshlab[CCC$^+$08] provides implementations for the sample-and-connect approach this work uses, more specifically these are an implementation of the ball-pivoting algorithm for surface reconstruction as suggested by Bernardini et al.[BMR$^+$99] and Poisson Disk sampling for which Lagae et al.[LD08] collected and compared various methods. As ball-pivoting algorithms are not creating hole free meshes it is interesting to have a look at alternative algorithms or solutions which fix the occurring errors such as Liepas [Lie03] hole filling method that is able to fix arbitrary holes in manifold meshes.

A hole-free alternative would be the marching cubes algorithm as described by Lorensen et al.[LC87], a divide-and-conquer algorithm that "marches" cubes along a 3-dimensional field, checking if the cubes corners are on different sides of a threshold permitting up to 15 different resulting cubes. The boundaries of point clouds can be used as this threshold in order to apply the marching cubes algorithm to point sets. Müller[MFSD09] suggests another solution using these boundaries. The three-dimensional space gets disassembled into multiple two-dimensional depth layers. Along those he sends what results to be marching squares which closely mimic the behavior of marching cubes. These 2D planes get reassembled to eventually create a 3D mesh.

Meshes can also be simplified. Cignoni et al.[CMS98] gathered several algorithms: mesh decimation, simplification envelopes, multiresolution decimation, mesh optimization, progressive meshes and quadric error metrics simplification. He then compared them for

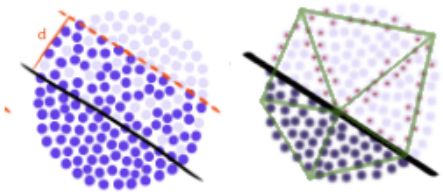
various metrics such as precision, runtime and memory requirements.

CHAPTER 3

Approach

The goal set is the creation of an occlusion management tool which is able to convey additional structural information of an identified structure inside CellVIEW. As previously discussed, this needs to be done while still allowing the user to monitor points of interest inside the structure. As phospholipids form structures such as the membrane, they are optimal subjects for this work. Previous occlusion management strategies used to see beyond the membrane were gradually removing the phospholipid elements in a *random* fashion.

To properly represent the individual molecules role as a building block of a structure rather than as a independently acting element, a new occlusion management technique is needed. This is done using a removal strategy that opts to make the remaining phospholipids appear in a wire-frame-like manner. The core technique to achieve this is



(a) (b)

Figure 3.1: Comparison between VisEQ and SirTom. The cutting plane is represented by the black line. (a) Showing the solution in VisEQ, providing a distance d input for a final cutoff, molecules between the distance and the cutting plane get removed at random. Source: [MMS+16] (b) Showing the solution by SirTom, instead of distance a transition function is used, providing smoother transitions. The mesh is represented in green, retained molecules are marked red.

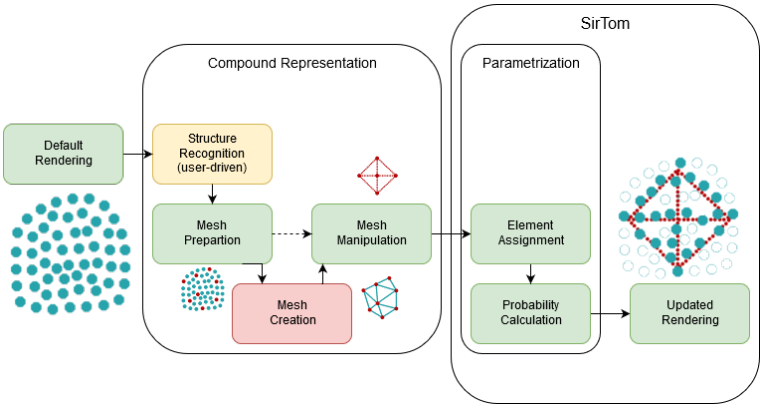


Figure 3.2: The process pipeline.

the SirTom algorithm which provides a gradual visibility parametrization in a structured way.

As the algorithm should retain the appearance of the membrane as a uniform anatomical structure we create a representation of the membrane surface via triangulation with equally sized triangles. The triangles should be sized in a manner that allows us to peek inside. Once this structure is created we assign individual lipids to the triangle. Removing central phospholipids first causes the triangles to gradually become transparent, ultimately causing the membrane to appear wire-frame-like as the elements close to the edges of the triangle get removed last. The approach removes elements at random, but the probability for an phospholipid to be removed is highest in the center of the triangle, and lowest at its edge.

As shown in Figure 3.2 the main process is split into 2 phases: The creation of the compound representation as preparation for the application of the SirTom algorithm and the algorithms application. As the first step of the Compound Representation phase the structures need to be user-identified. Preparation handles the point cloud creation as well as the first mesh creation. The step of creating the mesh currently is fully externalized to meshlab and needs user interaction. The precise steps are explained in the *meshlab.pdf* in the project files. The manipulation phase creates a new equilateral mesh which can be used as a decider for the Calculation Phase. SirTom uses the mesh to assign each molecule a triangle. During the next step the algorithm calculates a base-probability from the triangle, as to how important it is to retain structural information in its context via its barycentric coordinate inside the triangle. This is summarized as Parametrization. Then a transition function is applied in the shader to calculate the cutoff. Molecules remaining in the scene after the application of SirTom are called structural retainers.

3.1 Compound Representation

As already discussed it is essential to create a surface representation that is stable and retains the form of the compound structure. The data present in CellVIEW only provides us with the location of all molecules of such a structure. To circumvent this a point-cloud is created from these locations and then processed to create a mesh, which then gets further processed exactly as in the first two phases of Glinzners work[Gli16]. The third phase is not needed as SirTom is not depending on textured objects. Though texturing was considered as an alternative to using barycentric coordinates during the prototyping stage.

For SirTom to optimally pick structural retainers the mesh has to fulfill a few conditions: The mesh should consist of equally sized, equilateral triangles, secondly it resembles the original form of the structure as close as possible and finally its level of detail is picked in such a way, that structural retainers do not cause visual clutter and the second condition is still satisfiable.

3.1.1 Selection and Preparation

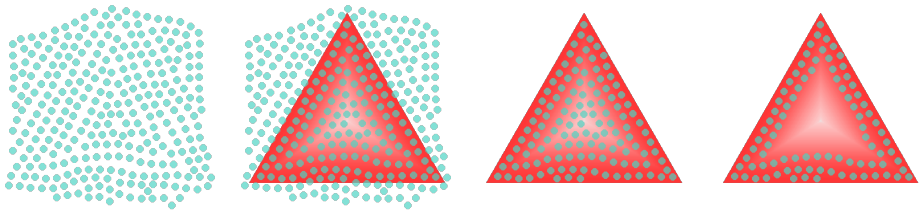
When selecting molecules to be handled it is important to make sure that they are actually part of a continuous structure, as the preprocessing steps will rely on this. A continuous structure is easily identifiable by each molecule neighboring to one another and thus eventually describing a surface which then can be turned into a mesh. Similar or corresponding surface molecules, such as the inner and outer membrane, are to be matched into the same structure, as they will be processed with the same mesh only if they were matched as one structure.

Once molecules have been selected as structure, their object-space positions are put into a point-cloud. This is done because these positions are offsets from the center of the dataset, providing us with a point-cloud centered around the zero coordinates. This attribute caused the consideration of alternative solutions on how to gather barycentric coordinates, which were later discarded due to the fact that this does not guarantee a mesh is convex. For more details on this see Section 3.2.

3.1.2 Mesh Creation and Manipulation

As the now existing point cloud causes unnecessary computational load in later steps it gets reduced with a function providing a well distributed sample from it. For this purpose we utilize the Poisson-Disk-Sample as it provides such an evenly distributed sample.

The remaining points in the cloud are always exactly on the surface of the structure and can as such can be treated as vertices for the mesh. Any algorithm that can resolve this point cloud into a manifold-mesh without cross-sections can be applied to create a base-mesh which then is used in the next step - Mesh Manipulation. While cross-sections are not a problem for the algorithm intrinsically, besides not representing the structure correctly, they cause visual clutter and hamper the observers ability to comprehend basic



(a) (b) (c) (d)

Figure 3.3: Example of the cutting process. (a) a collection of random molecules (b) the minimal barycentric coordinates of a triangle with the collection above it, a lower coordinate gets denoted by a stronger saturation (c) assignation of molecules to the triangle according to their positions (d) molecules with a high barycentric coordinate get cut from the scene

structural features. As it would be beyond the scope of this work to analyze all possible algorithms to achieve this a ball-pivot algorithm with additional hole-filling algorithm are applied.

As already mentioned, this base-mesh then gets treated fully analogous to the first two phases of Glinzners work. To summarize it, in the first phase newly generated pseudo-randomly distributed vertices which will later make up the mesh are put onto the old mesh' surface. Faces in the old mesh, called geometry mesh by Glinzner, that are proportionally big have a higher probability to receive a vertex for the new mesh, called texture-mesh in his work. These newly generated point for the texture mesh then repulse each other to create an evenly distributed mesh. In phase two the vertices get re-triangulated with a greedy algorithm creating a mesh close to the specifications with short runtime[Gli16]. The recommended amount of vertices for Glinzners algorithm is 50 for the membrane and 20 for the capsid in the test dataset.

The generated mesh has triangles with little skewness and similar size and depending on user specifications, a fitting level of detail. As the geometry mesh is already smaller in volume than the actual structure, while retaining its basic shape, Glinzners algorithm cannot extrude the bounds specified by it. As it cannot alter the base-meshes shape it allows us to conclude that the generated mesh is also smaller and of the same shape. Thus the conditions for the mesh are fulfilled by the automated parts of the algorithm.

3.2 SirTom Application

The SirTom algorithm itself parametrizes molecules in such a way, that with the help of a transition function the molecules in the center of a triangle get removed first. For this we need to know the location of a molecule in relation to the closest triangle of the texture-mesh, thus the first step the parametrization needs to take is to assign a triangle from the generated mesh to each individual molecule. The assigned molecules

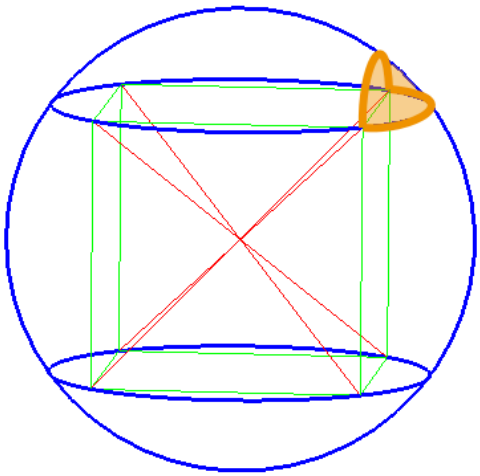


Figure 3.4: The orange region of the sphere cannot be orthographically projected onto the green cube along the cubes surface normals. Original source: wikimedia

then get projected onto their respective triangle. At this point it is trivial to extract the precise location of a projected molecule in relation to the triangle via its barycentric coordinates. The minimal value inside the barycentric coordinates dictates its proximity to an edge. This value is then normalized and compared to the transition function and the user-specified amount of elements to display, in order to decide weather or not the element is a structural retainer.

The molecules get assigned to a triangle with a greedy algorithm that is separated into two segments, finding the closest vertex inside the mesh and then finding the triangle where all of the combined vertex-molecule distance is smallest. This process is only valid due to the triangles being nearly the same size and positions with negative barycentric coordinates being considered as exactly on an edge.

To create as precise barycentric coordinates as possible the molecule coordinate is then orthographically projected onto the triangle along its normal, so that it lies planar to the triangle. This allows us to extract the barycentric coordinates which are then passed on to the next step of the SirTom algorithm. Negative barycentrics occur due to the molecules hovering above the mesh and only the vertices potentially being aligned with the actual surface. As the projection is orthographical certain molecules to close to the edge of the triangle might not be inside it. This can best be explained with the analogy of trying to orthographically projecting a circumscribing sphere onto its cube along the cubes surface normals (Figure 3.4).

An alternative process to get to barycentric coordinates would be to cast a ray from the molecule to the center of the mesh. As already discussed the mesh is always smaller than the actual surface. This is a faster solution as the used framework provides powerful

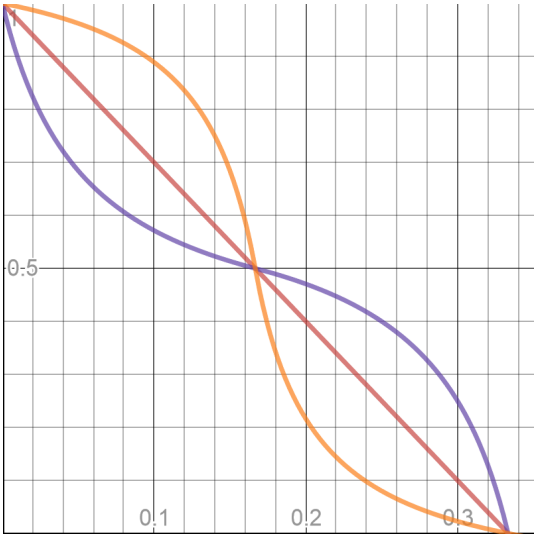


Figure 3.5: Examples of Equation 3.2 with (orange) $k = -0.68$, (blue) $k = 0.6$ and (red) $k = 0$; The x-axis represents minimal barycentric coordinate, the y-axis the base-probability.

ray-casting tools. This idea was discarded in favor of the above mentioned process due to one simple reason, the ray-casting solution required the mesh to be near convex to produce reliable results and thus its applicability to different datasets than the one used in testing is not guaranteed.

The minimal barycentric coordinate is later extracted and used as the base value for the probability function which gets calculated in Equation 3.1, this is done to stretch the minimal barycentric with the range $[0, 0.33]$ to the range of $[-1, 1]$.

This base value then gets plugged into Equation 3.2, which is a Normalized Tunable Sigmoid Function[Din10], to create the basic probability for each molecule. The k-value is user defined and allows for fine tuning as displayed in Figure 3.5.

$$n = 6x - 1 \tag{3.1}$$

$$\frac{1 - \frac{n - n*k}{k - |n|*2*k+1}}{2} \tag{3.2}$$

The function also provides a smooth transition based on the barycentric coordinate. After this point the input values from VisEQ are used to alter the base-probability to display more or fewer elements.

To conclude this chapter a quick overview over the whole process: The molecule coordinates are extracted and from a sample a mesh is generated, this mesh is then put through several processing steps to make sure it fulfills the algorithm preconditions. The

molecules from which the mesh is generated are assigned a triangle in the mesh, projected onto it and the barycentric coordinates calculated. The smallest of which is then used to calculate the per-molecule probabilities. These then get modified by user input via the VisEQ interface.

CHAPTER 4

Implementation and Demonstration

SirTom was added to CellVIEW-i and VisEQ, both of which are using the Unity-3D engine to render scenes. The engine provides a framework for script execution. These scripts were implemented in C# and integrated into the CellVIEW-i user interface. SirTom is implemented as MonoBehaviour, a Unity base class from which scripts are derived. Matthias Glinzner provided the source-code to his finished work which was included with minor alterations during the triangulation phase, giving the execution time more stability but allowing for one pair of edges to cross each other. The final phase of texturing the mesh was skipped as it is not essential to this work.

CellVIEW-i carries a collection of CPU- and GPU-Buffers containing information about each molecule. These informations are stored as $Vec4$ - these can be mapped to each other by the array-position thus describing a molecule. An additional $Vec4$-buffer was added to carry the barycentric coordinates to the GPU. The w-value of the $Vec4$ is used as a true/false indicator weather or not the currently observed molecule should be parsed by SirTom or the default VisEQ. It is also important to note that this buffer will always be created for every single molecule and initialized as $Vec4.zero$. As the order of items contained in the existing CellVIEW buffers is unknown it is necessary to check if the buffered element are marked as part of a structure during the execution of SirTom, so that the array-positions of the new and preexisting buffers align.

The mesh creation process is externalized to meshlab[CCC+08], which has all the functions introduced in Section 3.1 pre-implemented. To initialize the mesh creation phase it is necessary to select an identified structure with help of the expert interface of CellVIEW. This will cause CellVIEW to export a point cloud representing the coordinates in the .obj file format. To start with the main stage of SirTom it is necessary to provide an .obj with full-fledged mesh at the same location as the exported file was put to. To ease the creation

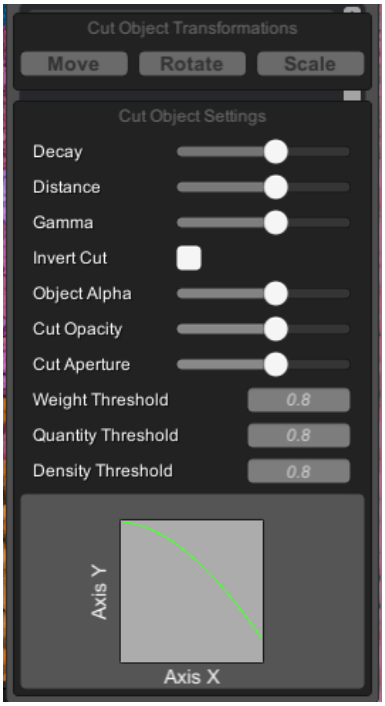
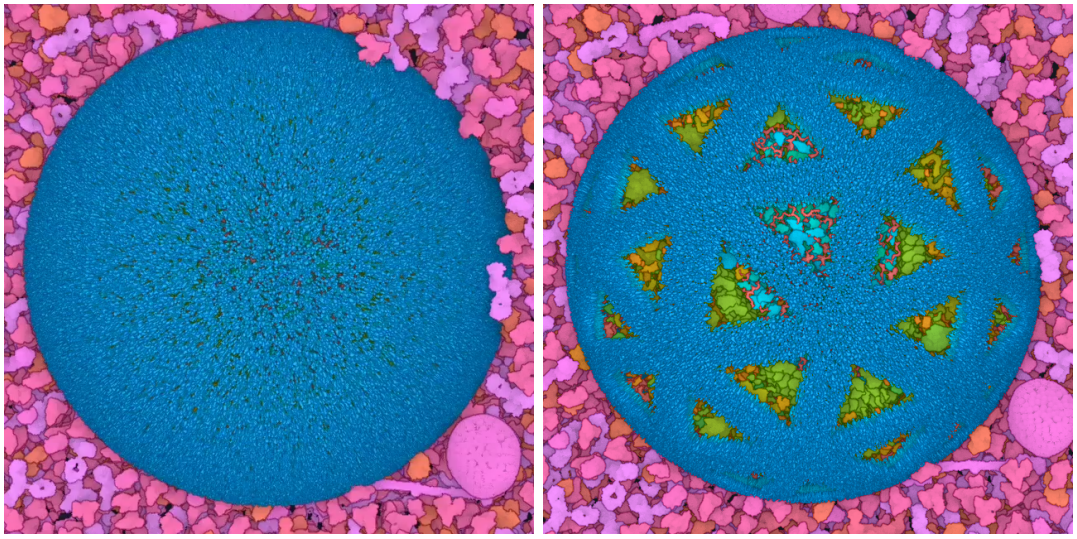


Figure 4.1: The VisEQ Cut Object interface - some of the Cut Object settings have different functionality compared to VisEQ

of this mesh there is an .mlx-script as well as a guide to the usage of said script in the meshlab.pdf which are both located in the project files. Meshlab provides several tools that can create suitable meshes, the included .mlx-script uses a Poisson-disk sampling algorithm [LD08] followed by a ball-pivot reconstruction and hole filling algorithm. These choices were specifically made since they provided the highest stability when loading and running the script. A fully automated version of the externalized meshlab process is implemented in the code, but not used, as meshlab-server, the command-line version of meshlab, is not fully functional with the described processes yet.

All steps up until and including the extraction of barycentric coordinates are performed on the CPU and need to be performed only once per molecule as the results may be persisted in the Unity *GameObject* if CellVIEW is run via Unity Editor. Only the final probability calculation is done on the GPU of the system. This is due to this being implemented as a direct alternative to the VisEQ standard probability calculation. VisEQ functionality is fully retained for non-structural elements even after SirTom starts applying its calculations to the to the structural units. The functionality of the Cut Object interface of VisEQ displayed in 4.1 is slightly different for molecules affected by SirTom. The decay defines the k-value of the probability function while the checkbox for cut inversion inverts the function so that elements that would usually retain the wire-frame structure get removed first. All other functionality that does not affect the



(a) (b)

Figure 4.2: Comparison of the membrane between the default result using VisEQ (a) and the occlusion management with SirTom (b), in both images 20% of the total phospholipids in the scene are removed, leaving 66% of the elements inside the cutting plane displayed. The transition functions are linear.

rendering of the cut object itself are defunct for SirTom.

4.1 Demonstration

This section contains examples for the usage of SirTom and how it affects the resulting render, showing the in-detail steps to replicate the result in 4.2. In both scenes the same amount of membrane elements are cut off by the occlusion management.

Once the CellVIEW application is running and the user skipped through the tutorial the controls get handed to the user. Here the user can use his mouse to control the scene and the default occlusion management as in previous iterations; The left mouse-button rotates the scene around the selected point, the middle mouse-button translates the scene and the right mouse-button or the mouse-wheel can be used to zoom in and out.

The first step to take is to enable the VisEQ interface by switching CellVIEW to Expert Mode with the corresponding button in the bottom right (see Figure 4.3). This also provides a button to assign molecules as a structure. Pressing this button will open an export dialogue for the point-cloud generated from the first resolvable molecule type inside the scene-tree of VisEQ. Taking the scene-tree from 4.4 this means that marking the "membrane" as a structure exports the "outer_membrane" element coordinates as a point-cloud. This makes SirTom look for a base-mesh in the same location once it

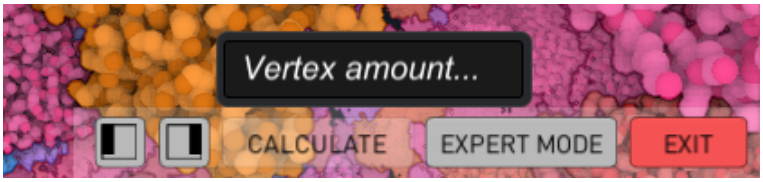


Figure 4.3: The bottom right of the CellVIEW interface, both the Calculate as well as the Expert Mode Button can be found here. In the black box above the Calculate button a user can specify the desired amount of vertices to be created during the mesh manipulation phase.

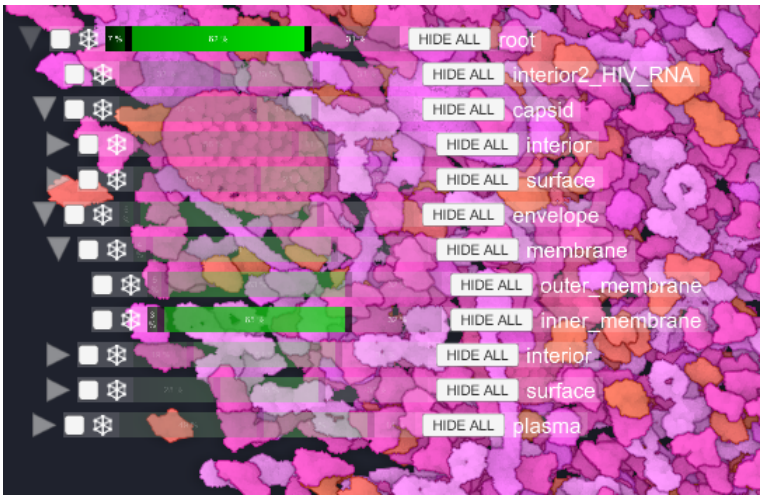


Figure 4.4: The expert mode has enabled the VisEQ interface as well as the structural buttons in the top-left corner of CellVIEW

is executed. An example process on how to generate a base-mesh is explained in the meshlab.pdf in the project files.

After a base-mesh is generated, a user has to enter a desired vertex amount into the black vertex amount box displayed in the bottom right of the window, above the "Calculate" button. Then, hitting the "Calculate" button on the bottom-right executes the remaining preprocessing steps and activates SirTom. Moving the slider to 20% hidden elements replicates 4.2b.

In Figure 4.5 a full scene rendered with SirTom is seen. It is observable that even though the triangles are nearly equilateral slight skewness already affects the thickness of the "wire-frame" lines. Figure 4.6 shows the full scene with the inverse SirTom probability. This provides little patches of lipids to be shown instead of a wire-frame and can be used as an additional way to render the scene. Finally 4.7 shows the membrane being cut with multiple planes while also applying the SirTom algorithm as a removal strategy for the lipids. Again it is observable how the slight skewness of the triangles will cause

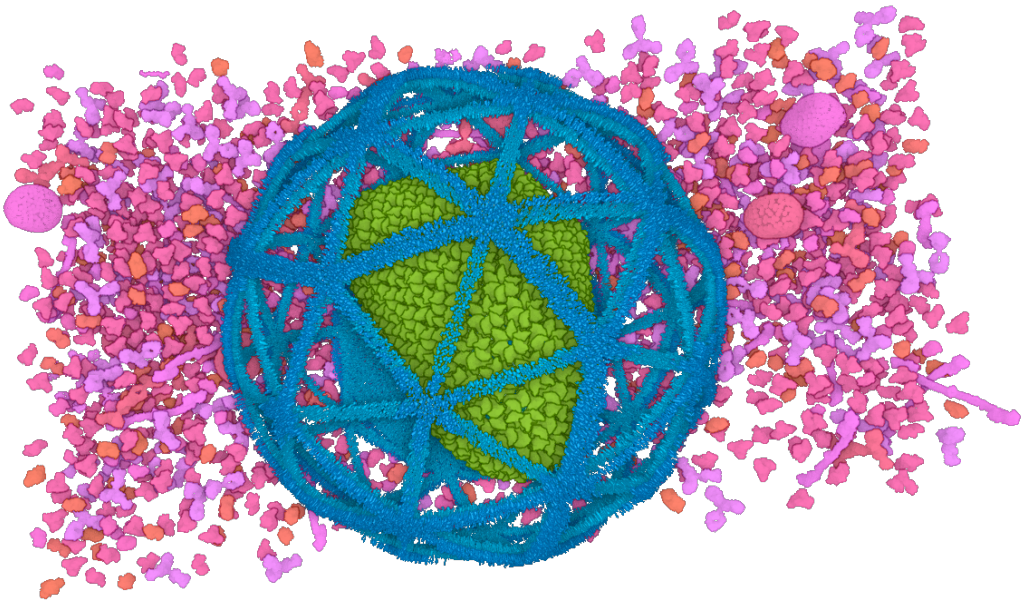


Figure 4.5: An example where most of the interior of the cell is removed to give a good look at the capsid in relation to the membrane

different line-thickness. The capsid is highlighted in this scene to allow it to stand out more prominently.

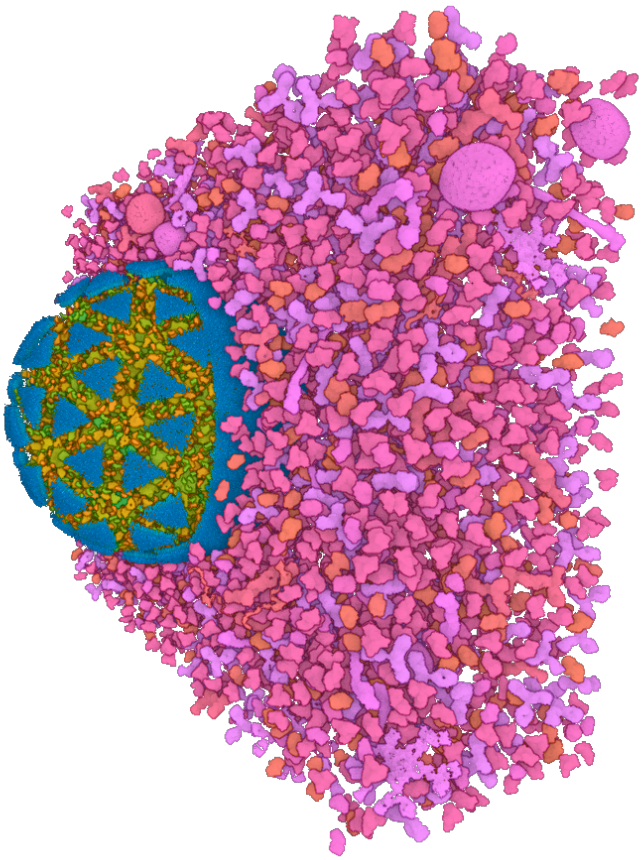


Figure 4.6: An example showing the inversion of the SirTom occlusion management, showing patches of lipids instead of a mesh-like structure

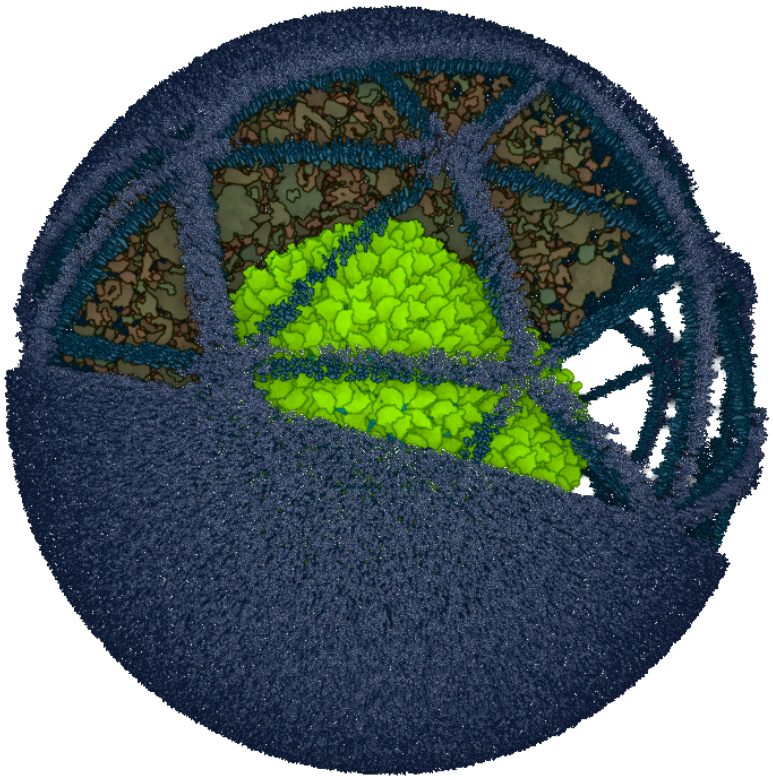


Figure 4.7: An example highlighting the capsid inside the cell with multiple cutting planes allowing a view into the cell

CHAPTER 5

Evaluation

This chapter will discuss qualitative properties of the new alghorithm as well as benchmarking the algorithms performance. The various code-segments directly involved with the unity engine have an estimated runtime of $O(m * n)$ where m is the number of molecules and n the number of vertices in the mesh. The benchmark was always applied to the same scene of an HI-Virus, with the calculations being done on its roughly 220 thousand membrane molecules. The export time was consistent with 550ms. The main source of computational power wasted is the lacking optimization on the mesh once it is exported from Glinzners algorithm[Gli16] to use with SirTom, as vertices are defined multiple times instead of reused. This increases the lookup time during the vertex assignment as pre-filter drastically. Glinzners algorithms runtime has been stabilized by allowing imperfect meshes to be accepted.

The specifications of the testing environment are as follows:

Intel Core i5-6600k @ 3.5GHz
16 GB RAM
GeForce GTX 970
Windows 8.1(10) Pro

The runtime analysis shows expected behavior for all code segments - the Vertex Assignment takes longest. As we can extract from Table 5.1 tripling the amount of vertices increases the time the algorithm needs to set up. The consumed VRAM stays stable due to the nature of the implementation already setting up the needed amount of memory before even being applied to molecules. During the calculation process an additional amount of RAM is needed, in the testing environment this caused for the need of up to 600MB being occupied,

Andrew Berry [Ber17] is proposing that the triangular structure caused by using a mesh as a structural representation for SirTom might be observed as a natural trait of cells,

Vertex Amount	Triangulation	Probability	Assignment	Total	vRAM
50 vertices	2394ms	3180ms	8412ms	14536ms	20.2MB
150 vertices	3092ms	14294ms	23351ms	41287ms	20.2MB

Table 5.1: Benchmark result on membrane

rather than an artifact of the structural representation. He suggested instead the use of a swiping effect to show how the structure looks on the inside and on the outside. He noted that given interactivity this might be not so much a problem. Deriving from that, preventing interactivity hampers the potential of understanding the structure, thus this solution is unsuited in a merely presentational manner.

Going back to Figure 4.2 we can see that SirTom allows a better view into a structure than the default VisEQ, granting an observer access to a point of interest while retaining more information about the cell on screen.

The nature of the SirTom algorithm and its dependence on a mesh representation of the objects has its advantages as well as drawbacks. The Compound Representation phase is not guaranteed to provide a valid mesh to use for further steps, a good example for this would be a hourglass like structure, where the central bottleneck can cause the mesh creation algorithms fail to properly map vertices to one another. The guiding hand of a human could resolve the problem during the mesh-creation. Other than this example the mesh representation makes this algorithm highly independent from the form of the molecular structure mapped.

CHAPTER 6

Conclusion

Although the initial goal was achieved there are still several ideas to test out and optimizations to be applied, for example we noted during development that for more attractive and variable, yet harder to use solutions, a texture could be applied to each triangle or the whole mesh, which would replace the barycentric calculation. We deem this a valid alternative, but it was discarded due to being deemed harder to control for the user.

As the distance from a molecule position to an edge is the deciding factor for the probability, sometimes bigger molecules like the ones in the capsid do overlap with the edge, but due to their size they still get removed. The element center is still so far away from the defined edge that it gets cut away. For this purpose a function could be implemented that allows the user to add a radius to the centroid, thus reducing the distance from the edge, possibly bettering the visualization. Furthermore a huge increase in performance will be achieved by picking better assignment algorithms. We imagine that ray-casting is a feasible solution for structures homeomorphic to spheres that are somewhat convex as casting towards the object is trivial (given the same source position in space) as a ray hits the geometry either when traveling towards the center or away from the center. In case it hits the geometry multiple times or in both travel directions this can be resolved by using the closest encounter as the projection point. This is particularly interesting in the Unity framework as it combines the steps of triangle-assignment, projection and barycentric coordinate calculation while being exceptionally cheap to use.

A topic that we did not discuss is the structure recognition as we settled early on this part of the solution to be entirely user-driven. We argue that it is possible to use certain mesh-creating algorithms combined with some additional research can provide valid structural identifiers, fully automating the process.

Applicability to other polygonal forms than triangles should be a given, considering that barycentric coordinates are generalizable for n-sided polygons[MBLD02]. Furthermore the resulting probability for each molecule could instead be used as opacity modifier or a totally different meaning could be assigned to them. SirTom is applicable to any point-cloud that represents a surface and should as such be applicable for use in all cases were a wire-frame representation of a point cloud is desirable

The possibility to optimize and alter the mesh creation and to swap out the molecule-mesh assignment for a more efficient one, as well as its broad applicability grants SirTom the possibility for growth and further development.

Bibliography

[ABCO+03] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T Silva. Computing and rendering point set surfaces. *IEEE Transactions on visualization and computer graphics*, 9(1):3–15, 2003.

[Baj82] Ruzena Bajcsy. *Three-Dimensional Object Representation*, pages 283–295. Springer Netherlands, Dordrecht, 1982.

[Ber17] Andrew Berry. private communication, 2017.

[BMR+99] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Claudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, Oct 1999.

[CCC+08] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. Meshlab: an open-source mesh processing tool. In *Eurographics Italian Chapter Conference*, volume 2008, pages 129–136, 2008.

[Che89] L Paul Chew. Guaranteed-quality triangular meshes. Technical report, Cornell University, 1989.

[CMS98] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers & Graphics*, 22(1):37 – 54, 1998.

[CP53] Robert B. Corey and Linus Pauling. Molecular models of amino acids, peptides, and proteins. *Review of Scientific Instruments*, 24(8):621–627, 1953.

[Dal08] John Dalton. *A New System of Chemical Philosophy ...* A New System of Chemical Philosophy. William Dawson & Sons, 1808.

[Din10] Dino Dini. Normalized tunable sigmoid function, 2010. Accessed: 2017.06.15.

[Eva93] Stephen V. Evans. Setor: Hardware-lighted three-dimensional solid model representations of macromolecules. *Journal of Molecular Graphics*, 11(2):134 – 138, 1993.

[FHJL88] Thomas E. Ferrin, Conrad C. Huang, Laurie E. Jarvis, and Robert Langridge. The midas database system. *Journal of Molecular Graphics*, 6(1):2 – 12, 1988.

[Fri09] Michael Friendly. Milestones in the history of thematic cartography, statistical graphics, and data visualization. 2009. Web document http://datavis.ca/milestones/.

[GKM+15] Sebastian Grottel, Michael Krone, Christoph Müller, Guido Reina, and Thomas Ertl. Megamol x2014;a prototyping framework for particle-based visualization. *IEEE Transactions on Visualization and Computer Graphics*, 21(2):201–214, Feb 2015.

[Gli16] Matthias Glinzner. Texturing of 3d objects using simple physics and equilateral triangle patches, 2016. 1.

[HDS96] William Humphrey, Andrew Dalke, and Klaus Schulten. Vmd: Visual molecular dynamics. *Journal of Molecular Graphics*, 14(1):33 – 38, 1996.

[JAAA+15] Graham T. Johnson, Ludovic Autin, Mostafa Al-Alusi, David S. Goodsell, Michel F. Sanner, and Arthur J. Olson. cellpack: a virtual mesoscope to model and visualize structural systems biology. *Nat Meth*, 12(1):85–91, Jan 2015. Article.

[KH13] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(3):29, 2013.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, pages 163–169, New York, NY, USA, 1987. ACM.

[LD08] Ares Lagae and Philip Dutré. A comparison of methods for generating poisson disk distributions. *Computer Graphics Forum*, 27(1):114–129, 2008.

[Lie03] Peter Liepa. Filling holes in meshes. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '03, pages 200–205, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[LRA+07] Wilmot Li, Lincoln Ritter, Maneesh Agrawala, Brian Curless, and David Salesin. Interactive cutaway illustrations of complex 3d models. *ACM Trans. Graph.*, 26(3), July 2007.

[LS80] Der-Tsai Lee and Bruce J Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Computer & Information Sciences*, 9(3):219–242, 1980.

[LTW04] Hong-Wei Lin, Chiew-Lan Tai, and Guo-Jin Wang. A mesh reconstruction algorithm driven by an intrinsic property of a point cloud. *Computer-Aided Design*, 36(1):1 – 9, 2004.

[MAPV15] Mathieu Le Muzic, Ludovic Autin, Julius Parulek, and Ivan Viola. cellview: a tool for illustrative and multi-scale rendering of large biomolecular datasets. In Katja Bühler, Lars Linsen, and Nigel W. John, editors, *Eurographics Workshop on Visual Computing for Biology and Medicine*, pages 61–70. EG Digital Library, The Eurographics Association, September 2015.

[MBLD02] Mark Meyer, Alan Barr, Haeyoung Lee, and Mathieu Desbrun. Generalized barycentric coordinates on irregular polygons. *Journal of graphics tools*, 7(1):13–22, 2002.

[MD03] Carsten Moenning and Neil A Dodgson. A new point cloud simplification algorithm. In *Proc. Int. Conf. on Visualization, Imaging and Image Processing*, pages 1027–1033, 2003.

[MFSD09] M.H. Müller-Fischer, S.D. Schirm, and S.F. Duthaler. Method of generating surface defined by boundary of three-dimensional point cloud, September 8 2009. US Patent 7,586,489.

[MMS+16] Mathieu Le Muzic, Peter Mindek, Johannes Sorger, Ludovic Autin, David Goodsell, and Ivan Viola. Visibility equalizer: Cutaway visualization of mesoscopic biological models. *Computer Graphics Forum*, 35(3):–, 2016.

[MN03] Niloy J. Mitra and An Nguyen. Estimating surface normals in noisy point cloud data. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, SCG '03, pages 322–328, New York, NY, USA, 2003. ACM.

[Pau31] Linus Pauling. The nature of the chemical bond. application of results obtained from the quantum mechanics and from a theory of paramagnetic susceptibility to the structure of molecules. *Journal of the American Chemical Society*, 53(4):1367–1400, 1931.

[PGK02] Mark Pauly, Markus Gross, and Leif P. Kobbelt. Efficient simplification of point-sampled surfaces. In *Proceedings of the Conference on Visualization '02*, VIS '02, pages 163–170, Washington, DC, USA, 2002. IEEE Computer Society.

[Reb93] S. Rebay. Efficient unstructured mesh generation by means of delaunay triangulation and bowyer-watson algorithm. *Journal of Computational Physics*, 106(1):125 – 138, 1993.

[SG11] David S Goodsell. Miniseries: Illustrating the machinery of life: Eukaryotic cell panorama. 39:91–101, 03 2011.

[VG05] Ivan Viola and Eduard Gröller. Smart visibility in visualization. In *Proceedings of EG Workshop on Computational Aesthetics Computational Aesthetics in Graphics, Visualization and Imaging*, pages 209–216, May 2005.