Varying Regions of Interest for Interactive Visualization of Macromolecules

Hugo Alexandre Pereira Fernandes
hugo.a.p.fernandes@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

November 2016

Abstract

This thesis explores a novel approach for interactive visualization of macromolecules, in which only regions of interest are rendered with high quality, leaving the remaining regions with lesser visual details. The algorithm uses both rasterization and ray tracing techniques to, together, generate the visualization of objects on the scene.

Our visualization is a new alternative to the existing molecular analysis software, more suitable to analyze areas of interest, by focusing those zones and granting better performance, hence allowing more interactivity capabilities.

Keywords: Macromolecules, solvent excluded surface, interactivity, ray tracing, implicit surfaces, level of detail, binding sites, GPU, regions of interest

1. Introduction

This project fits in a cooperative program from the University of Texas at Austin, University of Beira Interior and Instituto Superior Técnico, named A-MOP - Algorithms for Macro-Molecular Pocket Detection. Laboratory drug design based on structure consists in predicting which drugs act as ligands to a given protein. When this happens, the connected protein suffers changes on its functionality. To check if a ligand can connect to a protein, it is important to analyze the molecule binding sites. The problems are to correctly identify those sites and being able to visualize it interactively. There are already some binding sites detection algorithms such as Fpocket[14] and LIGSITE[12], but there is still lack of proper visualization techniques to this purpose due to the size of macromolecules, which causes excess of information and low performance.

1.1. Motivation

The motivation behind this work is based on the need to have better molecule interactive visualization techniques, in which algorithms that select zones of interest like cavities detection algorithms can be used to simplify the visualization. With such information, it is possible to reduce the excess of a complete molecule representation's data that are atoms farther away from selected areas. This can be achieved by granting different levels of detail depending on whether an atom belongs to the region of interest. This variant level of detail technique would also help users to focus their attention in zones that are considered important. The usage of low detailed surface in areas of the molecule also grants a faster rendering and therefore gaining a better frame rate and interaction capability.

A visualization that offer this capability of interaction and different levels of detail based on zones of interest would certainly help drug laboratories, since that drug search for a disease would be easier and faster by using the filtering and analyzing capabilities of this visualization.

1.2. Objectives and contributions

The goal of this thesis is to provide a level of detail based technique that allows the users to visualize regions of interest of a molecule. In this project, without loss of generality, the considered molecule features are protein cavities. The objective is to grant greater level of detail in those parts of the protein than remaining zones. The visualization should be interactive which requires a decent frame rate, about 25 frames per second, allowing the user to explore the molecule in real-time. To reach this goal, different techniques were studied, tested and benchmarked to determine which are the ones that allow such visualization.

The reference that we take as state of the art solution is Parulek et al. 2014 [19], which uses a hierarchy clustering that has a very high creation time. With our solution, we expect to avoid this problem and focus on the analysis of zones of interest,

while having the same goal of allowing the users to visualize and explore those high quality molecules interactively.

1.3. Thesis Outline

Section 1 addresses the motivation and the objectives of this thesis. Section 2 describes the necessary theoretical background to understand the molecular visualization field and the state of the art solutions in section 3 and the implementation of the novel proposed solution in section 4. Section 5 addresses the experimental visual and performance tests executed and its discussion. Section 6 compares the solution obtained with the initial thesis goals.

2. Background

Proteins are a big target of studies for different fields, including drug design. Some proteins are capable of binding to other molecules, affecting its functionality them. The connection made between the antibody proteins and the viruses are called binding sites. One point of interest is to study where those binding sites are and their features, so it can be found a protein that can neutralize the undesired molecule. Scientist concluded that often the binding sites are localized in cavities, which are, often, also named as pockets.

2.1. Molecular Representations

In molecular representation, usually it is used implicit and parametric surface representations. Implicit surfaces are represented by equation 1, where the domain is three dimensional and the result is one dimensional. When the equation result is equal to zero, the input point stands on the surface. When the result is lower than zero, the point is inside the surface and when it is higher than the isovalue k, the point is outside the surface. The last two cases can be swapped sometimes, depending on the intended interpretation.

$$f(x, y, z) = k \tag{1}$$

Parametric representations, on the other side, have the ability to generate points if there are input parameters. It can be represented as in equation 2, where the left side of the equation represents the resulting point and the right side is a calculation that works with two parameters u and v.

$$(x, y, z) = F(u, v) \tag{2}$$

The implicit function Van der Waals surface consists of using an implicit sphere centered in atom center position and with the radius corresponding to the atom. More complex molecular representations, such as Solvent Accessible Surface (SAS) [15] and Solvent Excluded Surface (SES) [10] are also used in more detailed visualizations. Either one of these two implicit surfaces are calculated by simulating rolling a sphere with the solvent radius on the already existing Van der Waals atoms spheres. As represented in figure 1, SAS is determined by the center of the solvent. On the other hand, SES uses the closest point to the Van der Waals surface.

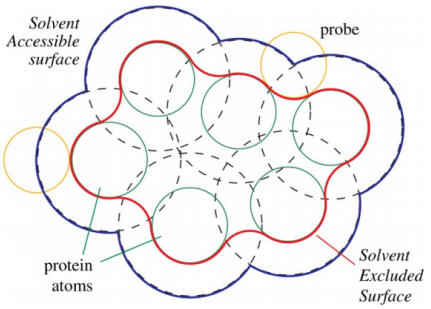

Figure 1: SAS - SES representation differences. Source: Krone et al. 2009 [13]

As described in Zhang and Feng work [26], general implicit distance fields can also be used for molecules. Those fields are based on functions that take into consideration the distance between a point and the atoms centers. The value returned by this function is compared to a threshold value to determine whether the implicit function involves the point. There are different functions for this purpose such as blobby molecules, metaballs and soft objects. Each one has different arguments and result in slightly different representations. The main reason to compare those functions is due to the different computational complexities.

2.2. Graphics visualization

In three dimensional graphical visualization, there are two main rendering methods: rasterization and ray tracing. Graphics pipelines that use rasterization are based on projecting three dimensional objects into the camera view plane. Rasterization is then performed, where pixels are drawn depending on the depth of each fragment. Rasterization rendering uses local illumination models, that consists in directly calculate surface illumination by using only light sources information.

The other rendering method is ray tracing [24], a method that is famous for being a more realistic approach and more detailed representation. Whitted ray tracing algorithm [25] is based on creating rays from the virtual camera position, that have direction to each near plane pixel. Those rays intersect objects in the scene, and new rays can be generated, such as shadow feelers, reflection and refractions rays, depending on the intersected object material. The resulting color is calculated using a shading model. In our work, we used Blinn-Phong shading [21][8].

When comparing these two methods, we can conclude that rasterization is usually faster since it is based on fast and simple transformations, but ray tracing provides a more realistic visualization since it follows an approach which is closer to reality and it also provides some features that are harder to get in rasterization such as global illumination. However, ray tracing suffers in terms of performance, mainly due to the number of rays that come out of the camera and their intersection calculations with the scene objects.

2.3. Implicit Surfaces

Rendering implicit surfaces visualization [9] is divided into two different sets, the direct and the indirect methods. Ray-tracing, particle-based, non-photo-realistic rendering and volume rendering are some examples of direct methods. Those direct techniques create a representation directly using the surface equation. The indirect methods involve a polygonization, which generates a mesh which can be rendered or used to other purposes, like 3d printing and 3d modeling since the mesh information can be stored.

When applied to high level of detail objects, polygonization loses its speed advantages and even fails to represent finer detailed objects. Unlike polygonization and rasterization solutions, ray tracing techniques are independent on implicit surface changes, at most acceleration structures need to be updated. In terms of quality and fidelity to the real surface, ray tracing often grants more realism and loyalty that may depend on some parameters on iterative ray marching methods or approximations.

2.3.1 Polygonization

In terms of polygonization techniques, as described in Araújo et al. 2015 [9] there are three types of approaches, the spatial decomposition, surface tracking and inflation and shrink-wrap.

The spatial decomposition consists on recursive space divisions and storing the cells that contain the surface. The surface tracking approach starts by using an initial point of the surface and iterates, querying the surface and generating a mesh of polygons. Finally, the inflation and shrink-wrap methods consists on having an initial volume that expands or shrinks until it covers closest to the shape of the surface.

Polygonization techniques are used to generate a mesh that represents implicit surfaces that can be rasterized in a fast way. The main problems of using this method are the need to compute the mesh every time the surface changes, less precision the real implicit surface and high memory requirements to highly detailed and large surfaces since it needs to store the mesh vertices data.

2.3.2 Ray tracing

As described in section 2.2, ray tracing technique grants a higher level of realism sacrificing performance. Its main source of overhead lays on intersection calculations between rays and scene objects. Since we are dealing with implicit surfaces, it is usually described by high complexity functions. There are two surface finding methods: analytic and iterative [23].

Analytic methods are based on equation solving, which is faster than the other two methods, but there are no mechanisms to solve to equations higher than forth order polynomials. Iterative methods are famous since they are independent of the function used to describe the implicit surface. It consists in testing values directly on the surface function until the result is close to zero. The problem associated to this technique is the difficulty on setting initial parameters.

John Hart in 1994 [11] introduced sphere tracing which is an iterative surface finding mechanism, that uses functions that return the minimal distance from a point to the surface to set the step size of the marching points. The algorithm consists in a loop where it is calculated the distance function giving as input the marching point. If the distance is lower than a threshold value, that determine the level of approximation to the zero, then it is returned an intersection. Otherwise, as the figure 2 shows, the step size is set by the distance returned by the function and the loop continues until the intersection is found. This iterative method variation guides the surface tracking, using the minimal distance to the surface, it is guaranteed that there are no misses in the search and even optimizing the number of iterations needed.

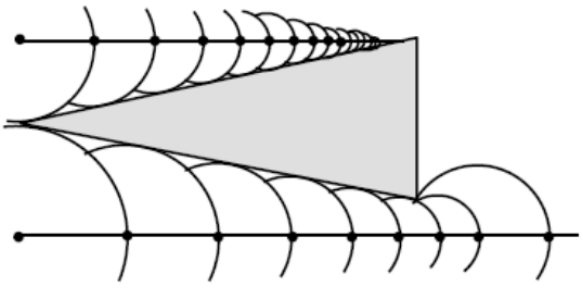

Figure 2: Hart 1994 [11]: Sphere tracing algorithm

2.4. Acceleration structures

Acceleration structures are used in ray tracing to minimize the amount of operations when calculating intersections [22]. In this project, there are some variations since it is being used point marching instead of using directly the rays. The scope of this project does not involve molecular dynamics, this

means that the positions of the atoms are constant during all execution time. Due to this assumption, the construction of the acceleration structures will not be address in great detail.

The acceleration structures used in this project were utilized to collect the nearest neighbor atoms around the point being tested in the sphere tracing marching. This is the major overhead in the ray tracing computation, affecting the capability of interaction of the user with the visualization. The acceleration structures used were Uniform Grid, Bounding Volume Hierarchy and K-d tree.

2.5. Screen-space ambient occlusion

In order to improve depth perception in a scene with an high amount of objects very close to each other, it was used a technique introduced by Martin Mittring [17], in Crysis 2 video game, named screen-space ambient occlusion (SSAO).

The main idea of the algorithm is to darken fragments which are surrounded by others that are closer to the camera. The term occlusion is used to quantify the amount of light that hit the fragment. An occluded fragment is hit by less light, while a less occluded is exposed to more light. In this project it was used the John Chapman [6] modified version of Crytek algorithm.

3. State of the art

This section is divided into two subsections: state of the art on molecular visualization and techniques that use level of detail.

3.1. Molecular visualization

Julius Parulek and Ivan Viola [20] proposed a solution to compute Solvent Excluded Surface (SES) representation without any precomputations. Due to this improvement, SES representation ray tracing grants some level of interactivity, allowing to change parameters values in real-time. Their solution generates a function that determines the minimal distance from a point to the surface. The method starts by collecting the closest atoms to the point being tested. Depending on the number of closest atoms and the intersection between pairs and triples of atoms, there is a different approach to build the final function. In the case where there are no atoms within the range, the resulting function is determined depending on the solvent radius. In case there is only one atom in the closest set, the function used is Van der Walls. When exactly two atoms are in the set, it is applied a toroidal implicit function, and when three or more atoms exist, it is used a more complex spherical triangle implicit function as illustrated in figure 3. This SES implementation was the one used in our work.

Although SES representation is seen as the most reliable molecular representation, Parulek and

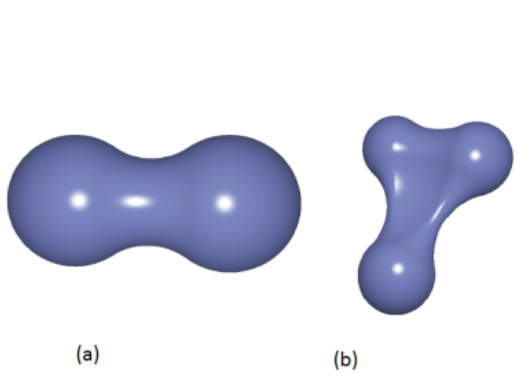

Figure 3: Parulek et al. 2012 [20]: (a) Toroidal implicit function; (b) Spherical triangle implicit function.

Brambilla [18] proposed a representation with visual results close to SES. The advantage of this new algorithm is that it has linear complexity, which is better than previous solution [20] cubic complexity. Since that there is still a significant visual difference between this representation and SES (which is considered by the community as one of the more suitable for molecular analysis), this solution was not used.

In 2010, Singh and Narayanan [23], proposed adaptive marching points algorithm, which is based on adapting marching point step size according to the distance to the surface. The first step is to get the two intersection points between the ray being processed and the near and far planes. Then, the starting point of the marching point algorithm is set to the near plane intersection point and the step length is set to a default value. A cycle is iterated, checking first if the marching point has not exceeded far plane intersection point. After that, a root finding method is performed in order check if there is a zero of the implicit function between the current marching point and the next one. If a zero is found, then this interval is isolated and solved. Otherwise, the marching point is incremented by using a function that depends on the value of the implicit surface and on whether there is a silhouette.

3.2. Level of detail based techniques

Parulek et al. 2014 [19] proposed a solution based on applying different implicit surface representations based on the distance to each atom. Atoms nearest to the camera are rendered with Solvent Excluded Surface representation, the next atoms are rendered with Gaussian kernels and the more distant atoms are rendered using spheres. Linear interpolation is used to connect the representations.

The more distant atoms are grouped in clusters to reduce memory requirements and to speed up the rendering process. This clustering, however, takes a long time to be created and it is a problem that we avoid in our solution.

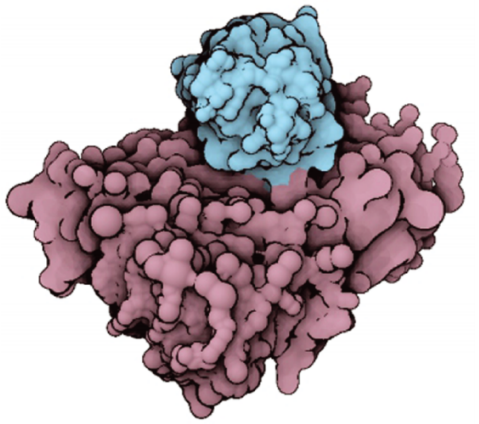

Figure 4: Parulek et al. 2014 [19]: Phospholipase bound to lipid membrane with 2808 clusters of atoms.

TexMol [7] is a molecular visualization software that also applies a level of detail based technique, although in a different way when compared to last method described. Molecular data is classified and generates an hierarchy. The lowest level of the hierarchy have individual atoms, which are represented with Van der Waals spheres. The second lowest level, residues, which include amino acids and nucleotides, are represented by the minimum bounding sphere that contains all the atoms that compose the structure. The third level has the molecular secondary structure, which contains helices, turns and sheets. Helices and sheets are represented with sets of cylinders and helices. Turns are displayed as the atoms cluster that compose it. The last and higher level are backbone chains that are represented as the set of atoms that define the backbone chain.

Lee et al. 2006 [16] described a polygonization level of detail technique that allows having good frame rate when rendering large size molecules. The first step of their approach consists in creating a set of polygons for each molecular model. Then, it is applied a molecular surface generation algorithm that returns simplified models of some parts of the molecule. The important molecular parts such as active sites are not subject to this simplification algorithm. Those models are classified and stored. The models to be used are determined by analyzing the running environment capability and by the distance between the camera and molecular models.

4. Implementation

This work explores a novel approach for interactive visualization of macromolecules, in which only regions of interest are rendered with high resolution, leaving the remaining regions with lesser visual details. The solution combines the advantages of polygonization and ray tracing. The set of atoms detected/classified as regions of interest are rendered using ray tracing and a high fidelity representation, such as Solvent Excluded Surface. The remaining atoms are rendered with low quality polygonization and rasterization, just to give a molecular context to the user.

The final implementation uses C++, freeglut 2.8.1-1.mp for MSVC [4], GLEW 1.11.0 [5] and CUDA Toolkit 7.5 [3].

The algorithm takes as input the whole molecule pdb file containing the data of the position an radius of the atoms. In order to select the regions of interest, other pdb files are parsed containing the identifiers of the atoms selected as important.

The rasterization technique handles the rendering of the atoms of least interest. Atoms that closer to the camera than a defined limit distance are rendered using mesh spheres and screen-space ambient occlusion is applied to help the user depth perception. The atoms that are farther away than the limit distance are rendered using billboards, which dark according to the distance to the scene camera.

Some data of the rasterization technique is stored in textures to optimize the sphere tracing algorithm. That data is the depth buffer and the color buffer used to render the zones of least interest. The influence surface of the regions of interest are also rendered in an apart framebuffer and its depth buffer is also stored in a texture for later use in the ray tracing step. The textures are made accessible to CUDA code by using CUDA-OpenGL interoperability functions.

The ray tracing technique is used to render the atoms in regions of interest, by using sphere t racing to track the solvent-excluded surface applied to those atoms. We used CUDA to exploit the modern GPUs capabilities for the ray tracing algorithm. For each ray, a thread is created to handle the computation of the corresponding pixel color. Threads are organized in rectangle blocks instead of linear blocks, since it attenuates divergence problems, increasing performance.

For each thread, the stored depth textures from rasterization step are queried in order to set the starting and maximum depth of the ray marching. The ROI spheres depth sets the initial depth, while the spheres and billboards depth is the maximum limit depth.

For each point of the sphere tracing algorithm, the closest neighbor atoms are retrieved and the

solvent-excluded surface function is calculated and if the returned value is lower than a defined threshold the surface is considered found, otherwise the point goes forward using the returned distance value. In the end, the pixel color is calculated using Blinn-Phong shading and it is stored in the color buffer to be finally rendered to the screen with the final result.

The sphere tracing algorithm uses two threshold values, one to check if the surface is considered found and the other for the Newton-Raphson termination. Those thresholds vary according to the user interaction. If the user is moving the camera, the thresholds get thicker and if the camera is stopped, the thresholds start to become thinner and stabilize after a set time interval.

5. Results

Our experimental tests were divided into two different methodologies: visual quality tests and performance tests. The visual tests are reported in section 5.1 and are focused in comparing visually the difference between a visualization that uses the same representation for the whole molecule and the result of using regions of interest to differ parts of the molecules. The performance tests in section 5.2 are used to evaluate our solution using concrete measures of performance such as time to render a frame and size of the acceleration structures. In the tables, T columns stand for time to render a frame and are measured in milliseconds and *FPS* stands for frames per second.

In all results that are displayed in the next sections, there are some parameters that are common between benchmark scenes which are: window resolution of 1280 by 720 pixels, solvent radius of 1.4, camera opening horizontal and vertical angles of $\frac{\pi}{4}$ and $\frac{ResY}{ResX} \times HorizontalAngle$, camera near plane at 0.05, camera far plane at 200, distance to convert spheres into billboards of 60.0, 10 neighbor atoms maximum limit while computing SES, the surface threshold varying between 0.05 and 0.2 and newton threshold varying between 0.005 and 0.2. The distance, position and threshold values are measured in Å.

Two benchmark scenes were used to test the solution. The benchmark scene *1AON* uses the molecule named "Crystral structure of the asymmetric chaperonin complex GroEL-GroES-(ADP)7" [1] with identifier 1AON. The whole molecule protein database file contains 58674 atoms. The second benchmark scene *1IGT* contains the "Structure of immunoglobulin" [2] with identifier 1IGT and has 12530 atoms.

In the benchmark scene *1AON*, the camera position is, in Å, $(x, y, z) = (184.090622, -71.322571, -26.695751)$ and orientation angles of the camera are $\alpha = 4.972095$ radians and $\beta = 0.53315$ radians. The α symbol is the camera horizontal rotation angle, while β is the camera rotation vertical angle.

The second benchmark scene *1IGT* has one test, with the camera in position $(x, y, z) = (49.140038, 1.700683, -65.313866)$ and view angles $\alpha = -0,6507368$ radians and $\beta = -0.1794062$ radians.

5.1. Visual quality tests

In this section, several images are displayed side by side, comparing representations of molecules that are rendered using solvent-excluded surface for the whole molecule and the visual result of rendering the molecule using zones of interest and distance to either use SES, mesh spheres or billboards representation.

5.1.1 Test 1AON

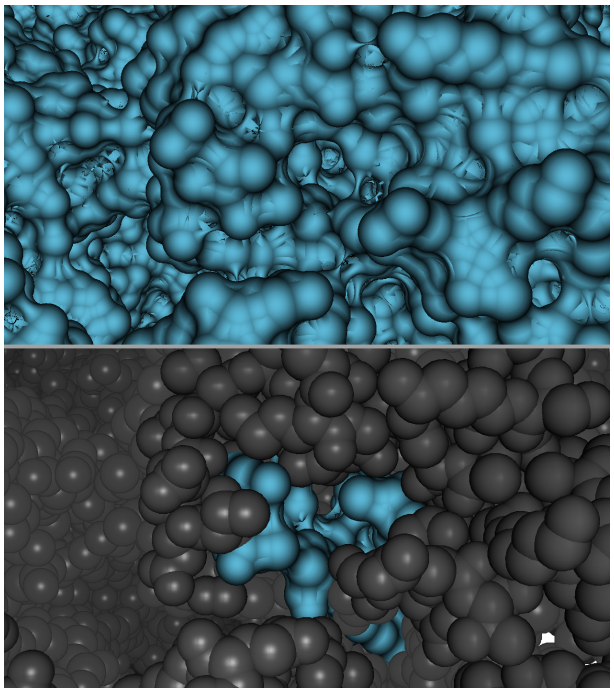

Figure 5: Visual comparison between rendering benchmark scene *1AON* with SES representation (above) and rendering using our level of detail based on zones of interest (below)

5.1.2 Test 1IGT

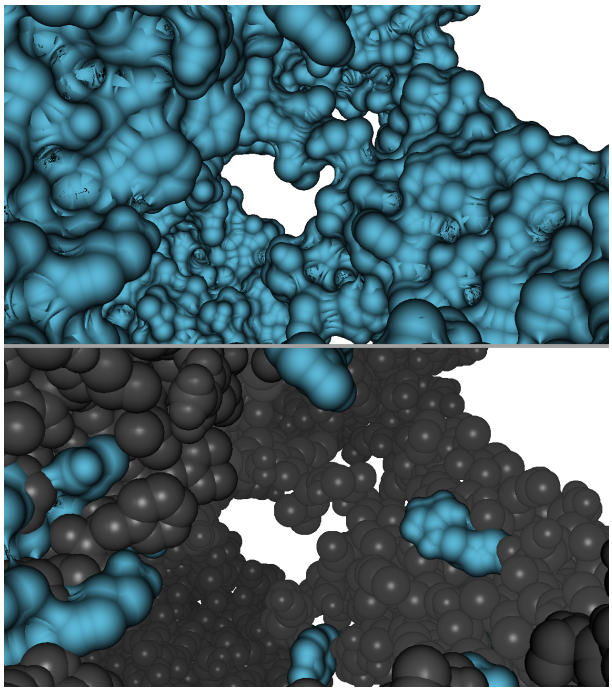

Figure 6: Visual comparison between rendering scene *1IGT* with SES representation (above) and rendering using our level of detail based on zones of interest (below)

5.2. Performance tests

In order to test the performance of the visualization, the benchmark scenes were tested in different machines. The first machine *M6000* is composed by a Intel i7-920 quadcore 2.67 GHz CPU, 6 gigabytes of RAM and a Nvidia Maxwell Quadro 6000 GPU. The machine *GTX970* is composed by a Intel i5-4690 quadcore 3.50GHz CPU, 8 gigabytes of RAM and a GPU Nvidia GTX 970 with 4 gigabytes of memory. Lastly, the computer *GTX980 Ti* has a Intel i7-6700 quadcore 3.40GHz CPU, 16 gigabytes of RAM and a Nvidia GTX980 Ti graphics processor.

5.2.1 Solution performance

In this section, the results of applying our solution of regions of interest to the benchmark scenes are described.

The molecule with identifier *1AON* has a total of 58674 atoms, 898 of which were selected by the regions of interest protein database files. The following list contains the resulting acceleration structures sizes of this scene:

- Uniform grid memory size: 45.4 megabytes
- BVH memory size: 0.07 megabytes
- K-d tree memory size: 0.03 megabytes

In terms of frame rate performance, the results are displayed in the table 1 for the test *1AON* using the thinnest thresholds and in the table 2 using the thickest thresholds.

Table 1: 1AON: Results with the thinnest thresholds

	M6000		**GTX970**		**GTX980Ti**	
	FPS	**T**	**FPS**	**T**	**FPS**	**T**
UG	21.7	46	20.4	49	30.5	33
BVH	15.9	63	13.3	75	21.1	47
K-d	16.4	61	13.7	73	21.7	46

Table 2: 1AON: Results with the thickest thresholds

	M6000		**GTX970**		**GTX980Ti**	
	FPS	**T**	**FPS**	**T**	**FPS**	**T**
UG	29.9	33	32.5	31	48.5	21
BVH	25.1	40	22.7	44	35.5	28
K-d	25.4	39	23.2	43	36.2	28

The molecule *1IGT* has a total of 12530 atoms, 159 of which are in regions of interest. The following list contains the resulting acceleration structures sizes:

- Uniform grid memory size: 11.69 megabytes
- BVH memory size: 0.01 megabytes
- K-d tree memory size: 0.01 megabytes

The performance outcome is in the table 3 using the thinnest thresholds and in the table 4 using the thickest thresholds.

Table 3: 1IGT: Results with the thinnest thresholds

	M6000		**GTX970**		**GTX980Ti**	
	FPS	**T**	**FPS**	**T**	**FPS**	**T**
UG	18.7	53	16.2	62	25.3	40
BVH	14.1	71	11.3	88	18.2	55
K-d	13.9	72	11.2	89	18.1	55

Table 4: 1IGT: Results with the thickest thresholds

	M6000		**GTX970**		**GTX980Ti**	
	FPS	**T**	**FPS**	**T**	**FPS**	**T**
UG	27.9	36	27.5	36	43.2	23
BVH	21.1	47	18.6	54	29.9	33
K-d	20.6	49	17.8	56	28.9	35

5.3. Full solvent-excluded surface versus regions of interest

Using the best setup of the ones tested in the performance section 5.2, the machine *GTX980Ti* and uniform grid as acceleration structure, the following tables describe the performance comparison between rendering the benchmark scenes using solvent-excluded surface for the whole molecule (full SES) and rendering the molecules using regions of interest (ROI).

The table 5 shows the comparison of the uniform grid size between full SES and ROI implementations.

Table 5: Uniform grid size comparison: Full SES vs ROI

	Full SES	**ROI**
1AON	429.66 Megabytes	45.44 Megabytes
1IGT	226.22 Megabytes	11.69 Megabytes

The table 6 shows the comparison of the frame rate between full SES and ROI implementations, using the thinnest thresholds.

Table 6: Frame rate: Full SES vs ROI (thinnest thresholds)

	Full SES		**ROI**	
	FPS	**T**	**FPS**	**T**
1AON	2.8	357	30.5	33
1IGT	3.3	303	25.3	40

The table 7 shows the comparison of the frame rate between full SES and ROI implementations, using the thickest thresholds.

Table 7: Frame rate: Full SES vs ROI (thickest thresholds)

	Full SES		**ROI**	
	FPS	**T**	**FPS**	**T**
1AON	4.7	213	48.5	21
1IGT	6.6	152	43.2	23

5.4. Discussion

The results obtained were within the expectations and the goals of the project. Using the GPU capabilities, the sphere tracing algorithm is processed at the same time for each pixel grants better performance than the sequential implementation. The mixing between the rasterization rendering and the ray tracing rendering was a challenging but successfully solved problem, with all the improvements done in the project evolution, the drawback of combining both techniques was abolished, keeping the advantages of smaller acceleration structures and the capacity of focusing the user's attention on the regions of interest.

5.4.1 Visual quality

The visual quality tests in section 5.1, the benchmark scenes that are rendered using SES for the whole molecule are very homogeneous and confusing. One of the goals of this work was for being able to select zones of interest in order to focus those important parts of the molecule, while leaving the rest of the molecular context. Using the additional files that create those regions of interest, the visual representation change depending on the importance of every atom to the user. Our solution used sphere tracing and SES to render the zones of interest, while the rest of the molecule is rendered with mesh spheres and billboards, depending on the distance of the atoms to the camera. Visually, the fact that the regions of interest are rendered with a more detailed representation and have a color that stands out, grabs the user attention and helps in handling the excess of information. In the future, there will be questionnaires to ask the molecular field specialists their evaluation of our solution.

5.4.2 Performance

Based on the performance results shown in the section 5.2, the overall conclusions are that the performance of the visualization is dependent on the region of interest atoms being rendered as demonstrated by the difference of the performance benchmark scenes in subsection 5.2.1. On other side, the frame rate scales well with the increase number of atoms, since that the great difference in the number of atoms between the benchmark scenes in tests *1AON* and *1IGT*, the frame rate is not drastically affected. The difference in the rendering time between the coarsest thresholds and the finnest thresholds is in average 22 milliseconds, granting smoother camera movements while the user is exploring the scene.

In terms of machines, the rating performance shows that *GTX970* is with lower results, followed by *M6000* and having *GTX980 Ti* has the best among the three machines. *GTX980 Ti* highlights in the results, having in every case considerably better frames per second and was chosen as the best machine for the last performance test in the section 5.3.

Uniform grid, as expected, is the acceleration structure with better results, since that there is no traversal, only a simple calculation to get the index of the grid in which a point is. It is the one that requires more memory space for a long margin but, even in the case of the larger tested molecule *1AON* the memory does not reach 50 megabytes, and the current GPUs have 2 gigabytes or more.

The bounding volume hierarchy and k-d tree have very similar results, since both suffer from the fact

that the influence atoms have much overlaps, forcing additional searches in the trees.

The scenarios in section 5.3 shows that displaying the molecule using the full solvent-excluded surface, the memory requirements increased substantially when compared with using regions of interest. When all the molecule is selected as solvent-excluded surface, the size of the acceleration structures increases, reducing the performance of the traversal and increasing the number of combinations computed in the sphere tracing algorithm, affecting negatively the overall performance. The solution without regions of interest also do not optimize the sphere tracing with the techniques described in implementation section 4.

6. Conclusions

After analyzing the existing solutions to render molecules, we came to the conclusion that there are two main types of rendering techniques used: polygonization combined with rasterization and ray tracing. Rasterization has performance as the main advantage, while ray tracing provides an higher level of realism. There are few solutions that use level of detail, and the ones that use it, use the level of detail mostly in the distance from the camera to the molecule. Most solutions do not use molecular features to affect their visualization.

In an attempt to provide a different solution that avoids some of the disadvantages enumerated before, we present a novel solution proposal. The proposed visualization uses molecular features to change the level of detail of molecular zones. It is used high quality ray tracing to render atoms near the zones of interest, leaving the rest of the atoms to be rendered with low quality polygonization and rasterization. The combination takes advantages of ray tracing and polygonization, while causing a difference between zones with high quality and zones with lower, focusing the user attention in zones with higher detail.

Additionally, the usage of both ray tracing and rasterization is exploited to get some optimizations. The depth of the fragments generated through rasterization are used in the sphere tracing algorithm to limit the depth in which the marching points track the surface, improving the overall frame rate.

6.1. Future Work

It is planned to make a questionnaire to molecular field specialists in order to get a more concrete feedback on the visual quality of our level of detail based on regions of interest solution.

In the future, it would be interesting to adapt the visualization to support molecular dynamics. This would imply that all the acceleration structures had a very short construction time, so that it could be rebuilt every frame and updated to the CUDA code. The support for molecular dynamics would be interesting to analyze a full sequence of molecular features like the binding action between a protein and a ligand, to evaluate how the atoms change during this phenomenon.

A second idea is to create a user interface, that would certainly complement the visualization and making it more useful to the users. Features like selecting atoms to get information about them or to convert the neighbor region to be a region of interest would make the visualization more complete and flexible.

Acknowledgements

I would like to thank my thesis supervisor professor João Madeiras Pereira, co-supervisor doctor Daniel Simões Lopes and to my colleague Jorge Martins. I would also like to acknowledge A-MOP: FCT Project UTAP-EXPL/QEQ-COM/0019/2014, the financial supports by national funds through the Portuguese FCT with references IT-MEDEX PTDC/EEISII/6038/2014 and UID/CEC/50021/2013. I would also like to gratefully acknowledge the support of NVIDIA corporation.

More personally, I would like to appreciate my family, closest friends and girlfriend support, you are all very special to me.

References

[1] 1AON: Crystral structure of the asymmetric chaperonin complex GroEL-GroES-(ADP)7. `http://www.rcsb.org/pdb/explore.do?structureId=1AON`. Accessed: 11-10-2016.

[2] 1IGT: Structure of immunoglobulin. `http://www.rcsb.org/pdb/explore.do?structureId=1IGT`. Accessed: 11-10-2016.

[3] CUDA Toolkit. `https://developer.nvidia.com/cuda-toolkit`. Accessed: 05-10-2016.

[4] freeglut: The Free OpenGL Utility Toolkit. `http://freeglut.sourceforge.net/`. Accessed: 05-10-2016.

[5] GLEW: The OpenGL Extension Wrangler Library. `http://glew.sourceforge.net/`. Accessed: 05-10-2016.

[6] SSAO tutorial. `http://john-chapman-graphics.blogspot.pt/2013/01/ssao-tutorial.html`. Accessed: 23-09-2016.

[7] C. Bajaj, P. Djeu, V. Siddavanahalli, and A. Thane. Texmol: interactive visual exploration of large flexible multi-component molecular complexes. In *Visualization, 2004. IEEE*, pages 243–250, Oct 2004.

[8] J. F. Blinn. Models of light reflection for computer synthesized pictures. *SIGGRAPH Comput. Graph.*, 11(2):192–198, July 1977.

[9] B. R. de Araújo, D. S. Lopes, P. Jepp, J. A. Jorge, and B. Wyvill. A survey on implicit surface polygonization. *ACM Comput. Surv.*, 47(4):60:1–60:39, May 2015.

[10] J. Greer and B. L. Bush. Macromolecular shape and surface maps by solvent exclusion. *Proceedings of the National Academy of Sciences*, 75(1):303–307, 1978.

[11] C. J. Hart. Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10):527–545, 1996.

[12] M. Hendlich, F. Rippmann, and G. Barnickel. Ligsite: automatic and efficient detection of potential small molecule-binding sites in proteins. *Journal of Molecular Graphics and Modelling*, 15(6):359 – 363, 1997.

[13] M. Krone, K. Bidmon, and T. Ertl. Interactive visualization of molecular surface dynamics. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1391–1398, Nov 2009.

[14] V. Le Guilloux, P. Schmidtke, and P. Tuffery. Fpocket: An open source platform for ligand pocket detection. *BMC Bioinformatics*, 10(1):1–11, 2009.

[15] B. Lee and F. Richards. The interpretation of protein structures: Estimation of static accessibility. *Journal of Molecular Biology*, 55(3):379 – IN4, 1971.

[16] J. Lee, S. Park, and J. i. Kim. View-dependent rendering of large-scale molecular models using level of detail. In *2006 International Conference on Hybrid Information Technology*, volume 1, pages 691–698, Nov 2006.

[17] M. Mittring. Finding next gen: Cryengine 2. In *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, pages 97–121, New York, NY, USA, 2007. ACM.

[18] J. Parulek and A. Brambilla. Fast blending scheme for molecular surface representation. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2653–2662, Dec 2013.

[19] J. Parulek, D. Jönsson, T. Ropinski, S. Bruckner, A. Ynnerman, and I. Viola. Continuous levels-of-detail and visual abstraction for seamless molecular visualization. *Computer Graphics Forum*, 33(6):276–287, 2014.

[20] J. Parulek and I. Viola. Implicit representation of molecular surfaces. In *Visualization Symposium (PacificVis), 2012 IEEE Pacific*, pages 217–224, Feb 2012.

[21] B. T. Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, June 1975.

[22] E. Reinhard, B. Smits, and C. Hansen. *Dynamic Acceleration Structures for Interactive Ray Tracing*, pages 299–306. Springer Vienna, Vienna, 2000.

[23] J. M. Singh and P. J. Narayanan. Real-time ray tracing of implicit surfaces on the gpu. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):261–272, March 2010.

[24] K. Suffern. *Ray Tracing from the Ground Up*. A. K. Peters, Ltd., Natick, MA, USA, 2007.

[25] T. Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349, June 1980.

[26] X. Zhang and C. Bajaj. Extraction, quantification and visualization of protein pockets. *Computational systems bioinformatics / Life Sciences Society. Computational Systems Bioinformatics Conference*, 6:275—286, 2007.