JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING

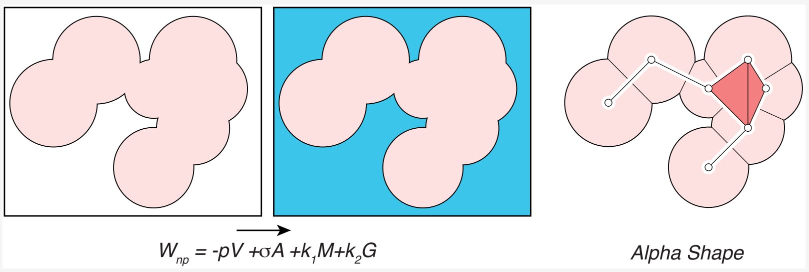
pubs.acs.org/jcim Article

Computing the Volume, Surface Area, Mean, and Gaussian Curvatures of Molecules and Their Derivatives

Patrice Koehl,* Arseniy Akopyan,* and Herbert Edelsbrunner*

Cite This: *J. Chem. Inf. Model.* 2023, 63, 973−985 Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information



ABSTRACT: Geometry is crucial in our efforts to comprehend the structures and dynamics of biomolecules. For example, volume, surface area, and integrated mean and Gaussian curvature of the union of balls representing a molecule are used to quantify its interactions with the water surrounding it in the morphometric implicit solvent models. The Alpha Shape theory provides an accurate and reliable method for computing these geometric measures. In this paper, we derive homogeneous formulas for the expressions of these measures and their derivatives with respect to the atomic coordinates, and we provide algorithms that implement them into a new software package, AlphaMol. The only variables in these formulas are the interatomic distances, making them insensitive to translations and rotations. AlphaMol includes a sequential algorithm and a parallel algorithm. In the parallel version, we partition the atoms of the molecule of interest into 3D rectangular blocks, using a *kd*-tree algorithm. We then apply the sequential algorithm of AlphaMol to each block, augmented by a buffer zone to account for atoms whose ball representations may partially cover the block. The current parallel version of AlphaMol leads to a 20-fold speed-up compared to an independent serial implementation when using 32 processors. For instance, it takes 31 s to compute the geometric measures and derivatives of each atom in a viral capsid with more than 26 million atoms on 32 Intel processors running at 2.7 GHz. The presence of the buffer zones, however, leads to redundant computations, which ultimately limit the impact of using multiple processors. AlphaMol is available as an OpenSource software.

INTRODUCTION

Biological nanomachines, such as proteins and nucleic acids, are essential for all cellular functions due to their abilities to store information, to provide transport to and out of the cell, to catalyze chemical reactions, and to interact and recognize ligands, among other things. Their functions are believed to be intimately related to their shapes (referred to as *structures*), as well as to the dynamics of these shapes. Our current knowledge of the structures and dynamics of large biomolecules remains inadequate. This is because only a few experimental techniques have the ability to gather structural data that are resolved in time and those that can are typically constrained to small length scales and to short time windows. Recently, new algorithms have been proposed for predicting the structures of proteins that have reached significant success. However, predicting and analyzing the dynamics of such structures are tasks that are still limited in scope, both with respect to time scales (usually microseconds to milli-seconds) and length scales (several nanometers for systems of up to hundred thousand atoms).

With the success of AlphaFold[1] and its successor Alpha-Fold2,[2] artificial intelligence has stormed into structural molecular biology in the recent years.[3,4] This is software designed by the company DeepMind to predict the structure of a protein based on its sequence only. AlphaFold has refined numerous deep learning techniques to predict these structures at near experimental-scale resolution, inspiring experimental structural biologists to rethink the way they study the function and evolution of proteins, as well as their impact on diseases.[5−7] AlphaFold's achievement has been made possible by the wealth of information present in the Protein Data Bank,[8] the database of experimentally determined protein structures (approximately 200,000 as of October 2022). In return, AlphaFold allowed for

Received: October 25, 2022
Published: January 13, 2023

the prediction of millions of previously unknown protein structures, all available in an open database.[9] There are, however, limitations to AlphaFold,[7] which only generates single-ranked conformations for a protein. As such, it is currently unable to provide information on ensembles of conformations for a protein, which may arise if this protein is intrinsically disordered. AlphaFold does not solve the protein folding problem as it is inherently static. It does not capture conformational mechanisms such as allostery. The study of these mechanisms still relies on simulations of molecular dynamics.

The standard approach to simulating the dynamics of a biomolecule is to solve numerically the Newton equations associated with all its atoms. The step size in time required for finding accurate solutions to those differential equations is extremely small (in the order of a femtosecond), leading to the need to compute the energy of the molecular system under study a large number of times. One evaluation of the energy is of order $O(N \log N)$, with N being the total number of atoms in the system (including the water molecules in the environment of the biomolecule). For large values of N, say in the millions, such a calculation and more importantly its repeats become computationally prohibitive. While it is possible to design hardware that is specific to such calculations and while many efforts are underway to improve the software that implement them[10−16] (currently allowing for molecular dynamics simulations of systems with up to 100 million atoms[17−21]), parallel efforts are put into developing simplified models in which the number of atoms is reduced to make the calculation more tractable. Of particular interest is to replace the explicit solvent with a potential of mean force that mimics its effect on the molecule. This is akin to deriving and computing a solvation free energy, W_{sol}, for the biomolecule. How to compute the nonpolar component of this solvation free energy is the topic of this paper. It is noteworthy that current versions of AlphaFold focus on the conformation of the protein alone, independent of its environment. Inclusion of solvation free energy to further refine the prediction is likely to be an addition in newer versions of the software, reinforcing the need to derive accurate and robust methods for computing such solvation free energies.

A Morphometric Approach to the Nonpolar Solvation Free Energy. The solvation free energy $W_{sol}(X)$ of a biomolecule with conformation X is set to capture the presence of a cavity within the solvent that enables it to accommodate the biomolecule and the vdW interactions between the water molecules and the atoms at the surface of the biomolecule, as well as the interactions between the charged atoms of the biomolecule in the presence of water. The first two contributions define the nonpolar effect, W_{np}, while the third one captures the polar effect, W_{pol}. These effects are additive, namely, $W_{sol} = W_{np} + W_{pol}$. They can be computed individually with the help of a thermodynamic cycle, as illustrated in Figure 1.

Eisenberg and McLachlan[22] proposed an atomic break down of the computation of the nonpolar part of the solvation free energy of a biomolecule. In their model, each atom is represented with its accessible surface area, ASA,[23] which is then scaled with a surface tensor factor referred to as atomic solvation parameter, or ASP, such that

$$W_{np} = \sum_{i=1}^{N} ASP_i ASA_i \tag{1}$$

The ASP is a signed number, positive for nonpolar atoms (large accessible surface areas are then penalized for such nonpolar

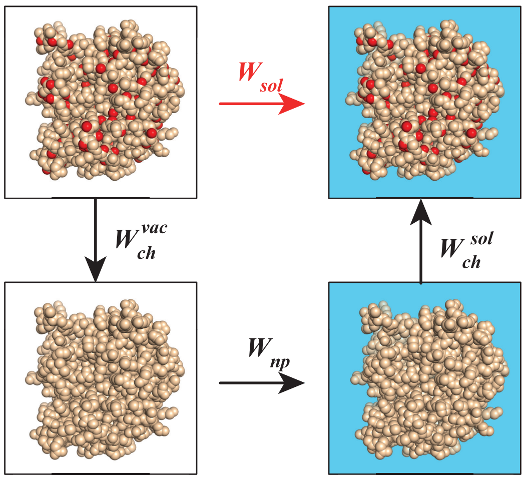


Figure 1. Computing the solvation free energy of a biomolecule. The solvation free energy, W_{sol}, is defined as a mean force potential that quantifies the energy that is required to solvate a molecule. It consists of two parts: (i) a polar contribution, W_{pol}, which accounts for the effects of the solvent on the charges of the biomolecule and (ii) a nonpolar contribution, W_{np}, which accounts for the formation of a hole within the solvent so that it can fit the biomolecule as well as for the vdW interactions between the biomolecule and the solvent (at the surface of the biomolecule). These two parts are best described with a thermodynamics cycle. First, the charges on the biomolecule (symbolized by the red balls) are neutralized in vacuo. The corresponding free energy cost is referred to as W_{ch}^{vac}. Second, the corresponding neutral molecule is solvated, with a cost W_{np}. Finally, the charges are added back to the molecule, now in solution, with an energetic cost W_{ch}^{sol}. The solvation free energy is the sum of those three contributions, namely, $W_{sol} = (W_{ch}^{vac} + W_{ch}^{sol}) + W_{np} = W_{pol} + W_{np}$ (red arrow).

atoms) and negative for polar atoms (i.e., favoring large accessible surface areas for them). This surface-only model, referred to as SA, is supported indirectly with the observation that the Gibbs free energy for transferring small compounds from nonaqueous liquids to water is linearly related to their accessible surface area. SA has become the preferred approach for studying the dynamics of a biomolecule with an implicit solvent, in conjunction with Poisson−Boltzmann (PBSA) or generalized Born (GBSA) .[24] It is interesting, however, to think back on the fact that W_{np} accounts for two effects, namely, the hole formation in the solvent and the vdW interactions between the atoms of the biomolecule and the solvent molecules. While the latter occurs near the boundary between the biomolecule and the solvent and is therefore proportional to the accessible surface area of the molecule, the former is proportional to the volume of that molecule. This apparent contradiction between a surface area model only and the fact that W_{np} includes a volume-based contribution currently fuels a debate on the geometric nature of W_{np}. Lum, Chandler, and Weeks for example have shown that W_{np} scales with the volume of the solute for small solutes, is proportional to the surface area for very large solutes, and should consider both geometric measures in between.[25] This idea that W_{np} for a molecular system depends on surface area and volume is derived from scaled particle theory.[26−28] It was shown to be a better representation of solvent effects than a surface-based solvation free energy.[29,30] However, even a combined surface area and volume representation for W_{np} seems to be deficient to represent length-scale dependence of this energy.[31] More recently, W_{np} has been expressed as a linear

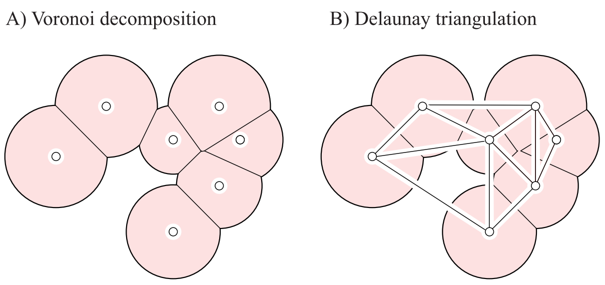


Figure 2. Voronoi decomposition and dual complex of a union of disks. (A) Given a finite set of disks, the Voronoi diagram corresponds to a decomposition of the whole plane into regions, one for each disk, such that any point that belongs to the region corresponding to disk D_i is closer to that disk than to any other disk, with the distance to D_i being the power distance (see text for details). In the graphics, we have restricted the Voronoi diagram to the region covered by the disks. This defines a decomposition of the union of disks into convex regions. (B) The Delaunay triangulation is the dual of the Voronoi diagram that is constructed by defining edges between disk centers of neighboring Voronoi regions. (C) The dual complex is a subset of the Delaunay triangulation, limited to the edges and triangles (dark red) whose corresponding Voronoi regions fully intersect within the union of disks.

combination of the four morphometric measures of the molecule:[32]

$$W_{np} = -pV + \sigma A + k_1 M + k_2 G \qquad (2)$$

In this equation, V, A, M, and G are the volume, surface area, mean curvature, and Gaussian curvature of the molecular system, while p, σ, and k_1 and k_2 are the pressure, surface tension, and bending rigidity parameters. This is under the assumption that the solvation free energy satisfies (i) *motion invariance*, namely, independence with respect to the location and orientation of the molecular system in space, (ii) *continuity*, basically that thermodynamics can be expressed in terms of geometry (a condition that is only violated for system whose size is similar to the size of the solvent), and (iii) *additivity*, i.e., that the energy of the union of two domains is the sum of the energy of the single domains subtracted by the energy of the intersection. This model has proved useful to study the solvation of proteins and their ligands.[33−35]

All the solvation models presented above consistently associate the nonpolar contribution to the solvation free energy of a molecule to the geometry of this molecule, the *continuity assumption*. In what follows, we discuss how different measures of this geometry, more specifically volume, surface area, mean curvature, and Gaussian curvature, and their positional derivatives can be computed efficiently, even for very large molecular systems.

Computing Geometric Measures of Biomolecules. A union of balls, with each ball representing an atom, is a typical geometric representation of a molecule. Lee and Richards[36] developed the first approach to computing the accessible surface area of a protein represented by such a union by first cutting it with a set of parallel planes. Shrake and Rupley[37] proposed instead a Monte Carlo numerical integration method to compute regions of the surfaces of atoms that are accessible. Many efficient implementations of this method have been proposed, including the use of look-up tables,[38] as well as of algorithms that make use of the parallel architecture of computer central processing units (CPU) .[39,40] All those methods have been expanded to compute also the volume of a union of balls.[41−44]

Numerical integration techniques, while practical, are not accurate, and more importantly do not easily provide derivatives for the quantities that they compute. This is true for the computations of surface area of a union of balls described above. Analytical alternatives have been proposed, although computing geometric measures of overlapping balls is not an easy task. Approximations have been proposed that treat overlapping balls using a probabilistic model[45−47] or by fully ignoring them.[48,49] Such approximations are ideally suited for the parallel architecture of graphics processing units (GPU) .[50] They remain approximations, however, that are prone to singularities introduced by numerical errors or by discontinuities in the derivatives.[51,52] Better analytical methods refine the geometric representation of the molecule, considered as a union of pieces of balls.[53−58]

This Work. To model the nonpolar contribution to the solvation energies of a biomolecular system, in support of the morphometric model described by eq 2, we consider weighted versions of four measures of the geometry of the union of balls, namely, the volume, surface area, integrated mean curvature, and integrated Gaussian curvature, as well as their derivatives with respect to the positions of the ball centers. This paper presents an extension of a large body of work in which these measures and their derivatives have been characterized before in the context of the Voronoi decomposition of a space filling diagram.[59−64] Its contributions are two-fold. First, we present comprehensive and consistent sets of equations for the expression of the four measures and their derivatives in intrinsic geometry, i.e., as functions of the distances between the centers of the atoms only. Second, we establish parallel algorithms for computing the measures and their derivatives, targeting very large molecular systems with millions of atoms. We use viral capsids to illustrate the performances of these algorithms. Information about the structures of such capsids was recovered from the Protein Data Bank[8] or from the VIPER database.[65]

Outline. The next section, Measuring Union of Balls, provides a brief description of the Alpha Shape theory and its application to measuring a union of balls. It includes subsections that provide expressions for the weighted volume, surface area, mean curvature, and Gaussian curvature and their derivatives, all expressed in terms of the distances between the centers of the balls. The explicit constituents for those expressions are provided in the Supporting Information, parts A−D. The following section, Algorithm and Implementation, describes our parallel implementation of this theory. It includes testing on a set of large virus capsids. The last section concludes the paper.

■ MEASURING UNION OF BALLS

Given a collection of N 3-dimensional bodies, P_i, any geometric measure of the union of the P_i can be derived from the principle of *inclusion−exclusion*. That is, a measure of the union, $\cup_i P_i$, is expressed as an alternating sum of the measures of the intersections of the P_i. To make the inclusion−exclusion formula amenable to computation, however, two issues need to be solved. First, we need to reduce significantly the number of terms it includes, as a brute force application of the formula leads to an algorithm with exponential running time, as the total number of potential intersections of P_i terms is $2^N - 1$. Second, we need analytical formulas for computing the measures of the those intersections of bodies. The next three subsections provide solutions to these two issues when the bodies are 3D balls.

Background on Voronoi Decompositions and Dual Complexes. Let us consider a finite set of closed balls, B_i, with centers z_i and radii r_i, and let S_i be the sphere that is the boundary of B_i. We define the *power distance* between a point x and a ball B_i as $\pi_i(x) = \|x - z_i\|^2 - r_i^2$. The *Voronoi region* of B_i includes all points x that are at least as close to B_i as to any other ball: $V_i = \{x \in \mathbb{R}^3 | \pi_i(x) \leq \pi_j(x)\}$. It is a convex polyhedron obtained as the common intersection of finitely many closed half-spaces, one per ball $B_j \neq B_i$. The collection of all Voronoi regions, V_i, is the *Voronoi diagram* of the balls. Note that their union covers the entire space. The intersection of the Voronoi diagram with the union of balls B_i decomposes this union into convex regions, as shown in Figure 2A.

The *Delaunay triangulation* is the dual of the Voronoi diagram. It is obtained by defining an edge between the centers of the balls B_i and B_j if and only if the two corresponding Voronoi regions share a common face. In addition, we generate a triangle connecting z_i, z_j, z_k if V_i, V_j, V_k intersect in a common line segment, and we generate a tetrahedron connecting z_i, z_j, z_k, z_l if V_i, V_j, V_k, V_l meet at a common point. A 2D version of the Delaunay triangulation is illustrated in Figure 2B. Assuming general positions of the balls, those are the only cases to be considered. We call this the *generic case*. This generic case is rare in practical implementations because of finite precision for the computer representations of the coordinates and radii. It is, however, possible to simulate a perturbation of the union of balls that always restores the generic case.[66]

Next, we limit the construction of Delaunay triangulation to within the union of balls. In other words, we draw a dual edge between the two vertices, z_i and z_j, only if $B_i \cap V_i$ and $B_j \cap V_j$ share a common face and similarly for triangles and tetrahedra. The result is a subcomplex of the Delaunay triangulation, which is referred to as the *dual complex* of the set of balls (see Figure 2C). Our objective is to use the dual complex $X = \cup_i B_i$, corresponding to a biomolecule to compute its nonpolar solvation free energy, which is expressed in a general form as

$$\begin{aligned} W_{np}(X) &= V_w(X) + A_w(X) + M_w(X) + G_w(X) \\ &= \sum_i a_i V_i + \sum_i b_i A_i + \sum_i c_i M_i + \sum_i d_i G_i \end{aligned} \tag{3}$$

Here, V_w, A_w, M_w, and G_w are the total weighted volume, weighted surface area, integrated weighted mean curvature, and integrated weighted Gaussian curvature of the union of balls. The a_i, b_i, c_i, and d_i are weights, while the V_i, A_i, M_i, and G_i are the contributions of ball i to the total corresponding measure of the union of balls. The sum extends over all atoms of this union. The Voronoi decomposition of the union of balls described above allows us to compute the different terms in these equation based on intersections of up to four balls only.

Area and Volume Formulas. Write K for the dual complex. A simplex, s, in K can be understood abstractly as a collection of balls: one ball if it is a vertex, two if it is an edge, three if it is a triangle, and four if it is a tetrahedron. As proved in ref 59, the inclusion−exclusion formula that corresponds to the dual complex gives the correct volume and surface area of a union of balls. Let s_i be the vertex corresponding to the ball B_i, s_{ij} the edges of balls B_i and B_j, s_{ijk} the triangle of balls B_i, B_j, B_k, and finally, s_{ijkl} the tetrahedron of four balls, B_i, B_j, B_k, and B_l. Then:

Proposition 1: (Area)

$$A_i = \mathcal{A}_i - \sum_{j|s_{ij}\in K} \mathcal{A}_{i;j} + \sum_{(j,k)|s_{ijk}\in K} \mathcal{A}_{i;jk} - \sum_{(j,k,l)|s_{ijkl}\in K} \mathcal{A}_{i;jkl} \tag{4}$$

$$= \gamma_i \mathcal{A}_i - \sum_{j|s_{ij}\in K} \gamma_{ij} \mathcal{A}_{i;j} + \sum_{(j,k)|s_{ijk}\in K} \gamma_{ijk} \mathcal{A}_{i;jk} \tag{5}$$

Proposition 2: (Volume)

$$V_i = \mathcal{V}_i - \sum_{j|s_{ij}\in K} \mathcal{V}_{i;j} + \sum_{(j,k)|s_{ijk}\in K} \mathcal{V}_{i;jk} - \sum_{(j,k,l)|s_{ijkl}\in K} \mathcal{V}_{i;jkl} \tag{6}$$

$$= \gamma_i \mathcal{V}_i - \sum_{j|s_{ij}\in K} \gamma_{ij} \mathcal{V}_{i;j} + \sum_{(j,k)|s_{ijk}\in K} \gamma_{ijk} \mathcal{V}_{i;jk} - \sum_{(j,k,l)|s_{ijkl}\in K} \mathrm{vol}\, F_{i;jkl} \tag{7}$$

Here, $\mathcal{V}_i$ is the volume of the ball B_i, $\mathcal{V}_{i;j}$ is the contribution of B_i to the volume of the intersection of the balls B_i and B_j, etc. Similar definitions are used for the surface areas $\mathcal{A}$.

Note that even though the eqs 4 and 6 for the surface area and volume are minimal as they only consider up to four levels in the inclusion−exclusion formula, it is possible to find even shorter expressions if noninteger coefficients are considered. Those expressions correspond to the short inclusion−exclusion method; it is described in detail in ref 59. In this method, the areas and volumes are expressed as the sums of the contributions of intersections of at most three balls, with angular coefficients γ_i, γ_{ij}, and γ_{ijk}, with the exception of the term vol $F_{i;jkl}$ (a fraction of the Voronoi region of B_i; see Supporting Information, part A). These coefficients γ are the normalized exposed angles of the simplices;[60] they integrate the contributions of the tetrahedra of the dual complex. For vertices and edges, these angles can be expressed as fractions of solid and dihedral angles inside tetrahedra. If we define $\Omega_{i;jkl}$ as the solid angle at vertex z_i and $\phi_{ij;kl}$ as the dihedral angle at the edge $z_i z_j$ in the tetrahedron defined by z_i, z_j, z_k, z_l, the coefficients are

$$\gamma_i = 1 - \sum_{j,k,l|\tau_{ijkl}\in K} \frac{\Omega_{i;jkl}}{4\pi} \tag{8}$$

$$\gamma_{ij} = 1 - \sum_{k,l|\tau_{ijkl}\in K} \frac{\phi_{ij;kl}}{2\pi} \tag{9}$$

$$\gamma_{ijk} = 1 - \sum_{l|\tau_{ijkl}\in K} \frac{1}{2} \tag{10}$$

Expressions for the derivatives with respect to the Cartesian coordinates of the centers of the balls are available for the surface area[61] and for the volume.[60] Alternate expressions are available for the same derivatives with respect to the distances between the center of these balls.[67] Note that these distances define internal coordinates for the system, which are invariant under rigid body transformations (rotations and translations). We recall those derivatives here:

Proposition 3: (Area derivative)

$$\frac{\partial A_i}{\partial r_{ab}} = \frac{\partial \gamma_i}{\partial r_{ab}}\mathcal{A}_i - \sum_{j|s_{ij}\in K}\left(\frac{\partial \gamma_{ij}}{\partial r_{ab}}\mathcal{A}_{i;j} + \gamma_{ij}\frac{\partial \mathcal{A}_{i;j}}{\partial r_{ab}}\right) + \sum_{(j,k)|s_{ijk}\in K}\gamma_{ijk}\frac{\partial \mathcal{A}_{i;jk}}{\partial r_{ab}} \tag{11}$$

Proposition 4: (Volume derivative)

$$\frac{\partial V_i}{\partial r_{ab}} = \frac{\partial \gamma_i}{\partial r_{ab}}\mathcal{V}_i - \sum_{j|s_{ij}\in K}\left(\frac{\partial \gamma_{ij}}{\partial r_{ab}}\mathcal{V}_{i;j} + \gamma_{ij}\frac{\partial \mathcal{V}_{i;j}}{\partial r_{ab}}\right) + \sum_{(j,k)|s_{ijk}\in K}\gamma_{ijk}\frac{\partial \mathcal{V}_{i;jk}}{\partial r_{ab}} - \sum_{(j,k,l)|s_{ijkl}\in K}\frac{\partial \mathrm{vol}\, F_{i;jkl}}{\partial r_{ab}} \tag{12}$$

The derivatives of the surface area and volume are expressed in eqs 11 and 12, respectively. They are derived from the corresponding simplified, angle-weighted inclusion−exclusion eqs 5 and 7, respectively. Note that there are no derivatives of $\mathcal{A}_i$ and $\mathcal{V}_i$, which are constant, and that there are no terms involving the derivatives of γ_{ijk}: these derivatives are piecewise zero because the γ_{ijk} are piecewise constant. Their values change at nongeneric states, where their derivatives are not defined.[60,61] Finally, we note that the derivatives of A_i and V_i with respect to the distance r_{ab} between the centers z_a and z_b of the two balls B_a and B_b are nonzero if and only if i, a, and b belong to a simplex of K.

Proofs of eqs 4, 5, 6, 7, 11, and 12 and additional formulas are provided in refs 59−61 and 67. We summarize them in Supporting Information, part A, for sake of completeness.

Mean Curvature formulas. Akopyan and Edelsbrunner recently derived theorems for computing the integrated mean curvature over the surface of a union of balls using the dual complex.[63] They distinguish between two terms: the contribution of the spherical patches and the contribution of the accessible circular arcs at the intersections of two spheres. Along these circular arcs, the mean curvature is partitioned equally between the two spheres involved. This leads to the following formula for the mean curvature and its derivatives in terms of the edge lengths in the dual complex:

Proposition 5: (Mean curvature)

$$M_i = \frac{A_i}{r_i} - \frac{\pi}{2}\sum_{j|s_{ij}\in K}\sigma_{ij}\alpha_{ij}r_{ij} \tag{13}$$

Proposition 6: (Mean curvature derivative)

$$\frac{\partial M_i}{\partial r_{ab}} = \frac{1}{r_i}\frac{\partial A_i}{\partial r_{ab}} - \frac{\pi}{2}\sum_{j|s_{ij}\in K}\left(\frac{\partial \sigma_{ij}}{\partial r_{ab}}\alpha_{ij}r_{ij} + \sigma_{ij}\frac{\partial \alpha_{ij}}{\partial r_{ab}}r_{ij} + \sigma_{ij}\alpha_{ij}\frac{\partial r_{ij}}{\partial r_{ab}}\right) \tag{14}$$

In addition to the contribution A_i of the sphere S_i to the total surface area of the union of balls, these equations involve three new terms, r_{ij}, α_{ij}, σ_{ij}, all associated with two balls B_i and B_j that form a simplex of the dual complex. The spheres S_i and S_j that bound those balls intersect at a circle S_{ij}; r_{ij} is the radius of this circle, and α_{ij} is the angle between the unit normals of the spheres at any point of S_{ij}; see Figure 3. Finally, σ_{ij} is the fraction of the length of S_{ij} that is at the boundary of the union of balls

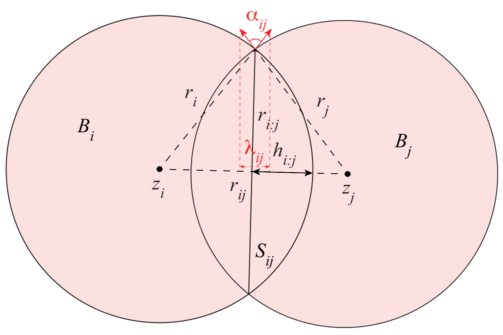


Figure 3. Intersection of two disks.

(i.e., not covered by other balls). Akopyan and Edelsbrunner[63] established formulas for these three terms as functions of the Cartesian coordinates of the centers of the balls in the union. In Supporting Information, part B, we revisit these formulas using internal coordinates (namely, the distances between the centers of the balls) instead.

Gaussian Curvature Formulas. In parallel to the mean curvature formula, Akopyan and Edelsbrunner established a formula for the Gaussian curvature that distinguishes between three terms: the contribution of the spherical patches, the contribution of the circular arcs at the intersections of two spheres, and the contribution of the accessible corners at the intersection of three spheres.[64] This leads to the following formula for the Gaussian curvature and its derivatives in terms of edge lengths in the dual complex:

Proposition 7: (Gaussian curvature)

$$G_i = \frac{A_i}{r_i^2} - \frac{\pi}{2}\sum_{j|s_{ij}\in K}\sigma_{ij}\lambda_{ij} + \frac{1}{3}\sum_{j,k|s_{ijk}\in K}\gamma_{ijk}\sigma_{i;jk} \tag{15}$$

Proposition 8: (Gaussian curvature derivative)

$$\frac{\partial G_i}{\partial r_{ab}} = \frac{1}{r_i^2}\frac{\partial A_i}{\partial r_{ab}} - \frac{\pi}{2}\sum_{j|s_{ij}\in K}\left(\frac{\partial \sigma_{ij}}{\partial r_{ab}}\lambda_{ij} + \sigma_{ij}\frac{\partial \lambda_{ij}}{\partial r_{ab}}\right) + \frac{1}{3}\sum_{j,k|s_{ijk}\in K}\gamma_{ijk}\frac{\partial \sigma_{i;jk}}{\partial r_{ab}} \tag{16}$$

All variables have been defined above, except for λ_{ij} and $\sigma_{i;jk}$, associated with two and three spheres, respectively. Two spheres, S_i and S_j, with centers z_i and z_j that form an edge in K intersect at a circle S_{ij}. λ_{ij} is the combined length of the unit normals of the spheres at any point of S_{ij} after projection on the line passing through z_i and z_j; see Figure 3. Three spheres, S_i, S_j, and S_k, that form a triangle in K intersect in two points, P_{ijk} and P_{ikj}. σ_{ijk} is the fraction assigned to i of the solid angle spanned by the unit normals of S_i, S_j, and S_k at one of those points. Akopyan and Edelsbrunner[64] established formulas for those two terms. In Supporting Information, part C, we revisit those formulas using internal coordinates.

The Nonpolar Solvation Free Energy W_{np}. Recall that the nonpolar contribution to the solvation free energy of a union of balls, X, is

$$W_{np}(X) = \sum_i a_iV_i + \sum_i b_iA_i + \sum_i c_iM_i + \sum_i d_iG_i \tag{17}$$

in which V_i, A_i, M_i, and G_i are the contributions of ball i to the total volume, surface area, integrated mean curvature, integrated Gaussian curvature of X, respectively, and a_i, b_i, c_i, and d_i are the coefficients corresponding to pressure, surface tension, and bending rigidities. In the previous subsections, we have established formulas for those contributions, as well as for their derivatives with respect to internal coordinates. For any given pair of balls, B_a and B_b, that belong to the dual complex, K, of X, we have

$$\frac{\partial W_{np}(X)}{\partial r_{ab}} = \sum_i a_i \frac{\partial V_i}{\partial r_{ab}} + \sum_i b_i \frac{\partial A_i}{\partial r_{ab}} + \sum_i c_i \frac{\partial M_i}{\partial r_{ab}} + \sum_i d_i \frac{\partial G_i}{\partial r_{ab}} \tag{18}$$

in which $\frac{\partial V_i}{\partial r_{ab}}, \frac{\partial A_i}{\partial r_{ab}}, \frac{\partial M_i}{\partial r_{ab}}, \frac{\partial G_i}{\partial r_{ab}}$ are given by eqs 12, 11, 14, and 16, respectively, with all details given in the Supporting Information. Once the derivatives in terms of internal coordinates are available, derivatives with respect to Cartesian coordinates are easily computed using the chain rule:

Proposition 9: (Derivative ofW_{np}) The gradient $\mathbf{a}$ of the nonpolar solvation free energy is

$$\begin{bmatrix} \mathbf{a}_{3i+1} \\ \mathbf{a}_{3i+2} \\ \mathbf{a}_{3i+3} \end{bmatrix} = \sum_{j|\tau_{ij} \in K} \frac{\partial W_{np}}{\partial r_{ij}} u_{ij} \tag{19}$$

in which $u_{ij} = (z_i - z_j)/r_{ij}$ is the unit vector along the edge from z_j to z_i.

■ ALGORITHM AND IMPLEMENTATION

Our software for computing geometric measures of biomolecules has gone through successive revisions. AlphaVol was our original software package, which implemented the Alpha Shape theory for the volumes and surface areas of biomolecules;[62] its origins can be traced to the Alpha Shape package.[68] AlphaVol was partially redesigned into a new package, UnionBall, with modification needed to deal with large molecular systems.[67] We have now completely redesigned UnionBall into a new package, AlphaMol, written in C++. AlphaMol implements all four intrinsic volumes, as well as their derivatives with respect to atomic coordinates. Each of these measures is possibly weighted, i.e., the contribution of each atom is weighted by a constant provided as input to the software, with a different constant for each of the intrinsic volumes. AlphaMol takes as input a set of balls in $\mathbb{R}^3$, each specified by the coordinates of its center and the radius, as well as by its four weights. In the case of biomolecules, the coordinates of the center of the balls are extracted from the corresponding PDB file, while the radii are defined according to the chemical nature of the atoms, using one of several standard sets of radii (in the following we use the OPLS force field[69]). These radii may be enlarged by the radius of a water probe (usually 1.4 Å), should the measures correspond to the accessible surface of the molecule. The algorithm includes three steps:

Step 1. Build the Delaunay triangulation.

Step 2. Extract the dual complex from the Delaunay triangulation.

Step 3. Compute the geometric measures of the union of balls using the dual complex.

Just like AlphaVol and UnionBall, AlphaMol uses standard algorithms from computational geometry for the first two tasks.[70,71] We have designed our own algorithm for measuring the union (step 3) .[60,61,63,64] We have made modifications to these algorithms compared to AlphaVol and UnionBall, as our interests are mostly measuring biomolecules, with a focus on scalability, namely, the ability to deal with very large biomolecules. We describe those modifications in the following subsections, for the sequential and parallel version of AlphaMol, respectively.

A Sequential Algorithm for Measuring Biomolecules. We implemented the randomized incremental algorithm from Edelsbrunner and Shah[70] to construct the Delaunay triangulation of a union of balls. In this algorithm, the triangulation is built incrementally, by adding one ball at a time. The input balls are preprocessed with a random permutation. Four dummy balls whose centers lie at infinity are added so that all input balls are contained in the tetrahedron they define. Let DT_i be the Delaunay triangulation at step i of the construction (DT_i contains the four balls at infinity as well as B_1, B_2, ..., B_i). The algorithm proceeds by iterating three steps:

```
for $i = 1$ to $N$ do
(1) Identify the tetrahedron $t \in DT_{i-1}$ that contains $z_i$.
(2) Add $z_i$ as a vertex and decompose $t$ into four tetrahedra.
(3) Flip locally all non-Delaunay triangles attached to $z_i$.
endfor.
```

The randomization guarantees a theoretical expected running time of $O(N \log\ N)$ with an additional linear term in the number of simplices in the Delaunay triangulation.[70] In $\mathbb{R}^3$, the number of simplices can be as large as a constant times N^2. However, for well-packed data—which is typical for biomolecules—this number is at most a constant times N, leading to an expected running time of $O(N \log\ N)$.

In practice, a different behavior is observed for very large molecules (such as macromolecular assemblies with millions of atoms, for example virus capsids). This is unfortunately a known problem. Virtual memory operating systems cache recently used data in memory, under the assumption that they are more likely to be used again soon. If the new ball to be inserted in step 1 is not included in one of the recent tetrahedra, the cache will not be useful. This scenario is likely if the order in which the balls are inserted is random. A possible solution is to create 3D locality, by ordering the balls first such that a ball at position i is mostly local with respect to the previous balls in the ordering. Interestingly, the order in which atoms are stored in a PDB file is inherently local.[67] UnionBall implemented this idea. Instead of randomizing the balls, as in AlphaVol, it keeps the ordering provided by the input PDB file. The effect was significant: UnionBall is substantially faster than AlphaVol, especially for large molecular systems.[67] With this simple trick, however, there are no guarantee for the expected running time.

Amenta, Choi, and Rote[72] developed a scheme for ordering points before computing the Delaunay triangulation that maintains enough randomness so that the theoretical complexity of the algorithm is conserved. Their Biased Randomized Insertion Order (BRIO) method was shown to significantly improve the running time of Delaunay triangulation for large number of points.[72] Later, Liu and Snoeyink[73] proposed a different method for ordering the points based on the Hilbert curve. They showed that reordering points using such a Hilbert curve significantly sped up the point location in step 1 of the Delaunay triangulation algorithm.[73]

We implemented our own version of BRIO in AlphaMol as an option. BRIO proceeds by organizing the points randomly into *rounds*, using a logarithmic scheme.[72] Within each round, points

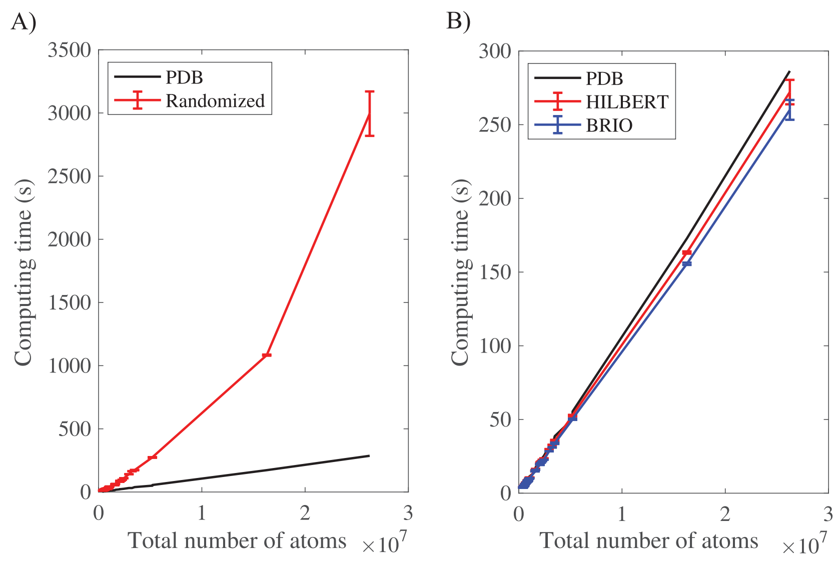


Figure 4. (A) The running times of AlphaMol for measuring biomolecules (the 68 virus capsids in our database; see Supporting Information, part E), when the atoms are inserted randomly (red line) or based on the order provided by the PDB file (black line). (B) Comparing the running times of AlphaMol when atoms are inserted based on the PDB order (black line), or randomly inserted, followed by ordering based on the Hilbert curve (red line) or followed by BRIO-Hilbert ordering (blue curve). In both (A) and (B), we show the mean and standard deviation over five random trials. Computations were performed on a single core on an iMac computer with an 3.8 GHz 8-core 10th-generation Intel Core i7.

can be inserted into any order, allowing for locality. In the original BRIO, within a round, the points were ordered using a *kd*-tree; we used the Hilbert curve instead. We added another option to AlphaMol, in which the ordering follows the Hilbert curve directly (this is equivalent to BRIO with a single round). We compared our versions of BRIO and of the Hilbert curve ordering with the predefined ordering imposed by the PDB file (as implemented in UnionBall) and with randomized ordering (as implemented in AlphaVol) on a set of 68 virus capsids (see Supporting Information, part E, for a full list). These virus capsids vary in size from 400,000 atoms to 26,000,000 atoms, representing a broad range of sizes for large biomolecular systems. Results of the comparisons of the different ordering schemes are illustrated in Figure 4.

As illustrated in Figure 4A, not randomizing the order in which points are inserted resulted in a significant improvement in performance. This was already observed with UnionBall.[67] Removing the randomization leads to an observed linear dependence of the running time on the number of weighted spheres considered. Randomization followed by ordering based on a spacing-filling curve, or on our modified BRIO method, leads to further speed-up, albeit small; see Figure 4B. Note that the differences in running time between the Hilbert curve ordering and BRIO are not significant. In the following, we use the Hilbert curve ordering when necessary. We note that both BRIO and the Hilbert curve method require preprocessing of the data that comes with its own computational cost. This cost, however, is minimal, representing 1.7% of the total computing time, on average; see Figure 5. The bulk of the calculation comes from computing the Delaunay triangulation (52.4% on average), followed by the extraction of the dual complex (27.4%), and the computations of the intrinsic volumes and their derivatives (18.5%).

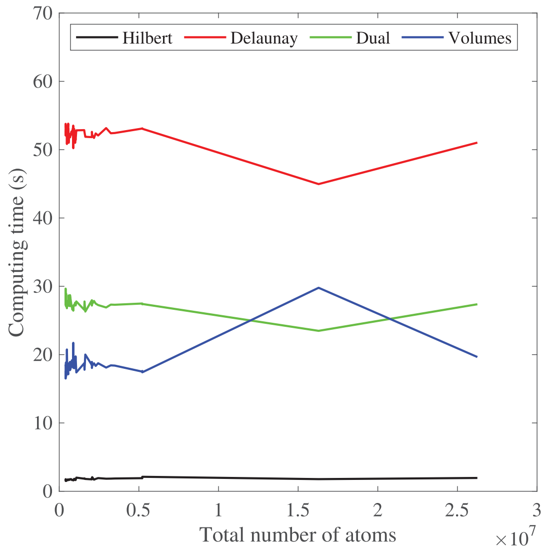


Figure 5. Fraction of running times of the different components of AlphaMol for measuring the 68 virus capsids in our database; (see Supporting Information, part E), in percentage: black, Hilbert curve ordering of the atoms (average 1.7%); red, weighted Delaunay triangulation of the ball centers (average 52.4%); green, filtering the Delaunay triangulation to generate the dual complex (average 27.4%); and blue, computing the intrinsic volumes based on the dual complex (average 18.5%).

A Parallel Algorithm for Measuring Biomolecules. As described above, measuring a biomolecule represented by a union of balls involves three steps: computing the Delaunay triangulation of the centers of the balls weighted by their radii, filtering the simplices of the Delaunay triangulation to build the dual complex, and building the inclusion−exclusion formulas for the intrinsic volumes of the union of balls. Of these three steps, the second and the third can be easily parallelized, as they basically involve iterating over simplices. Unfortunately, parallelizing the computation of the Delaunay triangulation is a difficult task that remains a hot topic in research.[74−80] Considering that computing the Delaunay triangulation is more than 50% of the total computing cost of AlphaMol (see Figure 5), this is a concern.

Many parallel Delaunay triangulation algorithms have been proposed. Most focus on partitioning the domain that contains the points so that each partition can be triangulated separately and in parallel with the others.[74−80] The bottleneck of this approach, however, is the merging of the triangulations from the different partitions to generate the complete triangulation.[76] The tetrahedra at the borders of a partition have to be connected to tetrahedra in adjacent partitions, often leading to local retriangulations. This merging step is sometimes referred to as *stitching* and is known to be difficult. We note, however, that the goal of AlphaMol is to compute the contributions of all atoms to the intrinsic volumes of the biomolecule of interest and not generate the overall Delaunay triangulation. We propose a different parallel strategy for computing these contributions, that still uses the concept of partitioning the whole domain, but with a focus on the volumes, and not the Delaunay triangulation. It is explained in Figure 6 and illustrated for a virus capsid in Figure 7.

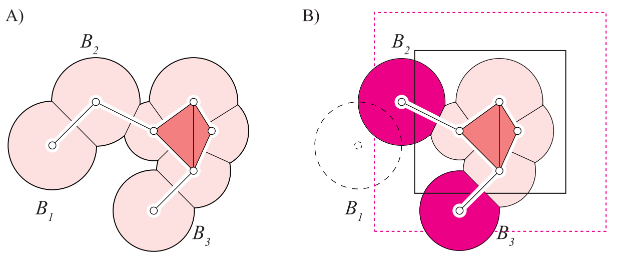


Figure 6. Splitting the computation of the intrinsic volumes of a union of balls. Let us consider the union of balls represented in panel A (this is the same union as in Figure 2, in which we show the dual complex overlaid with the restriction of the Voronoi diagram to within the portion of the plane covered by the balls). In panel B, we limit this union to those balls whose centers belong to the rectangular block shown with solid sides; those balls are colored rose. We expand this block with a buffer zone, delimited by the dashed, magenta sides. Balls B_2 and B_3, whose centers are within this buffer zone, interact with balls from the rectangular block. All other balls (here B_1) are ignored. Applying AlphaMol to the balls in the block and in the buffer zone leads to the correct computation of the intrinsic volumes for the balls in the block. The procedure is repeated in parallel for all blocks covering the union.

Briefly, we partition all atoms of the biomolecule into 3D rectangular blocks, using a *kd*-tree algorithm. Our goal is to apply the sequential algorithm of AlphaMol to each block, using a different processor for each block. We note, however, that atoms at the edges of a block may interact with atoms from neighboring blocks. If those atoms are not included in the calculation, the intrinsic volumes found for the atoms in the block will be inexact. We therefore expand each block with a buffer zone to include the atoms neighboring the block. The width of this buffer zone is set to $2R_{\max}$, in which $R_{\max}$ is the radius of the largest ball in the union of balls representing the biomolecule. This value ensures that all atoms that potentially interact with the atoms in the block will be accounted for. We then apply the AlphaMol algorithm to all atoms in the block and in its buffer zone. We finally retain the intrinsic volumes of the atoms in the block. The choice of the buffer zone guarantees that simplices sharing a vertex in the block are exactly the simplices sharing this vertex in the complete Delaunay triangulation, a property that does not hold for the vertices in the buffer zone. It follows that the intrinsic volumes of the balls whose centers lie in the block are computed correctly, while those of the balls whose centers lie in the buffer zone may be incorrect. Each block and its buffer is dealt with on a different processor, and as the partitioning of the atoms is balanced, we expect a speed-up proportional to the number of blocks.

Figure 8 shows the wall time for the execution of AlphaMol on the capsid of faustovirus (PDB code 5j7z,[81] 26 million atoms), with different numbers of threads requested by the software. The virus capsid is divided into m partitions based on a *kd*-tree, and each partition is handled separately on a different thread. Computations were performed on two different multicore computers, one with an Intel Xeon multicore CPUs running at 2.70 GHz, with 96 cores (192 threads), and a second with an AMD Threadripper multicore CPUs running at 2.2 GHz, with 32 cores (64 threads). The AMD computer is more recent than the Intel computer. As expected, we observe a significant speed-up when AlphaMol is run on multiple threads. The gain in time is significant. It takes 605 s with a single Intel Xeon thread and 321 s with a single AMD thread to compute the weighted volumes and their derivatives for the capsid of faustovirus. In comparison, it takes 161 or 82.3 s on four Intel threads or four AMD threads, respectively, and it only takes 30.9 or 21.2 s to do the same calculation using 32 Intel threads or 32 AMD threads, respectively. The speed-up is nearly linear between 1 and 32 threads for both types of processors. We note, however, that the speed-up is marginal between 32 threads and 64 threads for the AMD processor and between 64 threads and 128 threads for the Intel processor (see Figure 8). We believe that this behavior is reflective of the computer architecture, and not of the software itself. The AMD processor has 64 threads, but only 32 physical cores. We do not expect that two threads on a single core will be significantly faster than a single thread on that core, as those two threads share the same resources. The Intel processor has 96 physical cores. However, those cores are spread equally over four sockets, each with 24 cores. As such, we also expect a slow down in speed-up for large numbers of threads that need to share the same memory resources.

To reduce the risk that our conclusions are anecdotal, we repeated the calculations on all 68 virus capsids in our database. We only used the Intel computer, but found similar results with the AMD computer. Figure 9 shows the wall time as well as the speed-up for the execution of AlphaMol on the virus capsids, with different numbers of threads. Just like for the faustovirus, we observe a significant speed-up when multiple threads are added. This speed-up is nearly linear up to 32 cores, and then slows down. There are two reasons for the slow down in speed-up as more threads are added, namely, the architecture of the computer and progressively more redundant computations for larger numbers of threads. To illustrate the latter, we use two ways to calculate the speed-up. First, we compare the wall time with the total CPU time, represented as CLOCK/CPU. Based on this notion, the speed-up is nearly linear up to 32 processors, and slows down thereafter, still reaching 40 for 64 processors. It shows that there is little communication between the master processor and the at least 32 additional processors, indicating that the parallelism is effective. Second, we compare the wall time when running on k processors, with the running time from a serial (i.e., one processor) independent calculation. Based on this second notion, the speed-up is less effective, with 64 processors not being significantly more effective than 32 processors. The two notions should give similar results if the splitting is balanced so that each processor deals with about N/k atoms, N being the total number of atoms in the biomolecule. This is not the case, however, as the algorithm adds a buffer zone to each partition to make sure that the computation is correct. Each atom in the buffer zones will be considered at least twice,

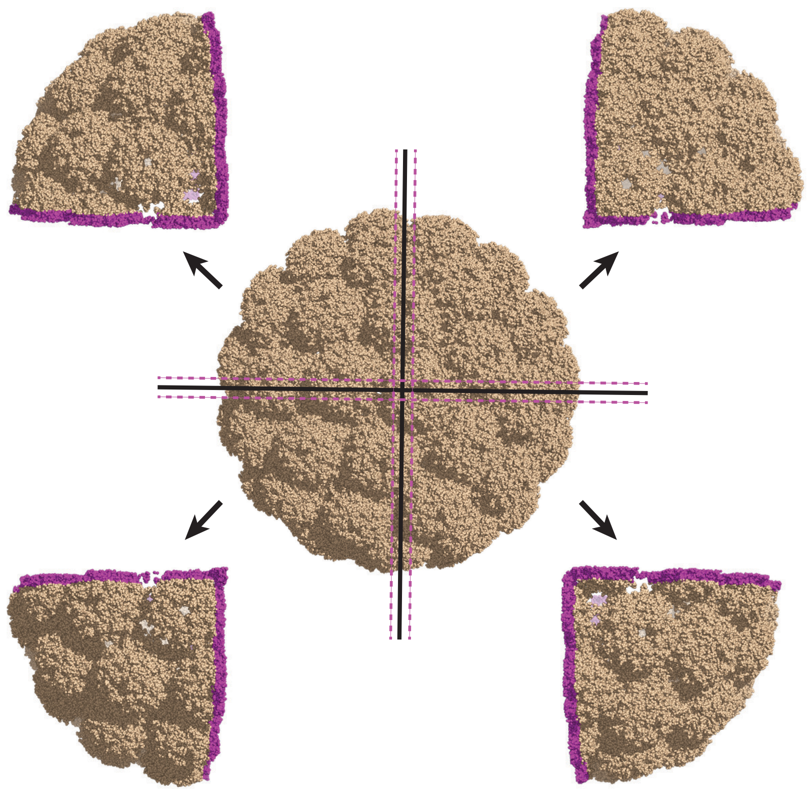


Figure 7. Splitting the computation of the intrinsic volumes of the murine polyomavirus (PDB code 1sid) over four processors. The 1,020,180 atoms are divided into four partitions of approximately equal size, using a *kd*-tree algorithm. Each partition correspond to a rectangular block. Each block is complemented with a buffer zone, such that atoms in this buffer zone (shown in *magenta*) may interact with the atoms of the block. Each block together with its buffer zone is assigned to one processor, which then runs the full AlphaMol algorithm. As each block with buffer zone includes approximately one-quarter of the atoms, and the computations on the blocks are run in parallel, it is expected that the total computation time be reduced by a factor of 4.

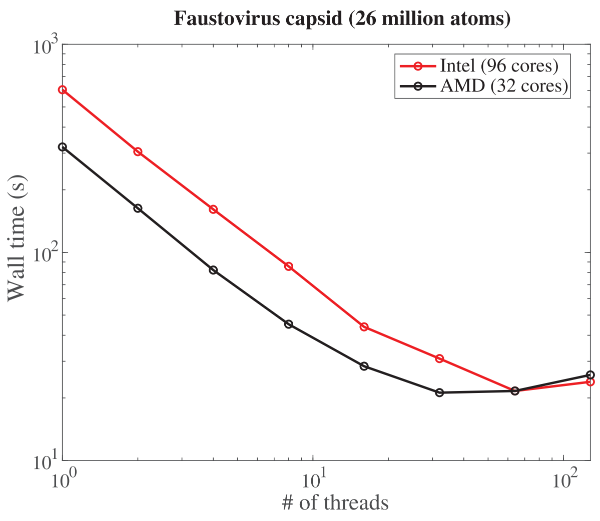


Figure 8. Execution (wall) time of AlphaMol as a function of the number threads requested, for computing the intrinsic volumes and their derivatives of the capsid of the faustovirus (PDB code 5j7v), which consists of approximately 26 million atoms. Computations were performed on an Intel Xeon processor at 2.70 GHz, with 96 cores (192 threads) (red), or an AMD Threadripper processor at 2.2 GHz, with 32 cores (64 threads).

since it also belongs to one of the partitions. As expected, the number of atoms treated at least twice increases with the number of processors; see Figure 10. This leads to redundant computations whose importance increases when the number of processors increases, ultimately leading to a plateau in the speed-up brought by the parallelism. It remains that we do reach a significant speed-up through this parallelism, with an average factor of 20-folds with 32 processors.

■ CONCLUSION

The Alpha Shape theory provides an accurate and robust method for computing the geometric measures of a biomolecule.[59−64] Among these measures, the intrinsic volumes are used to quantify the interaction between a biomolecule and surrounding water in the so-called morphometric model.[33−35] Several implementations of the Alpha Shape theory for measuring biomolecules exist, including our own, AlphaVol[62] and UnionBall.[67] These implementations, however, were limited to computing the volumes and surface areas of biomolecules.

In this paper, we have derived homogeneous formulas for the expressions of all four intrinsic volumes and their derivatives and implement them into a new package, AlphaMol. The only variables in these formulas are the interatomic distances, making them insensitive to translations and rotations. Recent spectacular advances in structural biology have produced an abundance of data on large macromolecular complexes, such as full size virus capsids[82] that contain several millions of atoms. Modeling these large systems is as important as modeling smaller proteins or nucleic acids; see for example the simulations of the HIV capsid that include over 60 million atoms.[17] To make sure that AlphaMol remains practical, we have adapted its underlying

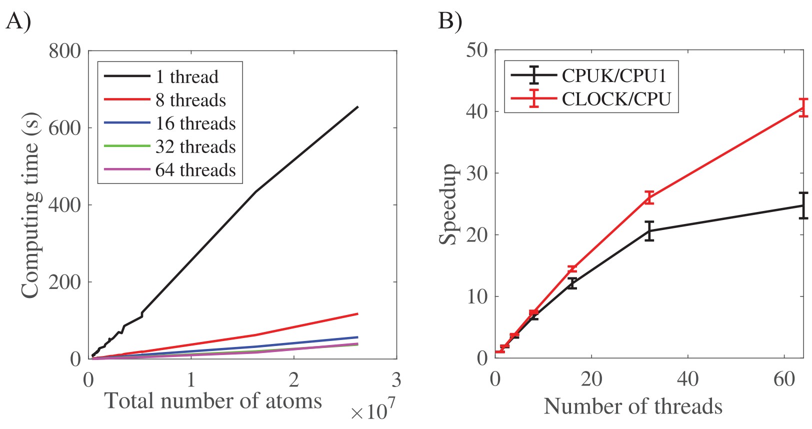


Figure 9. (A) Execution (wall) time of AlphaMol as a function of the number of atoms for different numbers of processors. (B) The speedup (computed as the ratio of total CPU time over wall time) is plotted against the number of threads used by the program. Computations were performed on an Intel Xeon processor at 2.70 GHz, with 96 cores (192 threads).

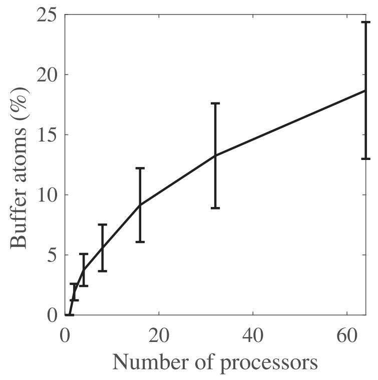


Figure 10. Percentages of atoms in the buffer zones as a fraction of the total number of atoms for the parallel version of AlphaMol as a function of the number of processors. An atom is said to belong to the buffer zones if it belongs to at least one of the buffer zones. The percentage increases with the number of processors, leading to progressively more redundant computations that ultimately limit the impact of using multiple processors.

algorithms in two ways. First, we have included an ordering scheme based on Hilbert curves to improve the localities of the atoms as they are introduced sequentially for generating the Delaunay triangulation of the atom centers, which is at the core of the Alpha Shape theory. Second, we have introduced a parallel version of AlphaMol, which partitions the atoms of the biomolecule of interest into 3D rectangular blocks, using a *kd*-tree algorithm. We then apply the sequential algorithm of AlphaMol to each block, augmented by a buffer zone to account for atoms that may overlap with atoms in the block. The presence of the buffer zones, however, leads to redundant computations that ultimately limit the impact of using multiple processors. In our current version, 32 processors led to a significant speed-up (20 times on average for 68 virus capsids ranging from 400,000 atoms to 26 million atoms), with marginal improvements for a higher number of processors.

Ultimately, we would like to push the parallelism to hundreds of processors, such as those that are available on a GPU. Recently, there was an attempt to do so for computing the Alpha Shape of a molecule,[83] using a mixed CPU-GPU algorithm showing speed-ups in the order of 25, which is similar to what we observed for multiple CPUs. We do see, however, some roadblocks for a GPU-only implementation of AlphaMol. To our knowledge, all current GPU implementations of the 3D Delaunay triangulation compute a near-Delaunay structure on the GPU, followed by transformations on the CPU to generate a valid Delaunay triangulation.[84] In addition, computing a valid Delaunay triangulation requires robust geometric predicates to account for possible degeneracies. In our implementation, we rely on the Simulation of Simplicity (SoS) to remove those degeneracies.[66] The SoS method is based or arbitrary precision arithmetics. There are currently only limited GPU implementations of arbitrary precision arithmetics, and they are still under development. Developing a GPU-based computation of intrinsic volumes of biomolecules remains, however, a priority for molecular simulations of very large biomolecular systems.

■ ASSOCIATED CONTENT

Data Availability Statement

All PDB files for the virus capsid structures are available from the Protein Data Bank, http://www.rcsb.org. The sequential version of AlphaMol is available as OpenSource software on github (https://github.com/pkoehl/AlphaMol).

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jcim.2c01346.

Proofs of the propositions in main text of the document: surface areas and volumes of ball intersections and derivatives; integrated mean curvature and derivatives; integrated Gaussian curvature and derivatives; and geometry of a tetrahedron. Table S.1: Database of virus capsid structures that are used in this study. (PDF)

■ AUTHOR INFORMATION

Corresponding Authors

Patrice Koehl − *Department of Computer Science, University of California, Davis, California 95616, United States*; orcid.org/0000-0002-0908-068X; Email: koehl@cs.ucdavis.edu

Arseniy Akopyan − *FORA Capital, Miami, Florida 33131, United States*; Email: akopjan@gmail.com

Herbert Edelsbrunner − *IST Austria, 3400 Klosterneuburg, Austria*; Email: edels@ist.ac.at

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jcim.2c01346

Funding

Open Access is funded by the Austrian Science Fund (FWF).

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

P.K. acknowledges support from the University of California Multicampus Research Programs and Initiatives (Grant No. M21PR3267) and from the NSF (Grant No.1760485). H.E. acknowledges support from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program, Grant No. 788183, from the Wittgenstein Prize, Austrian Science Fund (FWF), Grant No. Z 342-N31, and from the DFG Collaborative Research Center TRR 109, 'Discretization in Geometry and Dynamics', Austrian Science Fund (FWF), Grant No. I 02979-N35.

REFERENCES

(1) Senior, A.; Evans, R.; Jumper, J.; Kirkpatrick, J.; Sifre, L.; Green, T.; Qin, C.; Žídek, A.; Nelson, A.; Bridgland, A.; Penedones, H.; Petersen, S.; Simonyan, K.; Crossan, S.; Kohli, P.; Jones, D.; Silver, D.; Kavukcuoglu, K.; Hassabis, D. Improved protein structure prediction using potentials from deep learning. *Nature* **2020**, *577*, 706−710.

(2) Jumper, J.; Evans, R.; Pritzel, A.; Green, T.; Figurnov, M.; Ronneberger, O.; Tunyasuvunakool, K.; Bates, R.; Zidek, A.; Potapenko, A.; Bridgland, A.; Meyer, C.; Kohl, S. A. A.; Ballard, A. J.; Cowie, A.; Romera-Paredes, B.; Nikolov, S.; Jain, R.; Adler, J.; Back, T.; Petersen, S.; Reiman, D.; Clancy, E.; Zielinski, M.; Steinegger, M.; Pacholska, M.; Berghammer, T.; Bodenstein, S.; Silver, D.; Vinyals, O.; Senior, A. W.; Kavukcuoglu, K.; Kohli, P.; Hassabis, D. Highly accurate protein structure prediction with AlphaFold. *Nature* **2021**, *596*, 583−589.

(3) Hassoun, S.; Jefferson, F.; Shi, X.; Stucky, B.; Wang, J.; Rosa, E., Jr Artificial intelligence for biology. *Integrative and Comparative Biology* **2022**, *61*, 2267−2275.

(4) Xu, Y.; Liu, X.; Cao, X.; Huang, C.; Liu, E.; Qian, S.; Liu, X.; Wu, Y.; Dong, F.; Qiu, C.-W.; Qiu, J.; Hua, K.; Su, W.; Wu, J.; Xu, H.; Han, Y.; Fu, C.; Yin, Z.; Liu, M.; Roepman, R.; Dietmann, S.; Virta, M.; Kengara, F.; Zhang, Z.; Zhang, L.; Zhao, T.; Dai, J.; Yang, J.; Lan, L.; Luo, M.; Liu, Z.; An, T.; Zhang, B.; He, X.; Cong, S.; Liu, X.; Zhang, W.; Lewis, J. P.; Tiedje, J. M.; Wang, Q.; An, Z.; Wang, F.; Zhang, L.; Huang, T.; Lu, C.; Cai, Z.; Wang, F.; Zhang, J. Artificial intelligence: A powerful paradigm for scientific research. *Innovation* **2021**, *2*, 100179.

(5) Callaway, E. It will change everything": DeepMind's AI makes gigantic leap in solving protein structures. *Nature* **2020**, *588*, 203−205.

(6) Jones, D.; Thornton, J. The impact of AlphaFold2 one year on. *Nat. Methods* **2022**, *19*, 15−20.

(7) Nussinov, R.; Zhang, M.; Liu, Y.; Jang, H. AlphaFold, Artificial Intelligence, and Allostery. *J. Phys. Chem. B* **2022**, *126*, 6372−6383.

(8) Berman, H. M.; Westbrook, J.; Feng, Z.; Gilliland, G.; Bhat, T. N.; Weissig, H.; Shindyalov, I.; Bourne, P. The Protein Data Bank. *Nucl. Acids. Res.* **2000**, *28*, 235−242.

(9) Varadi, M.; Anyango, S.; Deshpande, M.; Nair, S.; Natassia, C.; Yordanova, G.; Yuan, D.; Stroe, O.; Wood, G.; Laydon, A.; Zidek, A.; Green, T.; Tunyasuvunakool, K.; Petersen, S.; Jumper, J.; Clancy, E.; Green, R.; Vora, A.; Lutfi, M.; Figurnov, M.; Cowie, A.; Hobbs, N.; Kohli, P.; Kleywegt, G.; Birney, E.; Hassabis, D.; Velankar, S. AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic Acids Res.* **2022**, *50*, D439−D444.

(10) Shaw, D.; Deneroff, M.; Dror, R.; Kuskin, J.; Larson, R.; Salmon, J.; Young, C.; Batson, B.; Bowers, K.; Chao, J.; Eastwood, M.; Gagliardo, J.; Grossman, J.; Ho, C.; Ierardi, D.; Kolossváry, I.; Klepeis, J.; Layman, T.; McLeavey, C.; Moraes, M.; Mueller, R.; Priest, E.; Shan, Y.; Spengler, J.; Theobald, M.; Towles, B.; Wang, S. Anton, a Special-purpose Machine for Molecular Dynamics Simulation. *Commun. ACM* **2008**, *51*, 91−97.

(11) Stone, J.; Hardy, D.; Ufimtsev, I.; Schulten, K. GPU-accelerated molecular modeling coming of age. *J. Mol. Graph. Modelling* **2010**, *29*, 116−125.

(12) Wang, Y.; Harrison, C.; Schulten, K.; McCammon, J. Implementation of accelerated molecular dynamics in NAMD. *Comput. Sci. Discovery* **2011**, *4*, 015002.

(13) Pierce, L. C.T.; Salomon-Ferrer, R.; de Oliveira, C. A. F.; McCammon, J. A.; Walker, R. C. Routine access to millisecond time scale events with accelerated molecular dynamics. *J. Chem. Theori. Comput.* **2012**, *8*, 2997−3002.

(14) Sweet, J.; Nowling, R.; Cickovski, T.; Sweet, C.; Pande, V.; Izaguirre, J. Long Timestep Molecular Dynamics on the Graphical Processing Unit. *J. Chem. Theori. Comput.* **2013**, *9*, 3267−3281.

(15) Shaw, D. E.; Grossman, J. P.; Bank, J. A.; Batson, B.; Butts, J. A.; Chao, J. C.; Deneroff, M. M.; Dror, R. O.; Even, A.; Fenton, C. H.; Forte, A.; Gagliardo, J.; Gill, G.; Greskamp, B.; Ho, C. R.; Ierardi, D. J.; Iserovich, L.; Kuskin, J. S.; Larson, R. H.; Layman, T.; Lee, L. S.; Lerer, A. K.; Li, C.; Killebrew, D.; Mackenzie, K. M.; Mok, S. Y. H.; Moraes, M. A.; Mueller, R.; Nociolo, L. J.; Peticolas, J. L.; Quan, T.; Ramot, D.; Salmon, J. K.; Scarpazza, D. P.; Schafer, U. B.; Siddique, N.; Snyder, C. W.; Spengler, J.; Tang, P. T. P.; Theobald, M.; Toma, H.; Towles, B.; Vitale, B.; Wang, S. C.; Young, C.Anton 2: Raising the Bar for Performance and Programmability in a Special-Purpose Molecular Dynamics Supercomputer. SC14: International Conference for High Performance Computing, Networking, Storage and Analysis; In *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*; IEEE, 2014; pp 41−53.

(16) Eastman, P.; Pande, V. OpenMM: A Hardware Independent Framework for Molecular Simulations. *Comput. Sci. Eng.* **2015**, *12*, 34−39.

(17) Zhao, G.; Perilla, J.; Yufenyuy, E.; Meng, X.; Chen, B.; Ning, J.; Ahn, J.; Gronenborn, A.; Schulten, K.; Aiken, C.; Zhang, P. Mature HIV-1 capsid structure by cryo-electron microscopy and all-atom molecular dynamics. *Nature* **2013**, *497*, 643−646.

(18) Sener, M.; Strumpfer, J.; Singharoy, A.; Hunter, C.; Schulten, K. Overall energy conversion efficiency of a photosynthetic vesicle. *Elife* **2016**, *5*, na DOI: 10.7554/eLife.09541.

(19) Phillips, J.; Hardy, D.; Maia, J.; Stone, J.; Ribeiro, J.; Bernardi, R.; Buch, R.; Fiorin, G.; Hénin, J.; Jiang, W.; McGreevy, R.; Melo, M.; Radak, B.; Skeel, R.; Singharoy, A.; Wang, Y.; Roux, B.; Aksimentiev, A.; Luthey-Schulten, Z.; Kalé, L.; Schulten, K.; Chipot, C.; Tajkhorshid, E. Scalable molecular dynamics on CPU and GPU architectures with NAMD. *J. Chem. Phys.* **2020**, *153*, 044130.

(20) Jung, J.; Kobayashi, C.; Kasahara, K.; Tan, C.; Kuroda, A.; Minami, K.; Ishiduki, S.; Nishiki, T.; Inoue, H.; Ishikawa, Y.; Feig, M.; Sugita, Y. New parallel computing algorithm of molecular dynamics for extremely huge scale biological systems. *J. Comput. Chem.* **2021**, *42*, 231−241.

(21) Gupta, C.; Sarkar, D.; Tieleman, D.; Singharoy, A. The ugly, bad, and good stories of large-scale biomolecular simulations. *Curr. Opin. Struct. Biol.* **2022**, *73*, 102338.

(22) Eisenberg, D.; McLachlan, A. D. Solvation energy in protein folding and binding. *Nature (London)* **1986**, *319*, 199−203.

(23) Richards, F. M. Areas, volumes, packing, and protein-structure. *Annu. Rev. Biophys. Bioeng.* **1977**, *6*, 151−176.

(24) Kollman, P.; Massova, I.; Reyes, C.; Kuhn, B.; Huo, S.; Chong, L.; Lee, M.; Lee, T.; Duan, Y.; Wang, W.; Donini, O.; Cieplak, P.; Srinivasan, J.; Case, D.; Cheatham, T. Calculating structures and free energies of complex molecules: combining molecular mechanics and continuum models. *Acc. Chem. Res.* **2000**, *33*, 889−897.

(25) Lum, K.; Chandler, D.; Weeks, J. D. Hydrophobicity at small and large length scales. *J. Phys. Chem. B* **1999**, *103*, 4570−4577.

(26) Reiss, H.; Frisch, H.; Lebowitz, J. Statistical mechanics of rigid spheres. *J. Chem. Phys.* **1959**, *31*, 369−380.

(27) Lebowitz, J.; Helfand, E.; Praestgaard, E. Scaled particle theory of fluid mixtures. *J. Chem. Phys.* **1965**, *43*, 774−779.

(28) Rosenfeld, Y. Scaled field particle theory of the structure and the thermodynamics of isotropic hard particle fluids. *J. Chem. Phys.* **1988**, *89*, 4272−4287.

(29) Levy, R.; Zhang, L.; Gallicchio, E.; Felts, A. On the nonpolar hydration free energy of proteins: surface area and continuum solvent models for the solute- solvent interaction energy. *J. Am. Chem. Soc.* **2003**, *125*, 9523−9530.
(30) Wagoner, J.; Baker, N. Assessing implicit models for nonpolar mean solvation forces: the importance of dispersion and volume terms. *Proc. Natl. Acad. Sci. (USA)* **2006**, *103*, 8331−8336.
(31) Chen, J.; Brooks, C., III Implicit modeling of nonpolar solvation for simulating protein folding and conformational transitions. *Phys. Chem. Chem. Phys.* **2008**, *10*, 471−481.
(32) König, P.-M.; Roth, R.; Mecke, K. Morphological thermodynamics of fluids: shape dependence of free energies. *Phys. Rev. Lett.* **2004**, *93*, 160601.
(33) Roth, R.; Harano, Y.; Kinoshita, M. Morphometric approach to the solvation free energy of complex molecules. *Phys. Rev. Lett.* **2006**, *97*, 078101.
(34) Hansen-Goos, H.; Roth, R.; Mecke, K.; Dietrich, S. Solvation of proteins: linking thermodynamics to geometry. *Phys. Rev. Lett.* **2007**, *99*, 128101.
(35) Harano, Y.; Roth, R.; Chiba, S. A morphometric approach to the accurate solvation thermodynamics of proteins and ligands. *J. Comput. Chem.* **2013**, *34*, 1969−1974.
(36) Lee, B.; Richards, F. M. Interpretation of protein structures: estimation of static accessibility. *J. Mol. Biol.* **1971**, *55*, 379−400.
(37) Shrake, A.; Rupley, J. A. Environment and exposure to solvent of protein atoms. Lysozyme and insulin. *J. Mol. Biol.* **1973**, *79*, 351−371.
(38) Legrand, S. M.; Merz, K. M. Rapid approximation to molecular-surface area via the use of boolean logic and look-up tables. *J. Comput. Chem.* **1993**, *14*, 349−352.
(39) Wang, H.; Levinthal, C. A vectorized algorithm for calculating the accessible surface area of macromolecules. *J. Comput. Chem.* **1991**, *12*, 868−871.
(40) Futamura, N.; Aluru, S.; Ranjan, D.; Hariharan, B. Efficient parallel algorithms for solvent accessible surface area of proteins. *IEEE Trans. Parallel Dist. Syst.* **2002**, *13*, 544−555.
(41) Rowlinson, J. S. The triplet distribution function in a fluid of hard spheres. *Mol. Phys.* **1963**, *6*, 517−524.
(42) Pavani, R.; Ranghino, G. A method to compute the volume of a molecule. *Computers and Chemistry* **1982**, *6*, 133−135.
(43) Gavezzotti, A. The calculation of molecular volumes and the use of volume analysis in the investigation of structured media and of solid-state organic reactivity. *J. Am. Chem. Soc.* **1983**, *105*, 5220−5225.
(44) Till, M.; Ullmann, G. M. McVol - A program for calculating protein volumes and identifying cavities by a Monte Carlo algorithm. *J. Mol. Model.* **2010**, *16*, 419−429.
(45) Wodak, S. J.; Janin, J. Analytical approximation to the accessible surface-area of proteins. *Proc. Natl. Acad. Sci. (USA)* **1980**, *77*, 1736−1740.
(46) Hasel, W.; Hendrickson, T. F.; Still, W.C. A rapid approximation to the solvent accessible surface areas of atoms. *Tetrahed. Comp. Method.* **1988**, *1*, 103−116.
(47) Cavallo, L. POPS: a fast algorithm for solvent accessible surface areas at atomic and residue level. *Nucl. Acids. Res.* **2003**, *31*, 3364−3366.
(48) Street, A. G.; Mayo, S. L. Pairwise calculation of protein solvent-accessible surface areas. *Folding & Design* **1998**, *3*, 253−258.
(49) Weiser, J.; Shenkin, P. S.; Still, W. C. Approximate atomic surfaces from linear combinations of pairwise overlaps (LCPO). *J. Comput. Chem.* **1999**, *20*, 217−230.
(50) Dynerman, D.; Butzlaff, E.; Mitchell, J. CUSA and CUDE: GPU-accelerated methods for estimating solvent accessible surface area and desolvation. *J. Comput. Biol.* **2009**, *16*, 523−537.
(51) Wawak, R. J.; Gibson, K. D.; Scheraga, H. A. Gradient discontinuities in calculations involving molecular-surface area. *J. Math. Chem.* **1994**, *15*, 207−232.
(52) Gogonea, V.; Osawa, E. An improved algorithm for the analytical computation of solvent-excluded volume. The treatment of singularities in solvent-accessible surface-area and volume functions. *J. Comput. Chem.* **1995**, *16*, 817−842.
(53) Richmond, T. J. Solvent accessible surface-area and excluded volume in proteins. Analytical equations for overlapping spheres and implications for the hydrophobic effect. *J. Mol. Biol.* **1984**, *178*, 63−89.
(54) Connolly, M. L. Computation of molecular volume. *J. Am. Chem. Soc.* **1985**, *107*, 1118−1124.
(55) Dodd, L. R.; Theodorou, D. N. Analytical treatment of the volume and surface area of molecules formed by an arbitrary collection of unequal spheres intersected by planes. *Mol. Phys.* **1991**, *72*, 1313−45.
(56) Irisa, M. An elegant algorithm of the analytical calculation for the volume of fused spheres with different radii. *Comput. Phys. Commun.* **1996**, *98*, 317−338.
(57) Vassetti, D.; Civalleri, B.; Labat, F. Analytical calculation of the solvent-accessible surface area and its nuclear gradients by stereographic projection: A general approach for molecules, polymers, nanotubes, helices, and surfaces. *J. Comput. Chem.* **2020**, *41*, 1464−1479.
(58) Duan, X.; Quan, C.; Stamm, B. A boundary-partition-based Voronoi diagram of *d*-dimensional balls: definition, properties, and applications. *Adv. Comput. Math.* **2020**, *46* (44), 1−25.
(59) Edelsbrunner, H. The union of balls and its dual shape. *Discrete Comput. Geom.* **1995**, *13*, 415−440.
(60) Edelsbrunner, H.; Koehl, P. The weighted-volume derivative of a space-filling diagram. *Proc. Natl. Acad. Sci. (USA)* **2003**, *100*, 2203−2208.
(61) Bryant, R.; Edelsbrunner, H.; Koehl, P.; Levitt, M. The area derivative of a space-filling diagram. *Discrete Comput. Geom* **2004**, *32*, 293−308.
(62) Edelsbrunner, H.; Koehl, P. The geometry of biomolecular solvation. *Combinatorial and Computational Geometry* **2005**, *52*, 243−275.
(63) Akopyan, A.; Edelsbrunner, H. The Weighted Mean Curvature Derivative of a Space-Filling Diagram. *Comput. Math. Biophys.* **2020**, *8*, 51−67.
(64) Akopyan, A.; Edelsbrunner, H. The Weighted Gaussian Curvature Derivative of a Space-Filling Diagram. *Comput. Math. Biophys.* **2020**, *8*, 74−88.
(65) Carrillo-Tripp, M.; Shepherd, C. M.; Borelli, I. A.; Venkataraman, S.; Lander, G.; Natarajan, P.; Johnson, J. E.; Brooks, C. L.; Reddy, V. S. VIPERdb2: an enhanced and web API enabled relational database for structural virology. *Nucl. Acids. Res.* **2009**, *37*, D436−D442.
(66) Edelsbrunner, H.; Mücke, E. P. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graphics* **1990**, *9*, 66−104.
(67) Mach, P.; Koehl, P. Geometric measures of large biomolecules: surface, volume, and pockets. *J. Comput. Chem.* **2011**, *32*, 3023−3038.
(68) Edelsbrunner, H.; Mücke, E. P. Three-dimensional Alpha Shapes. *ACM Trans. Graphics* **1994**, *13*, 43−72.
(69) Kaminski, G. A.; Friesner, R. A.; Tirado-Rives, J.; Jorgensen, W. L. Evaluation and reparametrization of the OPLS-AA force field for proteins via comparison with accurate quantum chemical calculations on peptides. *J. Phys. Chem. B* **2001**, *105*, 6474−6487.
(70) Edelsbrunner, H.; Shah, N. R. Incremental topological flipping works for regular triangulations. *Algorithmica* **1996**, *15*, 223−241.
(71) Edelsbrunner, H.*Weighted Alpha Shapes*; Technical Report UIUC-CS-R-92-1760; University of Illinois, Urbana, IL, 1992.
(72) Amenta, N.; Choi, S.; Rote, G.Incremental constructions con BRIO. In *SoCG03: Annual ACM Symposium on Computational Geometry*; ACM, 2003; pp 211−219.
(73) Liu, Y.; Snoeyink, J. A comparison of five implementations of 3D Delaunay tessalation. *Combinatorial and Computational Geometry* **2005**, *52*, 439−458.
(74) Cignoni, P.; Montani, C.; Perego, R.; Scopigno, R. Parallel 3D Delaunay triangulation. *Computer Graphics Forum* **1993**, *12*, 129−142.
(75) Hardwick, J. C.Implementation and Evaluation of an Efficient Parallel Delaunay Triangulation Algorithm. In *Ninth Annual ACM Symposium on Parallel Algorithms and Architectures*; ACM, 1997; pp 239−248.
(76) Chen, M.-B.The merge phase of parallel divide-and-conquer scheme for 3D Delaunay triangulation. In *International Symposium on*

Parallel and Distributed Processing with Applications (ISPA); IEEE, 2010; pp 224−230.

(77) Fuetterling, V.; Lojewski, C.; Pfreundt, F.-J.High-Performance Delaunay Triangulation for Many-Core Computers. In *Proceedings of High Performance Graphics*; ACM, 2014; p 97−104.

(78) Lo, S. 3D Delaunay triangulation of 1 billion points on a PC. *Finite Elements in Analysis and Design* **2015**, *102−103*, 65−73.

(79) Lin, J.; Chen, R.; Yang, C.; Shu, Z.; Wang, C.; Lin, Y.; Wu, L.Distributed and Parallel Delaunay Triangulation Construction with Balanced Binary-tree Model in Cloud. In *2016 15th International Symposium on Parallel and Distributed Computing*; IEEE, 2016; pp 107−113.

(80) Nguyen, C.; Rhodes, P. Delaunay triangulation of large-scale datasets using two-level parallelism. *Parallel Computing* **2020**, *98*, 102672.

(81) Klose, T.; Reteno, D.; Benamar, S.; Hollerbach, A.; Colson, P.; La Scola, B.; Rossmann, M. Structure of faustovirus, a large dsDNA virus. *Proc. Natl. Acad. Sci. (USA)* **2016**, *113*, 6206−6211.

(82) Montiel-Garcia, D.; Santoyo-Rivera, N.; Ho, P.; Carrillo-Tripp, M.; Brooks, C. L., III; Johnson, J. E; Reddy, V. S VIPERdb v3.0: a structure-based data analytics platform for viral capsids. *Nucl. Acids. Res.* **2021**, *49*, D809−D816.

(83) Masood, T. B.; Ray, T.; Natarajan, V. Parallel computation of alpha complexes for biomolecules. *Comput. Geom.* **2020**, *90*, 101651.

(84) Cao, T.-T.; Nanjappa, A.; Gao, M.; Tan, T.-S.A GPU Accelerated Algorithm for 3D Delaunay Triangulation. In *Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*; ACM, 2014; pp 47−54.