Eurographics Conference on Visualization (EuroVis) (2012)
M. Meyer and T. Weinkauf (Editors)

Fast Visualization of Gaussian Density Surfaces for Molecular Dynamics and Particle System Trajectories

Michael Krone[1], John E. Stone[2], Thomas Ertl[1], Klaus Schulten[2,3]

[1]Visualization Research Center, University of Stuttgart, Germany
[2]Beckman Institute, University of Illinois at Urbana-Champaign, USA
[3]Department of Physics, University of Illinois at Urbana-Champaign, USA

Abstract

We present an efficient algorithm for computation of surface representations enabling interactive visualization of large dynamic particle data sets. Our method is based on a GPU-accelerated data-parallel algorithm for computing a volumetric density map from Gaussian weighted particles. The algorithm extracts an isovalue surface from the computed density map, using fast GPU-accelerated Marching Cubes. This approach enables interactive frame rates for molecular dynamics simulations consisting of millions of atoms. The user can interactively adjust the display of structural detail on a continuous scale, ranging from atomic detail for in-depth analysis, to reduced detail visual representations suitable for viewing the overall architecture of molecular complexes. The extracted surface is useful for interactive visualization, and provides a basis for structure analysis methods.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Boundary representations

1. Introduction

Particle-based simulations are a widely used tool in many fields. Molecular dynamics (MD) simulations for example are applied to simulate biomolecules like proteins. The output of these simulations are often very large, time-dependent data sets composed of hundred thousands of time steps and millions of particles per frame. Visualization is a fundamental tool for interpreting the results. Instead of visualizing the particles as individual spheres, surface representations are often advantageous since many interesting phenomena occur at the boundaries between particle complexes.

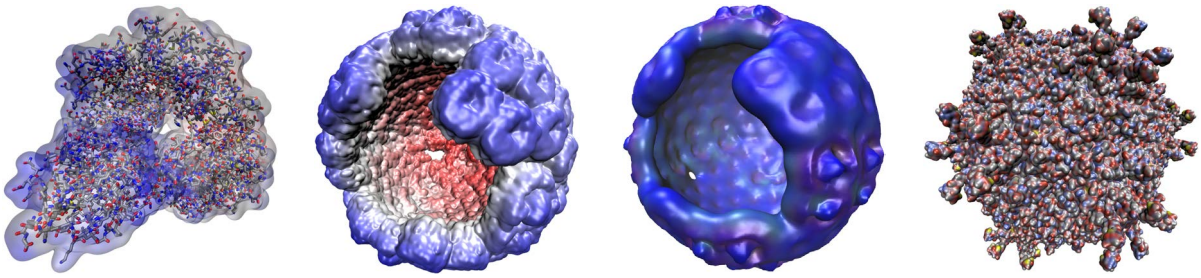
We present a fast method to visualize smooth surfaces for very large particle data sets. The algorithm is based on the Metaball definition [Bli82]. Our method makes use of the tremendous computing power offered by modern graphics processing units (GPU). The resulting surface can be used to model the analytically defined molecular surface very closely. The representation can also be adapted interactively to create a coarser, more approximate surface that is often desirable for larger numbers of particles, where atomic detail is not necessary. The surface can be colored to display per-atom biochemical attributes which is crucial for comprehensive analysis. The outline of our algorithm is quite simple: First, a uniformly spaced density map is calculated from the particles. Next, the surface is extracted using the Marching Cubes (MC) algorithm [LC87]. The extracted surface can be used for visualization as well as for further analysis.

2. Related Work

The Solvent Excluded Surface (SES) [Ric77, Con83, TA95] is defined by a spherical probe rolling over all atom spheres. The probe contact surface traces out the SES. It is the most widely used molecular surface since it provides a meaningful representation of the molecule with respect to the substrate. However, the computation of the SES is involved and even recent, parallel methods [LBPH10, KGE11] can only render the SES for dynamic data sets of less than 100 k atoms interactively on current hardware.

Another molecular surface definition was introduced by Blinn [Bli82] and is known as Metaballs. This is an implicit surface defined as all points $p \in \mathbb{R}^3$ which satisfy a certain equation $F(p) = 0$. Each particle i is represented by a density function $D_i(p)$ that usually degrades with the distance to the atom center a_i. The density value of all particles is added up for each point to a global density field $D(p) = \Sigma_i D_i(p)$. The isosurface is defined by a threshold value T as $F(p) = D(p) - T$. If a Gaussian density function with

DOI: 10.2312/PE/EuroVisShort/EuroVisShort2012/067-071



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

Figure 1: *(a) Semitransparent molecular surface of a small protein (7,231 atoms) combined with the stick model. (b) Molecular surface of a chromatophore (9.6 M atoms) computed with our method. The parameters are adjusted to create a coarse surface that shows the shape of the molecules but smooths out individual atoms. (c) The chromatophore data set in beads representation (<1 M beads). (d) A poliovirus capsid (0.7 M atoms) computed to show atomic detail.*

suitable parameters is used for D_i, the resulting smooth isosurface can model the electron density surface of a molecule closely [Bli82] (cf. Figure 1a, d). These surfaces can be rendered directly via GPU ray casting [MGE07, KSN08] or an intermediate density map can be computed and used for isosurface extraction. Falk et al. [FKRE10] used a uniform grid to visualize particle-based simulations. They applied GPU shader programs to generate the volume on the GPU and extracted an isosurface using GPU ray marching. Krone et al. [KFR*11] used a similar technique to visualize approximate molecular surfaces. Fraedrich et al. [FAW10] sample particles along a perspective grid and use GPU ray marching for rendering. Dias et al. [DG11] recently presented a CUDA-based approach for the construction of the density map and extraction of the isosurface via MC using CUDA.

Stone et al. [SPF*07] and Rodrigues et al. [RHS*08] have described fast GPU algorithms for computing electrostatic potential maps on uniform lattices, strongly influencing our GPU Gaussian density summation algorithm. Our approach differs primarily in that it computes the spatial acceleration structure entirely on the GPU with irregular atom bin sizes, enabling interactive display rates for large structures.

There are several GPU-accelerated optimizations for Lorensen and Cline's original MC algorithm [LC87]. A CUDA implementation based on Bourke's MC [Bou94] is included in the NVIDIA GPU Computing SDK. Dyken et al.'s [DZTS08] implementation is very similar to NVIDIA's, but achieves a noticable speedup by replacing prefix sums with a HistoPyramid [ZTTS06] traversal.

3. Algorithmic and Implementation Details

Our method is split into two parts: the generation of the volumetric density map from the atom positions and the isosurface extraction using MC. A triangulated surface can be beneficial for rendering and crucial for further computations (cf. Section 4 & 5). The whole algorithm is designed to run in parallel on the GPU. We implemented all steps of our algorithm in CUDA, optimized for NVIDIA's Fermi GPUs.

3.1. Volumetric density map generation

The most costly step in our molecular surface generation is the calculation of a scalar volume containing Gaussian densities summed from each of the particles in the neighborhood of each voxel. Gaussian density distributions can be used to approximate both electron density and solvent accessible surfaces for all-atom molecular models [Bli82, GGP96].

The density map generation algorithm accumulates Gaussian densities on a uniformly-spaced 3-D lattice defined within a bounding box that contains all particles; padding of the volume ensures that the extracted surface is not clipped off. The density map generation algorithm satisfies

$$\rho(\vec{r};\vec{r}_1,\vec{r}_2,\ldots,\vec{r}_N) = \sum_{i=1}^{N} e^{\frac{-|\vec{r}-\vec{r}_i|^2}{2\alpha^2}}, \qquad (1)$$

where the density ρ is evaluated at a position $\vec{r}$ by summing over all N atoms. Each atom i is located at position $\vec{r}_i$ and has an associated weighting factor α which is determined by multiplying its radius with user-defined weighting or scaling factors that customize the visualization to produce a surface with an appropriate user-defined level of detail.

The density map can be computed serially in $\mathcal{O}(n^2)$ by summing up the density contributions of all N atoms into each voxel. This quickly becomes impractical even for molecular complexes of moderate size. Since the magnitude of the Gaussian densities contributed by each atom decays very rapidly with increasing distance, the algorithm can be reduced to linear time complexity by eliminating density contributions from atoms beyond a cutoff radius. By limiting the set of contributing atoms to those within the cutoff radius, the work associated with each grid point becomes roughly constant since the atoms are largely uniformly distributed in space [VBW94]. Performance can be improved further by formulating data-parallel versions of the linear-time approach, taking advantage of the fact that the density ρ at each grid point $\vec{r}$ can be computed independently, and that the partial density contributions to a given grid point $\vec{r}$ from different atoms may also be computed in parallel.

We implemented a parallel *gather* approach for GPUs which involves construction of a uniform spatial acceleration grid with a grid spacing equal to the cutoff distance. All particles are sorted into this acceleration grid. For each grid point $\vec{r}$, the algorithm *gathers* density contributions by looping over the atoms contained in the 3^3 neighboring cells of the acceleration grid. Since each grid point is computed independently, there are no parallel write conflicts and no cache coherency is required among processing units. Our CUDA implementation reuses Y and Z atom distance components among multiple grid points in the same Y row, and Z column, by computing multiple grid points in each thread, where only the X distances are unique to each grid point. A secondary benefit of this *thread coarsening* approach is that many per-thread registers are reused and reads of atom data from global memory are amortized over multiple grid points, thus significantly increasing the arithmetic intensity.

When using per-atom colors for the molecular surface, a 3-D RGB-volume is computed in addition to the density map. The color of a grid cell is the sum of the color contributions of all atoms scaled by the associated density contribution and divided by the isovalue. This results in normalized colors and does not require any further post-processing of the color values prior to use in the isosurface extraction.

3.2. Marching Cubes isosurface extraction

We use the marching cubes (MC) algorithm [LC87] to extract an isosurface from the density map. Our implementation is an optimized and extended version of the MC example from the NVIDIA GPU Computing SDK. In the first step, the number of triangles that will be generated is determined for each cube of 2^3 voxels using a lookup table [Bou94]. A cube is considered *active* if it contains at least one triangle. The active cube flag (0 or 1) and the number of triangle vertices per cube are written to a `uint2` array. This step is called the classification. In the second step, a parallel prefix sum of the classification array is computed to get the total numbers of active cubes and triangle vertices. The third step is the compactification of the active cubes. Based on the prefix sum from the previous step, a lookup table is written. The number of entries in this lookup table equals the number of active cubes. Subsequently, this lookup table is used to address only active cubes. In the fourth and last step, the triangles per cube are generated. Due to the compactification in step two and three, this step is only executed for active cubes. Again, the 2^3 voxel values are read and each cube is classified, but now the triangles are actually generated and the vertices are written to the output array. To speed up the computation, we split the triangle vertex calculation and the normal calculation into two separate steps. The first kernel only writes the vertices while second kernel computes the per-vertex normal using central differences. The surface can also be colored according to the 3D color map created during the generation step. Each triangle vertex is located along an edge of the cube. Therefore, we just have to determine the two voxel coordinates at the start and end point of this edge. The vertex color can be calculated by linearly interpolating the two color values at these voxel coordinates.

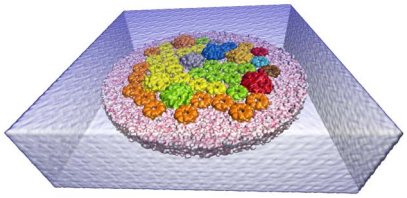


Figure 2: *Membrane patch data set (22.8 M atoms). For this screenshot, the surface of the water box (20 M atoms) was rendered at lower detail than the membrane and proteins to reduce the computation time.*

4. Enhancements and Extensions

In this section we showcase two possible extensions to our fast molecular surface extraction.

Bead simplification. When viewing protein systems of more than one million atoms, it is often desirable to use reduced-detail visualizations to show the overall architecture of the molecular complex instead of the clutter associated with atomic detail. Since biomolecular complexes are usually decomposable into nucleic acids, lipids, or amino acids, bounding spheres can be computed for the atoms they contain, and used as coarse-grained beads for input to the density map generation (cf. Figure 1c). The resulting order-of-magnitude reduction in particles yields a significant performance boost, allowing interactive visualization of much larger molecular complexes.

Molecular surface area. The Gaussian surface descriptors we use can model the solvent accessible surface area (SASA) [Ric77] quite accurately (below 2 % deviation from the area of the SES [GP95]). We implemented an interactive CUDA area calculation using the MC triangulation: after computing all individual triangle areas in parallel, a prefix sum is used to get the total area. In comparison to the rapid stochastic measurement of the SASA of Juba et al. [JV08], our solution is equally fast, but its accuracy is higher. Their area can differ by more than 10 % from the SES [JV08].

5. Results and Discussion

We measured the performance of our method using various data sets from real molecular dynamics simulations. Figure 2 shows our largest test case: a membrane patch with several proteins in water (22.8 M atoms). Our test system was an Intel Core i7 x980 (6×3.3 GHz) with 12 GB RAM and a NVIDIA GeForce GTX 580. In our test application, users can adjust the surface with just three easy to understand parameters: The grid spacing H, the scaling factor R for the atom

Table 1: *Performance measurements (all timings in seconds). Radius scaling R was set to 1.0 and isovalue I to 0.5 for all tests. H denotes the grid spacing. t_{sort} includes the time for uploading the particles to the GPU and sorting them into the acceleration grid, t_d is the computation time for the density map and t_{MC} the MC runtime. The last column shows the overall performance (computation + rendering) in fps.*

#Atoms	H	Map size	t_{sort}	t_d	t_{MC}	fps
147,976	1.0	183×184×184	0.007	0.048	0.009	13.2
754,200 (*Fig. 1d*)	1.0	364×364×364	0.01	0.18	0.05	3.5
955,226	1.0	220×220×220	0.008	0.189	0.012	4.2
2.37 M	2.0	429×394×58	0.03	0.17	0.016	3.9
9.62 M (*Fig. 1b*)	2.0	377×375×355	0.16	0.023	0.06	3.4
22.8 M (*Fig. 2*)	4.0	240×222×72	4.4	0.68	0.01	0.18

radius and the isovalue *I*. To see the molecular surface in atomic detail, we suggest a grid spacing of about 1.0 Å. For larger data sets a smoother, more abstract surface is often more desirable for the visual analysis since it shows the overall shape of the molecule rather than drawing the attention to insignificant single atom movements.

Table 1 shows the performance of our method. The frame rates include the transfer of all particles to the GPU, the density map calculation and the MC extraction for each frame. That is, using our technique it is possible to interactively visualize molecular surfaces for fully dynamic data sets with up to one million atoms in atomic detail. All timings were measured using per-atom coloring. If we compute just the density map without colors, t_d is 30% lower. The computation time is influenced by a combination of the following three factors: the number of particles; the grid resolution, which is a result of the grid spacing H; the number of triangles, which also depends on the spatial arrangement of the particles. These factors mutually influence the timings for the three steps (t_{sort}, t_d, t_{MC}). In most cases, the density map calculation is the most expensive part of our algorithm. However, for very large numbers of particles and small density maps, the sorting time becomes the limiting factor. The MC calculation is very fast and influences the frame rate only marginally. The beads representation reduces the number of particles by at least an order of magnitude for biomolecules. Additionally, the grid spacing can be larger since the beads are also much larger than the original atoms. This results in high frame rates even for very large data set. Figure 1c shows the bead surface for the chromatophore data set (Figure 1b, 9 M atoms). The number of beads is less than 1 M, resulting in more than 10 fps on our test system.

Recently, Krone et al. [KGE11] presented a fast method to analytically compute the SES on the GPU using CUDA for fully dynamic data. For a protein of 59,000 atoms (PDB-ID: 1AON) they measured 6.8 fps on an NVIDIA Quadro 7000. Our algorithm visualizes the molecular surface for this data set on a GeForce GTX 580 (which is comparable to the Quadro) with a grid spacing of 1.0 at 19 fps.

Our method is similar to the work of Dias et al. [DG11]. They also construct a density map and extract a molecular surface via MC using CUDA. For the density grid, they use an atom-based scattering approach rather than a voxel-based gathering approach as we do. For a protein of 7,231 atoms (shown in Figure 1a) they report a computation time of 10 s on a GeForce GTX 280. Our implementation computes the surface of this protein in 0.07 s (same GPU and grid size). Due to the lack of details in [DG11] we are not sure why our method is that much faster, but we assume that their density map calculation is much slower because they have to explicitly avoid write conflicts due to the scattering approach.

Fraedrich et al. [FAW10] sample only the visible particles in the scene into perspective, non-uniform grids in view space. These optimizations result in low computation times for very large, dynamic data sets. They render the isosurface using GPU ray casting. The performance of their method is comparable to our technique. However, since their density grid is view-dependent and non-uniform, the method cannot be used for further analytical processing (cf. Section 4). The triangle-based, undistorted isosurface that we obtain is also beneficial for applications that render the same data from multiple points of view, like 3-D stereoscopic rendering and can be used as input for available offline raytracers.

6. Summary and Conclusion

We have presented a fast method for computing smooth surfaces for large, dynamic particle data sets. Our algorithm is intended for visualization of MD simulation trajectories, however, it is also applicable to all other particle-based data, e.g. SPH simulations. The surface is extracted from an interactively computed density field using a parallelized MC implementation. The density field is computed using Gaussian kernels for each particle. All stages of our algorithm run completely on the GPU, leveraging its tremendous computation power to speed up the computation. The computations are highly parallel and take full advantage of the GPU. This results in interactive frame rates for data sets of more than one million particles. Our method is integrated in the publicly available molecular visualization software VMD [HDS96] as the "QuickSurf" representation, and it has has received very positive feedback from users from the field of biochemistry and biophysics. Users have reported that they like the high performance and ability to display very large structures as compared to other molecular surface computation methods, and they felt that the process of trajectory analysis is enriched by our method.

7. Acknowledgments

This work was partially funded by the German Research Foundation (DFG) as part of the Collaborative Research Center SFB 716, and by the U.S. National Institutes of Health, under grant P41-RR005969.

References

[Bli82] BLINN J. F.: A Generalization of Algebraic Surface Drawing. *ACM Transactions on Graphics 1* (1982), 235–256. 1, 2

[Bou94] BOURKE P.: Polygonising a Scalar Field. Online, http://www.paulbourke.net/geometry/polygonise/, 1994. (last accessed 01.04.2012). 2, 3

[Con83] CONNOLLY M. L.: Analytical molecular surface calculation. *Journal of Applied Crystallography 16* (1983), 548–558. 1

[DG11] DIAS S., GOMES A. J. P.: Graphics Processing Unit-based Triangulations of Blinn Molecular Surfaces. *Concurrency and Computation 23*, 17 (2011), 2280–2291. 2, 4

[DZTS08] DYKEN C., ZIEGLER G., THEOBALT C., SEIDEL H.-P.: High-speed Marching Cubes using HistoPyramids. *Computer Graphics Forum 27*, 8 (2008), 2028–2039. 2

[FAW10] FRAEDRICH R., AUER S., WESTERMANN R.: Efficient High-Quality Volume Rendering of SPH Data. *IEEE Transactions on Visualization and Computer Graphics 16* (2010), 1533–1540. 2, 4

[FKRE10] FALK M., KLANN M., REUSS M., ERTL T.: 3D visualization of concentrations from stochastic agent-based signal transduction simulations. In *Proceedings of IEEE International Symposium on Biomedical Imaging* (2010). 2

[GGP96] GRANT J. A., GALLARDO M. A., PICKUP B. T.: A Fast Method of Molecular Shape Comparison. *Journal of Computational Chemistry 17*, 14 (1996), 1653–1666. 2

[GP95] GRANT J. A., PICKUP B. T.: A Gaussian Description of Molecular Shape. *Journal of Physical Chemistry 99*, 11 (1995), 3503–3510. 3

[HDS96] HUMPHREY W., DALKE A., SCHULTEN K.: VMD – Visual Molecular Dynamics. *Journal of Molecular Graphics 14* (1996), 33–38. http://www.ks.uiuc.edu/Research/vmd/. 4

[JV08] JUBA D., VARSHNEY A.: Parallel Stochastic Measurement of Molecular Surface Area. *Journal of Molecular Graphics and Modelling 27*, 1 (2008), 82–87. 3

[KFR*11] KRONE M., FALK M., REHM S., PLEISS J., ERTL T.: Interactive Exploration of Protein Cavities. *Computer Graphics Forum 30*, 3 (2011), 673–682. 2

[KGE11] KRONE M., GROTTEL S., ERTL T.: Parallel Contour-Buildup Algorithm for the Molecular Surface. In *Proceedings of IEEE Symposium on Biological Data Visualization* (2011). 1, 4

[KSN08] KANAMORI Y., SZEGO Z., NISHITA T.: GPU-based fast ray casting for a large number of metaballs. *Computer Graphics Forum 27*, 3 (2008), 351–360. 2

[LBPH10] LINDOW N., BAUM D., PROHASKA S., HEGE H.-C.: Accelerated visualization of dynamic molecular surfaces. *Computer Graphics Forum 29* (2010), 943–952. 1

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of ACM SIGGRAPH Computer Graphics and Interactive Techniques* (1987), pp. 163–169. 1, 2, 3

[MGE07] MÜLLER C., GROTTEL S., ERTL T.: Image-Space GPU Metaballs for Time-Dependent Particle Data Sets. In *Proceedings of Vision, Modelling and Visualization (VMV '07)* (2007), pp. 31–40. 2

[RHS*08] RODRIGUES C. I., HARDY D. J., STONE J. E., SCHULTEN K., HWU W. W.: GPU acceleration of cutoff pair potentials for molecular modeling applications. In *CF'08: Proceedings of the 2008 conference on Computing Frontiers* (2008), ACM, pp. 273–282. 2

[Ric77] RICHARDS F. M.: Areas, volumes, packing, and protein structure. *Annual Review of Biophysics and Bioengineering 6*, 1 (1977), 151–176. 1, 3

[SPF*07] STONE J. E., PHILLIPS J. C., FREDDOLINO P. L., HARDY D. J., TRABUCO L. G., SCHULTEN K.: Accelerating molecular modeling applications with graphics processors. *Journal of Computational Chemistry 28* (2007), 2618–2640. 2

[TA95] TOTROV M., ABAGYAN R.: The contour-buildup algorithm to calculate the analytical molecular surface. *Journal of Structural Biology 116* (1995), 138–143. 1

[VBW94] VARSHNEY A., BROOKS F. P., WRIGHT W. V.: Linearly scalable computation of smooth molecular surfaces. *IEEE Computer Graphics and Applications 14*, 5 (1994), 19–25. 2

[ZTTS06] ZIEGLER G., TEVS A., THEOBALT C., SEIDEL H.-P.: *GPU Point List Generation Through Histogram Pyramids*. Research report, Max-Planck-Institut für Informatik, 2006. 2