Parallel Computation and Interactive Visualization of Time-varying Solvent Excluded Surfaces

Michael Krone
Visualization Research Center (VISUS)
University of Stuttgart
Allmandring 19
70569 Stuttgart, Germany
krone@vis.uni-stuttgart.de

Carsten Dachsbacher
Computer Graphics
Karlsruhe Institute of Technology (KIT)
Am Fasanengarten 5
76131 Karlsruhe, Germany
dachsbacher@kit.edu

Thomas Ertl
Visualization and Interactive Systems Group (VIS)
University of Stuttgart
Universitätsstraße 38
70569 Stuttgart, Germany
ertl@vis.uni-stuttgart.de

ABSTRACT

Molecular dynamics simulations are a principal tool in the study of molecules. The efficient investigation of the structure and dynamics of the simulated molecular systems benefits from interactive high-quality visualizations. Molecular surfaces are among the most common representations for visual analysis of the properties of these molecules, especially when studying proteins and other biomolecules. In this paper we evaluate the suitability of parallel graphics hardware (GPUs) for interactive computation and visualization of the Solvent Excluded Surface. This surface consists of spherical and toroidal geometric primitives, which can be rendered using ray casting on the GPU, thereby obtaining unsurpassed visual quality at interactive frame rates. The key, however, is the efficient calculation of the Reduced Surface that defines the occurrence and location of these primitives. The Reduced Surface can be computed quite efficient and allows for a partial update of the surface in regions where changes occur. However, the original algorithm to construct the Reduced Surface cannot be parallelized straightforward. We introduce a novel, parallel algorithm, leveraging programmable GPUs, for computing the Reduced Surface. This allows the high-quality rendering and interactive exploration of complex molecules and time-varying molecular datasets.

Categories and Subject Descriptors

I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Curve, surface, solid, and object representations*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Visible line/surface algorithms*; J.3 [**Computer Applications**]: Life And Medical Sciences—*Biology and genetics*

General Terms

Molecular Visualization, Isosurfaces, Point-based Data

Keywords

Time-varying Data, GPU, Ray Casting, Surface Extraction

1. INTRODUCTION

Molecular surfaces are among the fundamental representations in molecular graphics. They are most helpful for visualizing the phenomena appearing at the boundaries of molecules. These include among others docking, the analysis of interactions and assemblies, and solution.The *Solvent Excluded Surface* (SES) is well-suited for these purposes as it allows to explore all of the aforementioned phenomena. Therefore, this surface representation is commonly used in visualization applications. However, its computation is time-consuming and only possible at interactive rates for moderately large molecules. The SES ist often applied for the analysis of molecular dynamics simulations. Since the MD trajectories can contain up to several ten thousand frames, precomputation of the SES for all frames, like it is done in some available molecular viewers to gain interactive frame rates, is not feasible. To enable the rendering of such large trajectories at interactive frame rates, a method which maintains the SES out-of-core is essential.

The SES can be computed analytically by extracting the *Reduced Surface* [8]. In this paper we address the problem of computing the Reduced Surface in parallel and present a novel algorithm that leverages the computational power of modern, programmable graphics hardware. It is to be expected that the computing power of CPUs in the future will rather scale with the number of cores than by increasing the clock rates. Accordingly, parallel algorithms will be necessary to benefit from this progress. GPUs already showed enormous advancements over the last years. Thereby, modern GPUs provide a massivly parallel architecture for general computations.

2. BASICS AND PREVIOUS WORK

The *Solvent Excluded Surface* (SES), which goes back to Richard's *Smooth Molecular Surface* [7], is the most important molecular surface. It is defined as the boundary of the union of all possible probes not intersecting any atom of the molecule [4] and can be computed using a rolling ball algorithm: Consider a sphere of a certain radius—determined by the Van-der-Waals (VdW) radius of a solvent atom—probing the VdW surface of the molecule. The surface of the probe traces out the SES (Fig. 1). The SES consists of three different geometric primitives (Fig. 2) traced out by the probe while rolling over the VdW surface:

Spherical patches occur when the probe is rolling over the surface of a single atom and has no contact with any other sphere. All produding parts of the atom's VdW surfaces are part of the SES.

Toroidal patches are formed when the probe has contact to two atoms and rotates around the axis connecting the atom centers. The probe traces out a torus whose inner surface is part of the SES.

Spherical triangles appear when the probe is simultaneously in contact with three or more atoms. The interior, concave part of the embedded probe sphere is part of the SES.

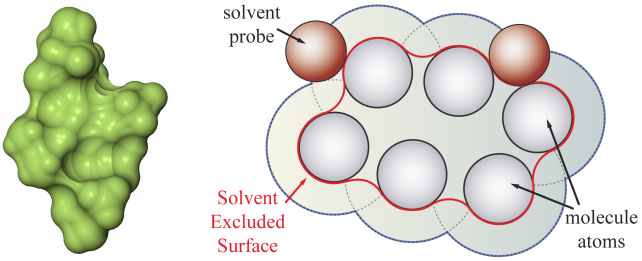

Figure 1: Left: SES; Right: Schematic of the SES, defined by a probe (shown in two sample positions) rolling over the atoms.

The analytic computation of the SES has first been introduced by Connolly [1]. Various methods have been introduced to speed up the computation of the SES: Alpha-shapes [3] are generalizations of the convex hull. Totrov and Abagyan [10] proposed a local contour-buildup algorithm. Varshney et al. [11] developed a parallel algorithm for extracting the SES on the CPU. The Reduced Surface (RS), on which the explanations in this paper are based, was developed by Sanner et al. [8, 9]. Recently, we presented a GPU-based ray casting that allows the visualization of the SES for large trajectories at interactive frame rates by using the RS [6].

Available molecular viewers like VMD [5] or PyMOL [2] are decompositing the SES into triangles for rendering. The advantage of this is that graphics hardware is designed for fast rendering of triangles. However, a high amount of storage is needed and visual artifacts are visible in close-up view even for high numbers of triangles. Ray casting provides high rendering quality, which is completely independent of the viewport since the exact mathematical description of the surface is used for image synthesis. It is particularly feasible since the SES is subdivided into geometric shapes whose surfaces are described by implicit functions.

3. REDUCED SURFACE COMPUTATION

The RS is a triangle mesh where each vertex, edge, and face corresponds to exactly one of the three geometric primitives—that is, the surface patches—of the SES (Fig. 2): When the probe is in a fixed position between three atoms, the centers of these atoms are the vertices of a triangle, which is called a *face of the RS* (RS-face). The edges of this triangle are called *edges of the RS* (RS-edges), while the centers of the related atoms are called *vertices of the RS* (RS-vertices). The RS is only valid for a specific probe radius r_p, if the radius is changed, it has to be recomputed. The algorithm to compute a RS as described by Sanner is sequential, building up the RS triangle by triangle. We briefly recapitulate this algorithm to depict the differences to our parallel algorithm. For a more detailed description we refer to Sanner's original work [8, 9].

3.1 Sequential Reduced Surface Computation

The first step is to find an initial RS-face, where the probe is in contact with three atoms and does not intersect any atom. Such a probe position can be found by searching the leftmost atom $a_{min(x)}$ and computing the neighborhood $N(a_i)$ of $a_{min(x)}$. The neighborhood of an atom a_i is defined as all atoms which can be touched by a probe being in contact with a_i. This includes all atoms which are inside a sphere with center a_i and radius $r_n = r_{a_i} + r_p + max(r_{a_j})$, where r_{a_i} is the radius of the atom a_i, r_p is the radius of the probe, and $max(r_{a_j})$ is the maximum atom radius. For all triples $a_{min(x)}$, a_i, a_j with $a_i, a_j \in N(a_{min(x)})$, a potential RS-face is computed and the probe position for this RS-face is tested for intersections with all other atoms in the neighborhood. The first detected probe position with no intersections determines the initial RS-face.

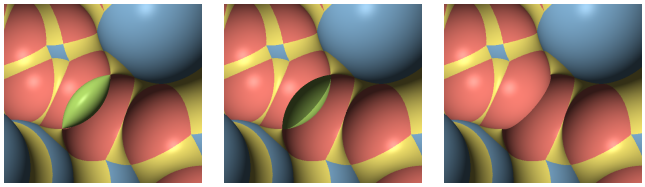

Figure 3: Different types of singularities: (a) Spindel torus, (b) Intersecting spherical triangles. The singularities are marked green and have to be cut off in the final image as shown in (c).

For all three RS-edges of the initial RS-face, an adjacent RS-face can be computed by comparing all potential adjacent RS-faces that share this RS-edge. Therefore, the probe defining the initial RS-face is rotated around the RS-edge until it hits another atom, denoting that the probe is again in a fixed position. In practice this can be done by computing the neighborhood of the RS-edge $N(t_{ij})$, where t_{ij} is the center of the torus being the trace of the probe rotating around the RS-edge's atoms a_i and a_j. For each potential RS-face a_i, a_j, a_k with $a_k \in N(t_{ij})$ the angle between the probe position $p_{ijk'}$ of the initial RS-face and the new probe position p_{ijk} is calculated. The potential RS-face with the least angle is the actual RS-face belonging to the RS. This operation is called the *treatment of an RS-edge* [9]. No further intersection tests have to be done for the probe of the newly generated RS-face. The RS is sequentially constructed by treating all new RS-edges of each RS-face as described above. The algorithm stops when all RS-edges are treated and the RS is a closed surface.

The SES can suffer from undesirable (self-)intersections shown in Fig. 3. These *singularities* must be identified and removed in a subsequent step to ensure the correct appearance of the SES. There are two kinds if singularities: spindle tori that occur when the outer radius of a torus is greater than the inner radius (Fig. 3(a)) and intersecting spherical triangles (Fig. 3(b)). While spindle torii can be identified by a single comparison of the two radii, the intersecting spherical triangles require a more elaborate treatment: All probes in fixed positions intersecting a spherical triangle have to be found.

3.2 Parallel Reduced Surface Computation

In this section we describe our parallel algorithm for the computation of the RS on programmable graphics hardware. As explained in Section 3.1, the RS algorithm was originally designed to run iteratively and therefore sequential. The first step to design a parallel variant of this algorithm is to devise a computation scheme which fits the architecture of modern graphics hardware. The GPU's massive computation power can be exploited best when starting many independent computations (threads) at once which then run in parallel without communicating with each other. In this case, the independent computations are the individual determination of an atom's neighborhood and probing its neighbor atoms. However, in order to keep the computation time at a minimum we need to restrict these

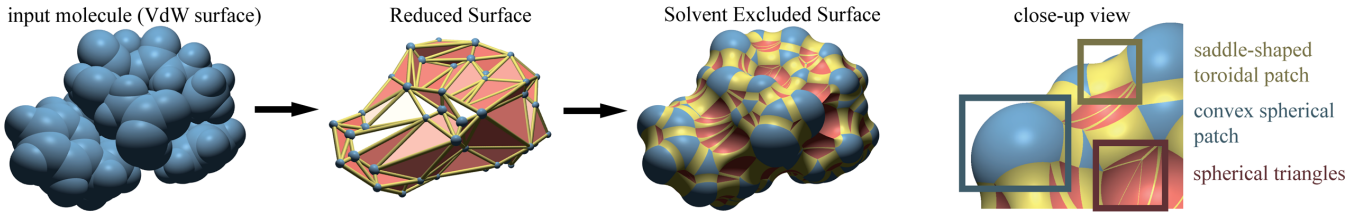

Figure 2: Visualization of the SES: Our parallel algorithm takes the molecule atoms as input and computes the RS for the visible part of the molecule. Every face, edge, and vertex of the RS corresponds to one of the geometric primitives forming the SES (right).

computations to those atoms that lie at the molecule's boundary and thus are part of the reduced surface. Therefore, we determine the visible atoms by using standard graphics hardware functionality and output sensitive visibility determination.

After having detected all visible atoms we launch the RS procedure as described in Section 3.1 for each of them on the GPU. Each instance of the computation first determines the neighborhood N_i of a visible atom. By using a coarse regular grid as spatial index structure, we greatly accelerate the neighborhood search.

A visible atom a_i can now form a triangle with any pair of neighboring atoms $a_j, a_k \in N_i$. For all these atom triples, we determine if the probe can be in contact with all three atoms. If this is the case, the triangle is a potential RS-faces and we compute the two possible probe locations. These two probe positions are then tested for intersections with the remaining atoms in the neighborhood. The triangle (a_i, a_j, a_k) can only be a potential RS-face if at least one of its probe positions does not intersect any of these other atoms. A large fraction of these potential RS-faces, however, is not part of the RS, because they are located in its interior. Similarly to the first step, we perform another visibility determination pass now detecting the faces that actually belong to the RS. The output of this step represent the visible part of the RS of the molecule and can be used to obtain the geometric primitives of the SES. However, due to the concave nature of two of the geometric primitives, the final geometric primitives of an occluded RS-face may be visible (see Fig. 2). Futhermore, we have to handle the singularities referred in Section 3.1. To remove the second type of singularities—the intersecting spherical triangles—the adjacent spherical triangles have to be found for each spherical triangle that is generated. However, these adjacent spherical triangles may be occluded by other parts of the SES and therefore not be identified as part of the RS during the visibility determination. By computing only the visible part of the RS, the singularity handling would be incomplete and thus add an additional computation step to search explicitly for occluded RS-faces adjacent to visible RS-faces. We only have to search for this occluded RS-faces at bordering RS-faces. A bordering RS-face has at least one RS-edge, which is not connected to a second RS-face. The adjacent occluded RS-face to this visible RS-face $F_{visible}$ can be found by computing the angle between $F_{visible}$ and all other potential RS-faces sharing this edge. The RS-face with the least angle is the RS-face that is required for the primitive intersection. Note that this step is executed in parallel for all bordering RS-Faces on the GPU. All RS-faces found in this step are also marked as visible RS-faces.

Now that we have computed the visible part of the RS, only a last step prior to the ray casting of the geometric primitives has to be executed: the singularity handling for the spherical triangles to ensure a correct rendering of the SES. For each RS-face marked as visible,

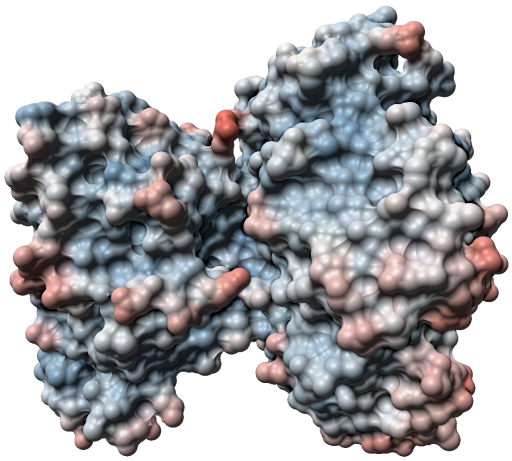

Figure 4: A protein (PDB-ID: 1TII) rendered with our method and colored according to the B-factor.

we use the Geometry Shader to compute the geometric primitives and generate a point sprite for each one (later used for initiating the GPU ray casting). The detection of intersecting spherical triangles for singularity handling (Fig. 3(b)) is done by computing the neighboring probes for RS-faces. For each probe in a fixed position we store all intersecting probes. The intersecting probes, which possibly lead to singularities, are stored in a texture to make them accessible via Fragment Shaders for the subsequent GPU ray casting. During the ray casting, each spherical triangle is tested for intersection with these probes.

4. RENDERING

In this section, we recapitulate the important details about the GPU ray casting used to render the three different primitives of the SES. Please refer to [6] for a more elaborate description. The general idea of ray casting is the same as of ray tracing: For each fragment—that is, for each pixel of the final image—a ray from the camera through this fragment is computed. This ray is then tested for intersection with all objects in the scene (in our case, the geometric primitives of the SES). If an intersection occurs, the color of the fragment can be determined due to the object's color and the light.

For the spherical patches (colored blue in Fig. 2), the whole sphere can be rendered, since those parts of the sphere that do not belong to the SES are lying in its interior. The ray-sphere-intersections are the roots of a quadratic function. Ray casting a spherical triangle (colored red in Fig. 2) is similar to ray casting a sphere with additional three cutting planes. For correct singularity handling, the spherical triangle is also cut with all intersecting probes to remove the undesired parts (Fig. 3(b)). The rendering of the toroidal

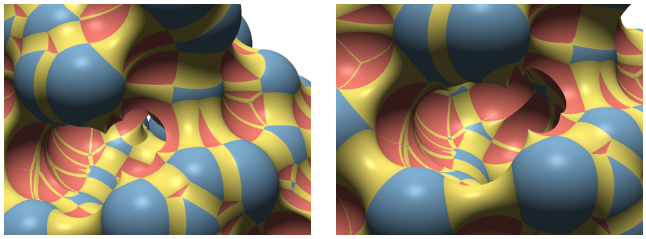

Figure 5: The imperfect singularity handling observable at the center of the cavity (left) disappears when viewing from a slightly different point of view (right).

patches (colored yellow in Fig. 2) requires root finding of a fourth-order polynomial. The toroidal patch belonging to the SES is the inner part of the torus located between the two atom spheres. This part of the torus is enclosed by a sphere, as described in [6]. Only the parts of the torus lying within this sphere belong to the SES and are rendered. The singularity handling for spindle tori can be reduced to a second simple sphere intersection test [6].

5. RESULTS AND DISCUSSION

Our algorithm has been designed having the possibilities and limitations of graphics hardware in mind to leverage their considerable parallel computational power. Furthermore, the applied GPU ray casting is an advantageous alternative to traditional 3D-graphics using tessellation, since it generates pixel-accurate visualizations at interactive frame rates. All computations executed on the GPU are implemented as GLSL shaders.

The parallelized computation of the SES presented in this paper yields a comparable performance to our recent work [6]. However, our novel method recomputes the entire visible part of the molecular surface and is not restricted to piecewise updates. Thus we can visualize time-dependent data, as well as change the probe size without any impact on the rendering performance. Table 1 presents the performance of our method. The measurements were done on an Intel i7 2.66 GHz with an Nvidia Geforce GTX285. Fig. 4 shows one of the test data sets rendered with our method. The most expensive steps in our algorithm are the computation of the potential RS-faces and the determination of adjacent, occluded RS-faces.

Note that a view-dependent computation also has inherent limitations. Consider the example shown in Fig. 5: Very deep concavities can be missed by both the visibility determination pass and the following search for adjacent, occluded triangles. Although the missed RS-face is not visible, its concave spherical triangle might intersect with visible ones, and thus missing it causes rendering errors. Fortunately this case is very rare and mostly occurs for small probes. The prerequisite for such rendering errors to occur is that the SES has to form a thin "wall" consisting of two layers of atoms, which are accessible to the solvent, which is not very commonly found. In addition, the rendering errors disappear in the majority of cases when viewing the molecule from a slightly different angle, due to the view-dependent nature of our algorithm (Fig. 5).

Table 1: Columns 1–4 show the timings for the stages of our algorithm in seconds: (1) determine visible atoms, (2) determine neighbors and compute potential RS-faces, (3) determine visible RS-faces and adjacent, occluded RS-faces, (4) extract geometric primitives and determine adjacency of visible RS-faces.

PDB-ID	#Atoms	fps	1	2	3	4
1A3I	$\sim$ 130	68	0.002	0.006	0.002	0.001
1CRN	$\sim$ 330	31	0.002	0.017	0.003	0.001
1RWE	$\sim$ 1,000	16	0.002	0.031	0.007	0.010
1J4N	$\sim$ 1,850	7	0.002	0.050	0.007	0.073
1VIS	$\sim$ 2,500	5	0.002	0.078	0.011	0.076
1TII	$\sim$ 5,500	3	0.003	0.172	0.033	0.125

6. CONCLUSIONS

In this paper we described a view-dependent, parallel algorithm for computing Reduced Surfaces directly on the GPU. This supports the interactive exploration of dynamic datasets such as molecular dynamics simulation data or the interactive adjustment of the probe size to consider different solvent molecules. To our knowledge, this is the first approach to compute the SES in parallel with an algorithm specifically designed to run on the GPU. Our method achieves comparable speed to recent work, however, it allows for fully dynamic data and is not restricted to static or only partially updated data sets. We believe that a parallel algorithm is better suited for future architectures. There is a strong trend in hardware development towards parallelization, not only regarding graphics cards, which constantly become faster, but also for modern multi-core CPUs. Our prototypical implementation qualifies as a proof of concept for the suitability of using the GPU to compute the SES.

7. ACKNOWLEDGMENTS

The authors wish to thank Sebastian Grottel for providing the visualization framework. This work is partially funded by the DFG as part of the Collaborative Research Center SFB 716.

8. REFERENCES

[1] M. L. Connolly. Analytical Molecular Surface Calculation. *Journal of Applied Crystallography*, 16:548–558, 1983.

[2] W. L. DeLano. *The PyMOL Molecular Graphics System*. DeLano Scientific, Palo Alto, CA, USA, 2002. http://www.pymol.org.

[3] H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Trans. Graph.*, 13(1):43–72, 1994.

[4] J. Greer and B. L. Bush. Macromolecular shape and surface maps by solvent exclusion. In *Proceedings of the National Academy of Science*, pages 303–307, Jan 1978.

[5] W. Humphrey, A. Dalke, and K. Schulten. VMD – Visual Molecular Dynamics. *Journal of Molecular Graphics*, 14:33–38, 1996.

[6] M. Krone, K. Bidmon, and T. Ertl. Interactive Visualization of Molecular Surface Dynamics. *IEEE Transactions on Visualization and Computer Graphics*, 15(6), 2009.

[7] F. M. Richards. Areas, Volumes, Packing, and Protein Structure. *Annual Review of Biophysics and Bioengineering*, 6(1):151–176, 1977.

[8] M. Sanner. *Sur la modélisation des surfaces moléculaires*. PhD thesis, Université de Haute-Alsace, France, 1992.

[9] M. F. Sanner, A. J. Olson, and J.-C. Spehner. Reduced Surface: An Efficient Way to Compute Molecular Surfaces. *Biopolymers*, 38(3):305–320, Dec 1996.

[10] M. Totrov and R. Abagyan. The Contour-Buildup Algorithm to Calculate the Analytical Molecular Surface. *Journal of Structural Biology*, 116:138–143, 1995.

[11] A. Varshney, F. P. Brooks, and W. V. Wright. Linearly Scalable Computation of Smooth Molecular Surfaces. *IEEE Computer Graphics and Applications*, 14(5):19–25, 1994.