JCTC Journal of Chemical Theory and Computation

TMSmesh: A Robust Method for Molecular Surface Mesh Generation Using a Trace Technique

Minxin Chen*,† and Benzhuo Lu*,‡

Department of Mathematics, SooChow University, Suzhou 215006, China and State Key Laboratory of Scientific/Engineering Computing, Institute of Computational Mathematics, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China

Received July 4, 2010

Abstract: Qualified, stable, and efficient molecular surface meshing appears to be necessitated by recent developments for realistic mathematical modeling and numerical simulation of biomolecules, especially in implicit solvent modeling (e.g., see a review in B. Z. Lu et al. *Commun. Comput. Phys.* **2008**, *3*, 973−1009). In this paper, we present a new method: tracing molecular surface for meshing (TMSmesh) the Gaussian surface of biomolecules. The method computes the surface points by solving a nonlinear equation directly, polygonizes by connecting surface points through a trace technique, and finally outputs a triangulated mesh. TMSmesh has a linear complexity with respect to the number of atoms and is shown to be capable of handling molecules consisting of more than one million atoms, which is usually difficult for the existing methods for surface generation used in molecular visualization and geometry analysis. Moreover, the meshes generated by TMSmesh are successfully tested in boundary element solutions of the Poisson−Boltzmann equation, which directly gives rise to a route to simulate electrostatic solvation of large-scale molecular systems. The binary version of TMSmesh and a set of representative PQR benchmark molecules are downloadable at our Web page http://lsec.cc.ac.cn/∼lubz/Meshing.html.

Introduction

Molecular surface mesh is widely used for visualization and geometry analysis in computational structural biology and structural bioinformatics. Recent developments in realistic mathematical modeling and numerical simulation of biomolecular systems raise new demands for qualified, stable, and efficient surface meshing, especially in implicit-solvent modeling (e.g., see a review in ref 1). Main concerns for improvement on existing methods for molecular surface mesh generation are efficiency, robustness, and mesh quality. Efficiency is necessary for simulations/computations requiring frequent mesh generation or requiring mesh of large systems. Robustness here means the meshing method is stable and can treat various, even arbitrary, sizes of molecular systems within computer power limitations. Mesh quality relates to mesh smoothness (avoiding sharp solid angles, etc.), uniformness (avoiding elements with very sharp angles or zero area), and topological correctness (avoiding isolated vertices, element intersection, single-element-connected edges, etc.). The quality requirement is critical for some numerical techniques, such as finite element method, to achieve converged and reasonable results, which makes it a more demanding task in this respect than the mesh generations only for the purposes of visualization or some structural geometry analysis.

Various definitions of molecular surface, including the van der Waals (VDW) surface, solvent accessible surface (SAS), solvent excluded surface (SES), molecular skin surface,[2] minimal molecular surface,[3] and Gaussian surface, etc., have been proposed to describe the shapes of molecular structure. The VDW surface is defined as the surface of the union of the spherical atomic surfaces with the VDW radius of each atom in the molecule. The SAS and SES are represented by

* Corresponding authors. E-mail: chenmx@gmail.com, bzlu@lsec.cc.ac.cn.

† SooChow University.

‡ Chinese Academy of Sciences.

10.1021/ct100376g
Published on Web 11/30/2010

the trajectory of the center and the interboundary of a rolling probe on the VDW surface, respectively. The molecular skin surface is the envelope of an infinite family of spheres derived from atoms by convex combination and shrinking. The minimal molecular surface is defined as a result of the surface free energy minimization. Different from these definitions, the Gaussian surface is defined as a level set of the summation of the Gaussian kernel functions as follows

$$\{\vec{x} \in \mathbb{R}^3, \phi(\vec{x}) = t_0\} \tag{1}$$

where

$$\phi(\vec{x}) = \sum_{i=1}^{N} e^{d(||\vec{x} - \vec{c}_i||^2/r_i^2 - 1)} \tag{2}$$

$\vec{c}_i$ and r_i are the location and radius of atom i, the parameter d is negative and controls the decay speed of the kernel functions. When $|d|$ is increased, the resulting Gaussian surface is closer to the VDW surface. In this work, the value of d and t_0 are set as -1 and 1, respectively. Compared with other definitions of molecular surface, the Gaussian surface is smooth and more suitable to represent the electron density of a molecule.[4] The VDW surface, SAS, and SES can be approximated well by the Gaussian surface with proper parameter selection.[4,5] The Gaussian surface has been widely used in many problems in computational biology, such as docking problems,[6] molecular shape comparisons,[7] calculating SAS areas,[8] and the generalized Born models.[9]

With various definitions of molecular surface that have been proposed, numerous works have been devoted to the computation of molecular surface. The representative ones are described as follows. In 1983, Connolly proposed algorithms to calculate the molecular surface and SAS analytically.[10,11] In 1995, a popular program, GRASP, for visualizing molecular surfaces was presented.[12] In 1997, Vorobjev et al. proposed SIMS, a method of calculating a smooth invariant molecular dot surface, in which an exact method for removing self-intersecting parts and smoothing the singular regions of the SES was presented.[13] Sanner et al. presented a tool based on α shapes,[14] named MSMS, for meshing the SES.[15] Ryu et al. proposed a method based on β shapes that are a generalization of α shapes.[16] More recently, Zhang et al. used a modified dual contouring method to generate mesh for biomolecular structures,[17] and a later tool, GAMer, was developed for improving the mesh quality.[18] Can et al. proposed LSMS to generate the SES on grid points using level-set methods.[19] Chavent et al. presented MetaMol to visualize the molecular skin surface using the ray-casting method,[20] and Cheng et al. used restricted union of balls to generate mesh for molecular skin surfaces.[21] So far, these methods or tools usually successfully calculated different surfaces of small- or medium-sized biomolecules, but they are not suitable for large molecules with more than hundreds of thousands of atoms. Moreover, most of these methods, such as GRASP, MSMS, and LSMS were designed for molecular visualization and geometry analysis in computational structure biology or structural bioinformatics. Among those, MSMS is the most widely used one for molecular surface triangulation because of its high efficiency. However, the generated mesh is not a manifold and is composed of very irregular triangles. For some numerical modeling using, for instance, finite element/boundary element methods, the mesh quality usually needs to be improved through mesh topology checking (picking out the irregular nodes/edges/elements and rearranging the mesh), surface mesh smoothing, and so on.[1] In this paper, we develop a robust method, named TMSmesh that is capable of finishing meshing the Gaussian surface for biomolecules consisting of more than one million atoms in 30 min on a typical 2010 PC, and the mesh quality is shown to be applicable to boundary element method simulations of biomolecular electrostatics.

As the Gaussian surface is an implicit surface, the existing techniques for triangulating implicit surface can be used for the Gaussian surface. These methods are divided into two main categories: spatial partition and continuation methods. The well-known marching cubes[22] and dual contouring methods[23] are examples of the spatial partition methods. This kind of method divides the space into cells and polygonizes the implicit surface in the cell whose vertices have different signs of the implicit function. An assumption is required that the implicit function is linear in the cell. As shown in the following sections, TMSmesh does not require this assumption. The continuation methods[24−26] are of another category. These methods mesh the implicit surface by growing current polygonization's border through the predictor−corrector method, which predicts the next surface point in the tangent direction of the current one and corrects it on the surface. The predictor−corrector method is used in TMSmesh to generate the next corrected point on the surface from current one, and the topology connection is confirmed by checking the continuity between the corrected and the current points, otherwise we restart the predictor−corrector from the current point with a smaller step size, until the continuity is fulfilled. The above process is defined as the trace technique in this paper, and it can be seen as a generalization of the predictor−corrector method. The quality of mesh triangles is well controlled in continuation methods, but techniques for avoiding overlapping, filling the gap between adjacent branches, and selecting proper initial triangles are required. In TMSmesh, no problems of overlapping, gap filling, and selecting initial seeds need to be considered, because the Gaussian surface is polygonized by connecting presampled surface points.

This paper is organized as follows. In Method Section, we present our method for polygonizing the Gaussian surface. Some examples and applications are presented in the Results Section. The final section, Conclusion, gives some concluding remarks.

Method

In this section, we describe the algorithm for meshing the Gaussian surface. Our algorithm contains two stages. The first stage is to compute the points on the surface by solving a nonlinear equation $\phi(\vec{x}) = t_0$. The second stage is to polygonize the Gaussian surface by connecting the generated points. In the following subsections, each stage is described in detail.

Computing the Points on the Gaussian Surface. From the definition of Gaussian surface, the points on the Gaussian surface are the roots of nonlinear equation $\phi(x, y, z) = t_0$, where $\phi(x, y, z)$ is defined in eq 2. Therefore, solving $\phi(x, y, z) = t_0$ is equivalent to computing the points on the Gaussian surface. In this method the equation is solved by the following steps.

Suppose the molecule is placed on a three-dimensional orthogonal grid consisting of $n_x \times n_y \times n_z$ cubes. For an arbitrary cube $[x_i, x_i + h] \times [y_i, y_i + h] \times [z_i, z_i + h]$, where (x_i, y_i, z_i) is the lower-left front corner, and h is the edge length of the cube. To decide whether the cube has an intersection with the surface, we proposed the following lower and upper bounds of $\phi(x)$ in the cube for Gaussian surface:

$$L_i = \sum_{k=1}^{N} e^{-d} L_{k,i}^x L_{k,i}^y L_{k,i}^z \le \phi(x, y, z) \le \sum_{k=1}^{N} e^{-d} U_{k,i}^x U_{k,i}^y U_{k,i}^z = U_i \quad (3)$$

for $(x, y, z) \in [x_i, x_i + h] \times [y_i, y_i + h] \times [z_i, z_i + h]$, where

$$U_{k,i}^{\alpha} = \begin{cases} 1, & c_\alpha^k \in [\alpha_i, \alpha_i + h] \\ \max\{e^{d(\alpha_i - c_\alpha^k)^2/r_k^2}, e^{d(\alpha_i + h - c_\alpha^k)^2/r_k^2}\}, & c_\alpha^k \notin [\alpha_i, \alpha_i + h] \end{cases} \quad (4)$$

$$L_{k,l}^{\alpha} = \min\{e^{d(\alpha_i - c_\alpha^k)^2/r_k^2}, e^{d(\alpha_i + h - c_\alpha^k)^2/r_k^2}\} \quad (5)$$

with $\alpha \in \{x, y, z\}$ and $\vec{c}_k = (c_x^k, c_y^k, c_z^k)$. $U_{k,i}^{\alpha}$ and $L_{k,i}^{\alpha}$ are the upper and lower bounds along α-dimension of the kernel located at atom k in the cube, respectively. $U_{k,i}^{\alpha}$ and $L_{k,i}^{\alpha}$ take either 1 or a value of the kernel at the boundary of the cube. If $t_0 \in [L_i, U_i]$, then the Gaussian surface $\phi(x, y, z) = t_0$ may have an intersection with cube $[x_i, x_i + h] \times [y_i, y_i + h] \times [z_i, z_i + h]$, otherwise there is no surface point in the cube. Note that the upper-bound U_i and the lower-bound L_i depend on the edge length of the cube h. The bounds are sharper when h is smaller. Above estimation is easy to combine with an octree data structure to decide intersection more adaptively.

The above estimation technique allows the deletion of the majority of cubes, which do not intersect the surface. In each of the left cubes, some surface points are sampled through root finding. Suppose the cube $[x_{i_0}, x_{i_0} + h] \times [y_{i_0}, y_{i_0} + h] \times [z_{i_0}, z_{i_0} + h]$ is one of them, we solve the nonlinear equation $\phi_{ij}(x) \triangleq \phi(x, y_{i_0} + i\tilde{h}, z_{i_0} + j\tilde{h}) = t_0$, for each $\{i, j\}$, $i, j = 1, ..., [h/\tilde{h}]$. The $\tilde{h}$ is to control the vertex density of the mesh. To find the roots, $\phi(x, y_i, z_j)$, $x \in [x_{i_0}, x_{i_0} + h]$, is approximated by the following Mth-degree polynomial:

$$p_{ij}(2(x - x_{i_0})/h + 1) = \sum_{n=1}^{M} (2i + 1) a_n L_n[2(x - x_{i_0})/h + 1]/2 \quad (6)$$

where $a_n = \int_{-1}^{1} \phi[hx/2 + (x_{i_0} + h/2), y_i, z_j] L_n(x) dx$, $L_n(x)$ is the nth-degree Legendre polynomial, M is set as 10, and h is 4 Å in our work. Then $p_{ij}[2(x - x_{i_0})/h + 1] = t_0$ is solved using Jenkins−Traub method.[27] The real roots of $p_{ij}[2(x - x_{i_0})/h + 1] = t_0$ in $[x_{i_0}, x_{i_0} + h]$ should be checked by $|\phi(x, y_i, z_j) - t_0| < \varepsilon$ (ε is an error tolerance) and be improved by Newton iterations, if needed. This process may lose some roots of $\phi_{ij}(x) = t_0$, due to approximation of $p_{ij}[2(x - x_{i_0})/h + 1]$ to $\phi_{ij}(x)$, but they would be found through trace processes in the polygonization stage.

Trace Step. In this subsection, the trace step employed in polygonization stage is described in detail. The objective of the trace step is to connect two (previously identified) grid−surface intersection points. In the trace step, the next connected surface point is predicted and corrected from the initial point with an initial step size, and the connection is confirmed through checking the continuity between the two points. If the continuity is not fulfilled, then restart the prediction and the correction process from the initial point with a smaller step size. Because every trace step is performed either on xy- or yz-planes between lines parallel to the x-axis in the polygonizaton stage, we discuss the details of the trace step on the xy-plane between two lines parallel to the x-axis as an example. Suppose there are two lines $y = y_0$ and $y = y_1$ on the xy-plane with four surface points (x_0, y_0), (x_1, y_0), (x_2, y_1), (x_3, y_1) on them and $y_1 > y_0$, $\phi_x(x_0, y_0) > 0$, Algorithm 1 connects (x_0, y_0) and (x_1, y_0) (see Figure 1). Moreover, the extreme point (x^*, y^*) is caught during the trace step to preserve the details of the surface.

In the step 2 of Algorithm 1, $\vec{p}_0'$ is the predicted surface point from $\vec{p}_0$ along the tangent direction at $\vec{p}_0$ with step size h_2. Step 3 is to correct $\vec{p}_0'$ back to the surface along the gradient direction of $\phi(x)$ at $\vec{p}_0'$. Step 4 is to check whether $\vec{p}_1$ is an extreme point along the x-direction. In step 5, condition ($\bigstar_1$) is used to determine if the step-size h_2 is acceptable through checking whether the angle between the normal directions at $\vec{p}_0$ and $\vec{p}_1$ is small enough, otherwise restart the prediction and the correction from $\vec{p}_0$ with a

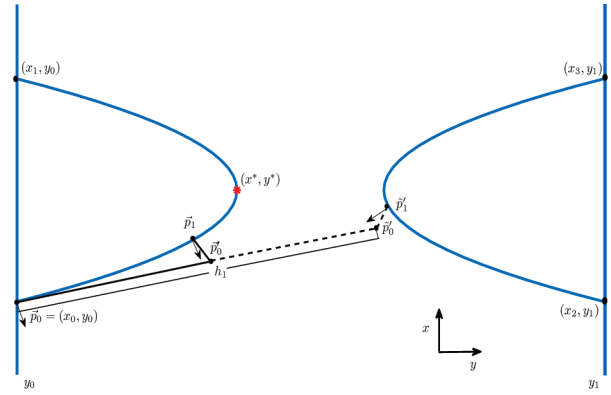

Figure 1. Schematic picture of Algorithm 1. Curved lines indicate the surface on the xy-plane. Arrows on the surface denote the normal directions of the Gaussian surface on the xy-plane pointing to the outside of the molecule. The $\vec{p}_0$ is the initial point, (x^*, y^*) is an extreme point along x-direction, and (x_1, y_0) is the final connected point on the surface obtained through the trace step from $\vec{p}_0$. As an illustration, $\tilde{x}_0'$ and $\tilde{x}_1$ on the dashed line that kinks near the curve on the right-hand side are the predicted and corrected points with a larger initial step h_1, which can be avoided through conditions ($\bigstar_1$) in algorithm 1. While $\vec{p}_0'$ is the predicted point along the tangent direction of $\vec{p}_0$ with a smaller step size $h_1/2$, and $\vec{p}_1$ is the corrected surface point from $\vec{p}_0'$.

Algorithm 1 Trace step

Input: Initial step size h_1 and initial surface point $\vec{p}_0$. The y-coordinates of two adjacent lines on xy-plane, y_0 and y_1. User defined small positive value ε and the bound for the cosine value δ ($0<\delta<1$).

Step 1, initialize $h_2 = h_1$ and $\vec{p}_0 = (x_0, y_0)$.

Step 2, let $\vec{p}_0' = \vec{p}_0 + h_2(-\phi_y(\vec{p}_0), \phi_x(\vec{p}_0))$.

Step 3, use Newton iterations to find t, s.t.

$$\phi(\vec{p}_0' + t(\phi_x(\vec{p}_0'), \phi_y(\vec{p}_0'))) = t_0.$$

Let $\vec{p}_1 = \vec{p}_0' + t(\phi_x(\vec{p}_0), \phi_y(\vec{p}_0))$.

Step 4, if $|\phi_x(\vec{p}_1)| < \varepsilon$, $\vec{p}_1$ is an extreme point along x direction, add it to the extreme point list.

Step 5, if $\cos((\phi_x(\vec{p}_1), \phi_y(\vec{p}_1)), (\phi_x(\vec{p}_0), \phi_y(\vec{p}_0))) < \delta$ (condition $\star_1$) or $(\phi_x(\vec{p}_1)\phi_x(\vec{p}_0) < 0$ and $\min(|\phi_x(\vec{p}_1)|, |\phi_x(\vec{p}_0)|) > \varepsilon)$ (condition $\star_2$), let $h_2 = h_2/2$ and go to step 2.

Step 6, if $(\vec{p}_0(y) - y_0)(\vec{p}_1(y) - y_0) < 0$ (or $(\vec{p}_0(y) - y_1)(\vec{p}_1(y) - y_1) < 0$) [a], interpolate $\vec{p}_0$ and $\vec{p}_1$ to get the connected point (x_1, y_0) (or (x_1, y_1)), let $\vec{p}_1 = (x_1, y_0)$ (or $\vec{p}_1 = (x_1, y_1)$) and stop.

Step 7, let $\vec{p}_0 = \vec{p}_1$ and go to step 1.

Output: The final connected surface point (x_1, y_0) on $y = y_0$ (or (x_1, y_1) on $y = y_1$), and the extreme point(s) if exist.

[a] Where $\vec{p}_0(y)$ and $\vec{p}_1(y)$ denote y-coordinates of point $\vec{p}_0$ and $\vec{p}_1$, respectively.

smaller step size; δ is a user-specified bound for cosine value of the angle. If the condition is not sufficient in some cases, then other conditions can be added, such as continuity of higher order derivatives. In the case that an extreme point along the x-direction exists between $\vec{p}_0$ and $\vec{p}_1$, condition ($\bigstar_2$) is used to detect it. In step 6, if the line segment connecting $\vec{p}_0$ and $\vec{p}_1$ crosses the line $y = y_0$ (or $y = y_1$), then the point of intersection is the final trace point. In step 7, $\vec{p}_0$ is replaced by $\vec{p}_1$ and starts tracing the next connected surface point from step 1. This process indicates that the final connected point can be located through trace step from initial point, therefore, the position of the final point needs not be known before the trace process. For this reason, a disjointed part of the whole surface will not be missed after the polygonization stage unless no points from the disjointed part are found in the stage of the surface point sampling.

Polygonization. This section is devoted to the polygonization step which connects the presampled points obtained through the process described in the section of computing surface points. Because solving $\phi(x, y, z) = t_0$ for different y, z values is equivalent to finding the intersection points of the surface and the different lines parallel to x-axis, polygonization of the whole surface can be achieved through connecting these points on every adjacent four lines. The problem is how to connect the surface points on the adjacent four lines. Suppose we have surface points set P consisting of four lists of points:

$$\{x_1^i, y_0, z_0\}, \quad i = 1, ..., n_1 \tag{7}$$

$$\{x_2^i, y_0 + \tilde{h}, z_0\}, \quad i = 1, ..., n_2 \tag{8}$$

$$\{x_3^i, y_0, z_0 + \tilde{h}\}, \quad i = 1, ..., n_3 \tag{9}$$

$$\{x_4^i, y_0 + \tilde{h}, z_0 + \tilde{h}\}, \quad i = 1, ..., n_4 \tag{10}$$

in the four adjacent lines:

$$\begin{cases} y = y_0 \\ z = z_0 \end{cases}, \begin{cases} y = y_0 + \tilde{h} \\ z = z_0 \end{cases}, \begin{cases} y = y_0 \\ z = z_0 + \tilde{h} \end{cases}, \begin{cases} y = y_0 + \tilde{h} \\ z = z_0 + \tilde{h} \end{cases} \tag{11}$$

Without loss of generality, assume $n_1 > 0$ and $(x_1^1, y_0, z_0) \triangleq \vec{p}_0$ is chosen to be the initial point. Through invoking the trace step described in former subsection, Algorithm 2 is to connect the points on the four adjacent lines to form small closed loops, i.e., polygons, on the surface.

Algorithm 2 is repeated until all points are connected, i.e., P is empty. This algorithm traces vertices of polygons in xy- and xz-planes alternately. The variable idx records the location of the last trace step, and the sig_x records the direction of the first trace step. If the traced vertex is not in the same xy- or xz-plane as $\vec{p}_0$, then the next trace direction will be reversed. After Algorithm 2 is finished, $\vec{p}_j$, $j = 0, ..., i - 1$, and the extreme points (if they exist) along the x-direction obtained during the trace steps are connected and form a polygon on the surface. Figure 2 shows some

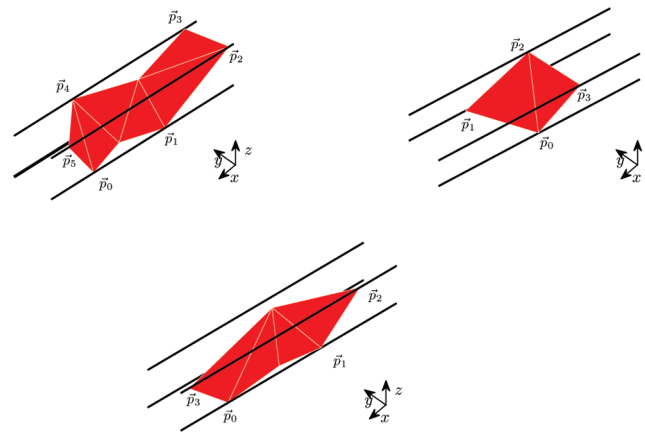

Figure 2. Some polygons with different numbers of vertices obtained from Algorithm 2. The unlabeled vertices are extreme points obtained from the trace steps.

Algorithm 2 Connecting the points on the four adjacent lines from the initial point $\vec{p}_0$ to form one polygon.

Input: The set P containing coordinates of all sampled points on the four adjacent lines. The grid space $\tilde{h}$. The yz-coordinates of lower-left line $\{y_0, z_0\}$.The initial point $\vec{p}_0$.

Find the connected point $\vec{p}_1$ on adjacent line using trace step on xy-plane from the initial point $\vec{p}_0$ along the tangent direction $(-sig_x\phi_y(\vec{p}_0), sig_x\phi_x(\vec{p}_0), 0)$, where sig_x is the sign of $\phi_x(\vec{p}_0)$.
Let $idx = 1$, $i = 1$ and $P = P - \{\vec{p}_0\}$.
while $\vec{p}_i \neq \vec{p}_0$ **do**
  if $\vec{p}_i$ is in set P **then**
    Let $P = P - \{\vec{p}_i\}$.
  end if
  if $idx = 1$ **then**
    if $\vec{p}_i(y) = y_0$ **then**
      Find the connected point $\vec{p}_{i+1}$ on adjacent line using trace step on xz-plane from $\vec{p}_i$ along the tangent direction $(-sig_x\phi_z(\vec{p}_i), 0, sig_x\phi_x(\vec{p}_i))$.
    else if $\vec{p}_i(y) = y_0 + \tilde{h}$ **then**
      Find the connected point $\vec{p}_{i+1}$ on adjacent line using trace step on xz-plane from $\vec{p}_i$ along the tangent direction $(sig_x\phi_z(\vec{p}_i), 0, -sig_x\phi_x(\vec{p}_i))$.
    end if
    Let $idx = 2$.
  else if $idx = 2$ **then**
    if $\vec{p}_i(3) = z_0$ **then**
      Find the connected point $\vec{p}_{i+1}$ on adjacent line using traces step on xy-plane from $\vec{p}_i$ along the tangent direction $(-sig_x\phi_y(\vec{p}_i), sig_x\phi_x(\vec{p}_i), 0)$.
    else if $\vec{p}_i(3) = z_0 + \tilde{h}$ **then**
      Find the connected point $\vec{p}_{i+1}$ on adjacent line using trace step on xy-plane from $\vec{p}_i$ along the tangent direction $(sig_x\phi_z(\vec{p}_i), -sig_x\phi_x(\vec{p}_i), 0)$.
    end if
    Let $idx = 1$.
  end if
  Let $i = i + 1$.
end while
Output: The polygon whose vertices are $\vec{p}_j, j = 1, \cdots, i$, and the extreme points (if they exist) along x-direction obtained during the trace steps.

examples of polygons with a different number of vertices obtained with Algorithm 2. The polygon can be simpler when the distances between the adjacent lines are shorter due to the smoothness of Gaussian surface. Based on the polygonized surface, the triangulation of the surface can be produced using standard polygon triangulation methods.[28]

Results

Performance. In this section, the performance of TMSmesh is compared with those of LSMS and MSMS. LSMS is a very fast program using a level-set method to present the surface based on cubic grids. MSMS is a typical and efficient software to triangulate the SES in the modeling area. A set of biomolecules with different sizes is chosen as a test benchmark. The meshing softwares are run on the molecular PQR files (PDB + atomic charges and radii information). For tests of the mesh tool and its applications in this work, we prepare a PQR benchmark (see Table 1) that can be found and is downloadable at our web page http://lsec.cc.ac.cn/~lubz/Meshing.html. It is worth making a note here about the vertex density used in TMSmesh and LSMS for comparison with MSMS surface density. For TMSmesh, grid spaces of 1.0 and 0.7 Å are chosen to approximate the molecular surface vertex densities 1/Å^2 and 2/Å^2, respectively. For LSMS, the current implementation works only on the following grid sizes: 16^3, 32^3, 64^3, 128^3, 256^3, and 512^3 (requiring a 4GB memory machine). Therefore, for each molecule, a proper grid size in LSMS is chosen to achieve the approximate density of 1/Å^2 or 2/Å^2, according to the maximum molecular length in *xyz* directions.

Table 2 shows the CPU time and memory use for these methods with 1 and 2 vertex/Å^2 mesh density. All computations run on Dell Precision T7500 with Intel(R) Xeon(R) CPU 3.3 GHz and 48GB memory under 64bit Linux system. As shown in Table 2, TMSmesh costs less memory than LSMS and MSMS but much more CPU time for small- or medium-sized molecules. The main cost of TMSmesh is in the polygonization stage that connects the presampled surface points on parallel lines through invoking the trace steps intensively. During each trace step, prediction and correction need to be performed several times with a small step size about 0.1 to 0.2 Å, i.e., there needs to be 5−10 prediction−correction steps within 1 Å distance on the surface to ensure the continuity of curves connecting vertices. However, LSMS directly searches and approximates the molecular surface based on cubic grid points using the level-set method. MSMS analytically computes molecular surface by first generating

Table 1. Description of Molecules in the PQR Benchmark

molecule (name or PDB code)	number of atoms	description
GLY	7	a single glycine residue
ADP	39	ADP molecule
2JM0	589	PDB code, chicken villin headpiece subdomain containing a fluorinated side chain in the core
FAS2	906	fasciculin2, a peptidic inhibitor of AChE
AChE monomer	8280	mouse acetylcholinesterase monomer
AChE tetramer	36 638	the structure of AChE tetramer, taken from ref 29
30S ribosome	88 431	30S ribosome, the PDB code is 1FJF
70S ribosome	165 337	obtained from 70S_ribosome3.7A_model140.pdb.gz on http://rna.ucsc.edu/rnacenter/ribosome_downloads.html
3K1Q	203 135	PDB code, a backbone model of an aquareovirus virion
2X9XX	510 727	a complex structure of the 70S ribosome bound to release factor 2 and a substrate analog, which has 4 split PDB entries: 2X9R, 2X9S, 2X9T, and 2X9U[30]
1K4R	1 082 160	PDB code, the envelope protein of the dengue virus[31]

Table 2. CPU Time and Memory Use for Molecular Surface Generation by TMSmesh, LSMS, and MSMS[a]

		CPU time (S)			memory use (MB)		
molecule	number of atoms	TMSmesh	LSMS[b]	MSMS	TMSmesh	LSMS	MSMS
FAS2	906	1.2	0.05	0.1	2	10	2
		1.8	0.1	0.1	4	19	2
AChE monomer	8280	12	0.1	0.6	11	21	21
		19	0.4	0.8	19	33	21
AChE tetramer	36 638	40	0.5	5.9	36	346	75
		62	3.6	7.1	54	257	79
30S ribosome	88 431	92	3.6	16.2	64	260	198
		151	28.1	19.1	100	2016	212
70S ribosome	165 337	180	3.8	46.2	127	262	469
		283	28.6	Fail	185	2100	–
3K1Q	203 135	226	4.0	51.5	131	262	383
		359	28.9	55.1	192	2100	410
2X9XX	510 727	577	30.5	Fail	271	2100	–
		910	Fail	Fail	410	–	–
1K4R	1 082 160	1260	30.5	Fail	630	2168	–
		2080	Fail	Fail	890	–	–

[a] The data in the first and second row for each molecule are corresponding to the density 1 vertex/Å^2 and 2 vertex/Å^2, respectively. [b] The failed cases in this column require a grid size of 1024^3 or larger, which is not supported by LSMS.

the so-called reduced surface that is obtained directly from atomic geometry information. Both LSMS and MSMS avoid the time-consuming step, polygonization. This makes LSMS and MSMS cost less CPU time than that of TMSmesh. Nevertheless, either the surface topology or the smoothness may not be guaranteed in LSMS and MSMS. TMSmesh is expected to be speeded up using an adaptive box structure, parallel computing, and more sophisticated polygonization algorithm. For LSMS, the cost is proportional to L^3, where L is the number of grids in one dimension. Therefore, the memory requirement and the CPU time increase dramatically when L becomes large. For MSMS, the computational complexity is $O[N\,log(N)]$, where N is the number of atoms, but the singularity of the molecular surface may cause numerical instability and produce incorrect results. In TMSmesh, the number of cubes is proportional to the number of atoms, since the edge length of cube is fixed to be 4 Å in our work. In addition, the calculations are done locally, and no global information is needed during the process of estimating the bounds of $\phi(x)$ in each cube, computing surface points, and tracing of the left cubes intersecting the surface. The reason is that calculating the values of $\phi(x)$ and its gradients only need to sum Gaussian kernels for near

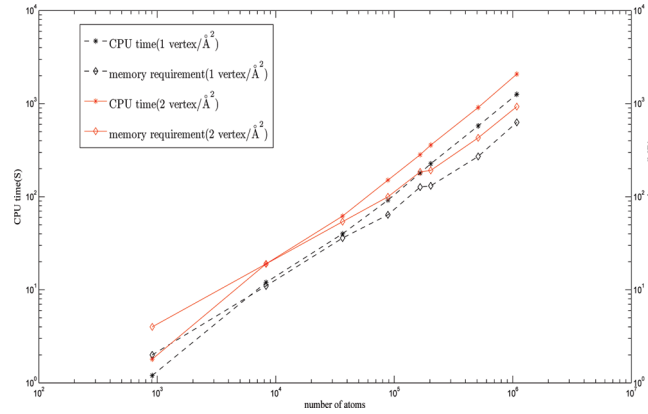

Figure 3. Computational performance of TMSmesh.

atoms, as the Gaussian kernel $e^{d(\|\vec{x}-\vec{c}_i\|^2/r_i^{2-1})}$ decreases to 0 faster when $\|\vec{x}-\vec{c}_i\|$ is large. As shown in Table 2 and Figure 3, the complexity of TMSmesh is $O(N)$. Compared to LSMS and MSMS, TMSmesh produces triangulations of a smooth surface, and it can be successfully applied to a biomolecule consisting of more than one million atoms, such as the dengue virus as shown in Figure 4. Because the virus structure is among the largest ones in the Protein Data Bank

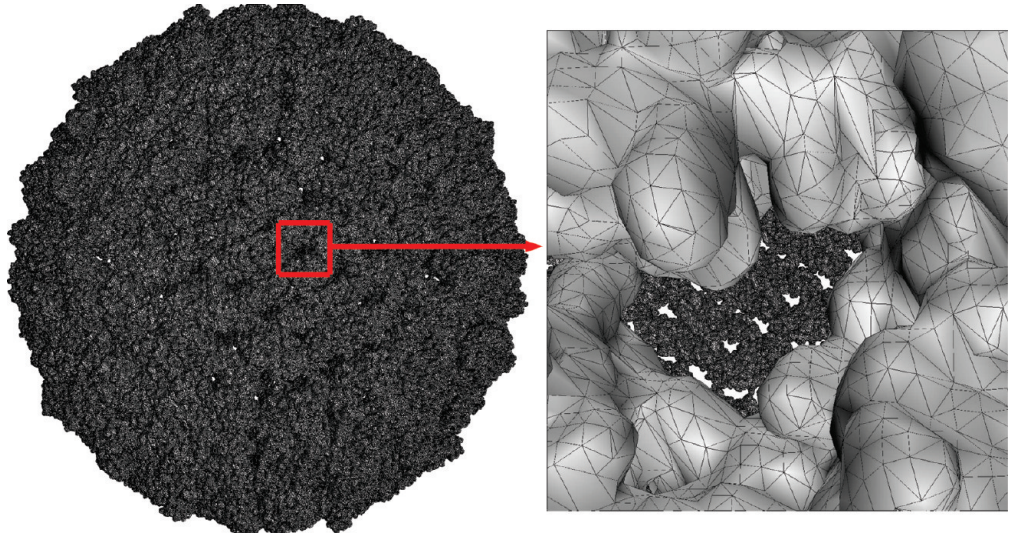

Figure 4. Surface triangular mesh of the envelope protein of the dengue virus (PDB code 1K4R).[31] The left-hand side is the whole surface mesh, and the right-hand side is a close view of a selected part with a gap on the surface (surrounded by the box). Because the structure is a shell, the inner surface of the other side of the shell is also shown through the gap. The mesh density is 1 vertex/Å^2.

(PDB), together with consideration of good algorithm stability, TMSmesh can be expected to be capable of handling the arbitrary size of molecules available in PDB.

Because MSMS is a widely used tool for surface meshing in molecular modeling, we compare the qualities, in particular uniformness, of triangles produced by TMSmesh and MSMS. The distributions of the ratios of the shortest to longest edge lengths of each triangle are used to describe the uniformness of meshes. A ratio of 1.0 corresponds to an equilateral triangle, and a ratio close to 0 indicates poor uniformness of the triangle. In other words, the higher the ratio, the better the quality of the triangle. The ratio distributions of meshes for a large molecule, AChE tetramer, and a relatively small one, FAS2, are shown in Figure 5. TMSmesh and MSMS meshes with two densities 2 and 10 vertex/Å^2 are compared. It is shown that at both low and high densities, the ratios of TMSmesh meshes are clustered around 0.75. Comparatively, at a low density of 2 vertex/Å^2, the ratios of MSMS meshes distribute more evenly in [0,1] (see the first and third rows of Figure 5). At a high density of 10 vertex/Å^2 (see the second and the bottom rows of Figure 5), the distributions of MSMS mesh ratios are improved and clustered around 0.65 that still indicates less uniform triangle clusters than TMSmesh results; furthermore, at the region with a ratio close to 0, TMSmesh also generates less percentage of such poor quality triangles than MSMS. In addition, successful applications (see the following subsection) of all of our meshes to boundary element simulations also indicate improvement in mesh quality relative to MSMS mesh in the sense of right topology (e.g., without single-element-connected edges, isolated points), uniformness, and smoothness.

Finally, it is worth making a note of the molecular cavity as explored by many other tools. TMSmesh does not differentiate the outer and interior surfaces of cavities in the meshing process. The cavities can be located through the connectivities of the triangle elements, because an internal cavity is a disjointed component of Gaussian surface (eq 1), and its normal directions $\nabla\phi(x)$ are inward. The same method of finding internal cavities is used in GRASP.[12]

Application to Boundary Element Simulation of Electrostatics. Similar to other surface generation softwares, such as MSMS and LSMS, the surface mesh generated by TMSmesh preserves molecular surface features and thus can be applied to molecular visualization and analysis of surface area, topology, and volume in computational structure biology and structural bioinformatics. Furthermore, the goal of this work is to extend applications to some advanced mathematical modeling of biomolecules, which places demands upon the quality and the rigorous topology of the meshes.

In this work, we test the meshes in the usage of a boundary element method to calculate the Poisson−Boltzmann electrostatics. The BEM software used is a publicly available PB solver AFMPB.[32] MSMS meshes have already been used in many previous boundary element PB works for smaller molecules and have demonstrated to generate reasonable results. Because LSMS mesh is built on cubic grid that can deviate somehow from the curved molecular surface (unless the grid space is small enough), we did not perform AFMPB computation with LSMS mesh. Instead, the AFMPB results from meshes of TMSmesh and MSMS are compared. Our test cases, up to molecular size of 2X9XX (which is a complex structure of the 70S ribosome bound to release factor 2 and a substrate analog)[30] due to the memory limit of our machine, show that AFMPB can go through and produce converged results with all of our meshes for the molecules described in Table 2. However, the solver fails for meshes directly obtained with MSMS for 30S ribosome and larger molecules in Table 2, which is due to singularities or incorrect topologies in MSMS meshes. Figure 6 shows the solvation energies by AFMPB as well as the surface areas and molecular volumes computed from the meshes of three small molecules, GLY, ADP, and 2JM0 (see Table 1) using different mesh densities. The figure indicates that BEM

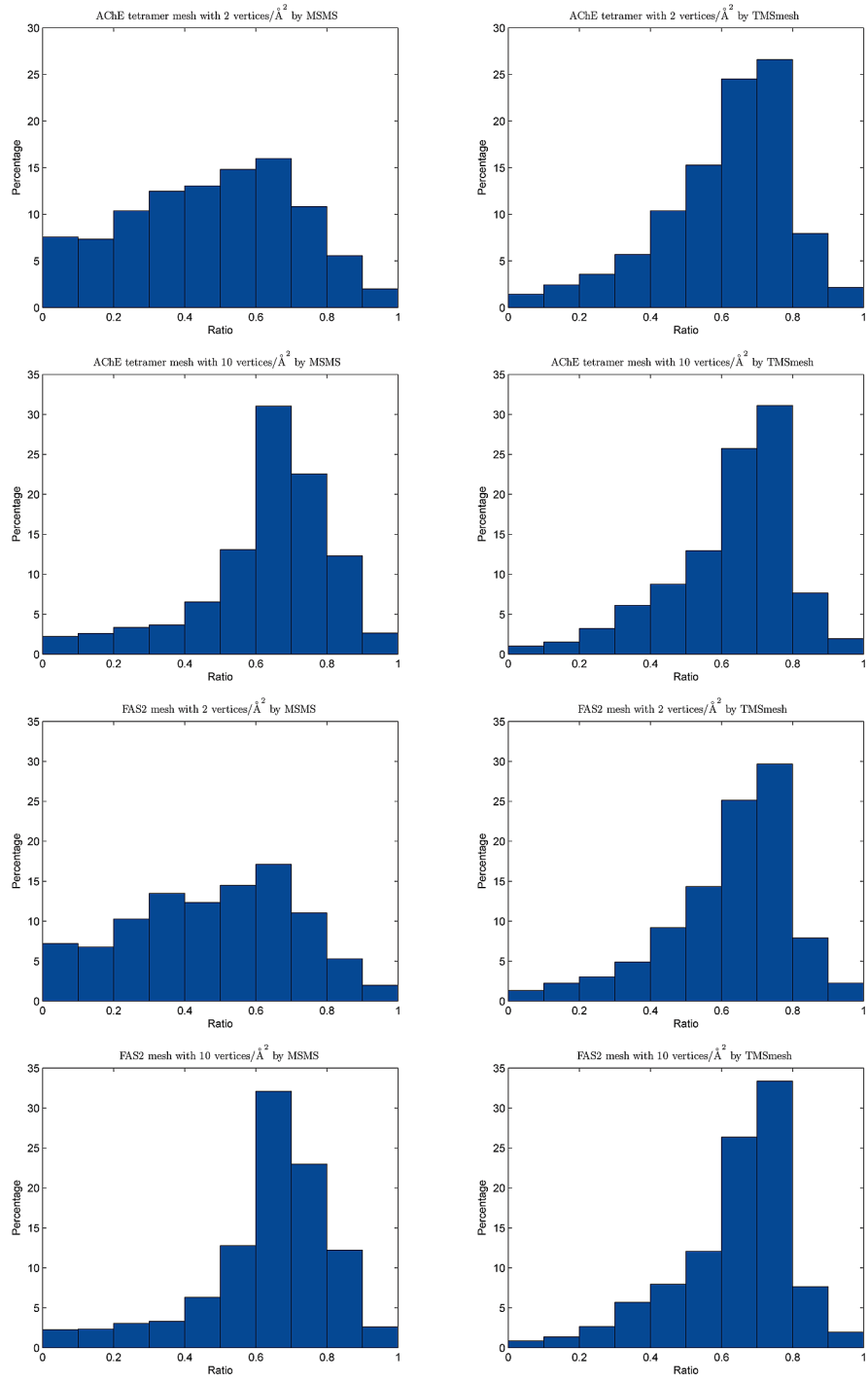

Figure 5. Distributions of ratio of the shortest edge length to the longest edge length of each triangle produced by MSMS (left column) and TMSmesh (right column). The first two rows are for AChE tetramer meshes with densities of 2 and 10 vertex/Å^2, respectively. The last two rows are for FAS2 meshes with densities of 2 and 10 vertex/Å^2, respectively. A ratio of 1.0 corresponds to an equilateral triangle.

calculations work for all the meshes with different densities produced by TMSmesh, and the mesh leads to convergent and reasonable results for energy, area, and volume when the mesh resolution is increased. This is reasonable because the mesh converges to the implicitly defined Gaussian surface with the increasing of the resolution. Figure 6 also shows that the results computed by TMSmesh converge more smoothly than those of MSMS as the number of triangles increased. There are some discrepancies between the quantities calculated with TMSmesh and MSMS meshes, which is due to the different definitions of molecular surface used in the mesh tools.

It is known that BEM is a memory-saving approach to solve the PBE for macromolecules, which indicates that a combination of TMSmesh and a BEM solver, such as AFMPB, is a promising way to handle large-scale molecular systems for electrostatic calculations. As a representative molecular system, we choose the complex structure 2X9XX (see Table 1), which contains 510 727 atoms with a dimension of 270 × 392 × 384 Å (Figure 7). The molecular surface

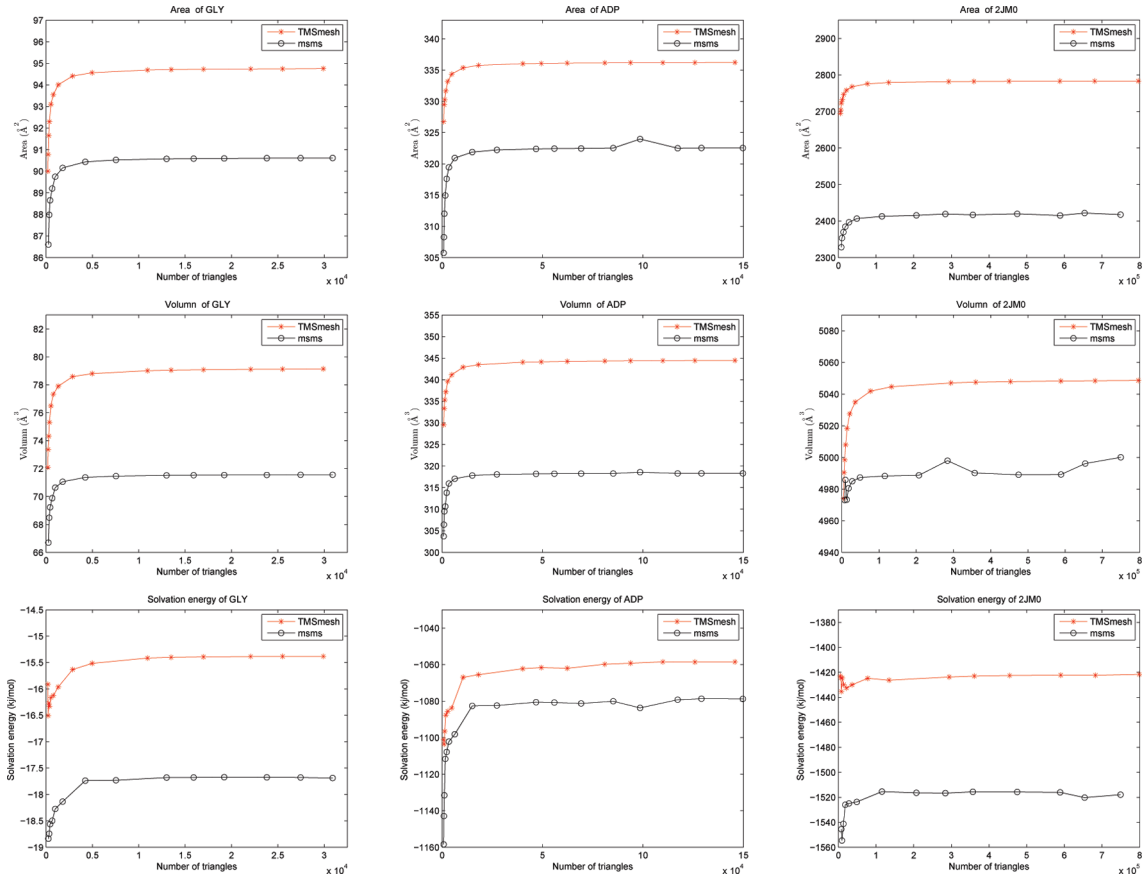

Figure 6. Area (first row), volume (second row), and solvation energy (third row) for GLY (left column), ADP (middle column), and 2JM0 (right column).

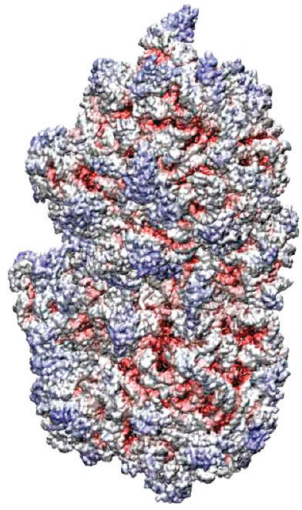

Figure 7. Electrostatic potential surface of the complex structure 2X9XX calculated with AFMPB[32] using a surface mesh density 1 vertex/Å^2.

discretization using TMSmesh with a mesh density of 1 vertex/Å^2 results in 3 693 500 triangular elements and 1 841 858 nodes. Figure 7 shows the computed electrostatic potentials mapped on the molecular surface.

Conclusion

We have described a new method for molecular surface meshing by a tracing surface technique. The implemented software TMSmesh is shown as a robust tool for meshing molecular Gaussian surfaces in the sense that: (1) It can stably handle arbitrary sizes of molecules available in PDB on a typical desktop or laptop machine, even for the not "good" molecular structures (such as ones with strong atomic clash) and (2) the generated mesh has good quality (smoothness, uniformness, and topological correctness). The mesh converges to the smooth Gaussian surface when the mesh resolution increased and from which all the calculations of electrostatic solvation energy, surface area, and molecular volume show good convergence performance and reasonable results. Therefore, in addition to usual applications of molecular visualization and geometry analysis, the mesh is also shown to be applicable for numerical simulations with boundary element methods. Specifically, a combination of TMSmesh and BEM solver opens a possible route to simulate electrostatic solvation of large-scale molecular systems on a desktop computer.

In order to simulate more complicated and wider ranges of biophysical processes using a variety of numerical techniques and modeling approaches, the current meshing method needs further improvements. First, efficiency seems to be the current bottleneck in some possible applications where the mesh needs to be either generated for large systems or generated frequently, such as in multiple-conformational analysis, BEM-based implicit solvent MD simulations[33,34] (whereas in some finite difference-based MD simulations,[35] surface meshing is not required), or elastic modeling of

conformational changes. Second, surface mesh smoothness needs to be further improved. Third, molecular volume mesh generation based on surface mesh is required for some finite element types of simulations. In addition, in the case of higher order numerical computations, the Gaussian surface may need to be discretized more accurately using curved elements. It would be convenient to generate curved elements by small modification of TMSmesh, because the method produces an abundance of trace information in the meshing procedure.

Finally, it is worth a mention regarding PB calculations. It is hard to conclude so far which surface specification is the best for biophysical studies due to being complicated by some other factors (like the atomic radii) in the setup of a PB calculation that can also affect the final results. Likewise, the Gaussian surface model and the meshing approach adopted in this work for PB electrostatic calculations will need further systematic studies and comparisons with experiments or other computational methods.

Acknowledgment. We thank the reviewers for their comments and suggestions, which are very helpful for improving the paper. M.X. was supported in part by the China NSF (NSFC20872107). B.Z. was supported by the Chinese Academy of Sciences, the State Key Laboratory of Scientific/Engineering Computing, and the China NSF (NSFC10971218).

Note Added after ASAP Publication. This article was published on the web on November 30, 2010. Algorithm 1 and Algorithm 2 have been revised to remove the color. The correct version was published on December 3, 2010.

References

(1) Lu, B. Z.; Zhou, Y. C.; Holst, M. J.; McCammon, J. A. *Commun. Comput. Phys.* **2008**, *3*, 973–1009.

(2) Edelsbrunner, H. *Discrete Comput. Geom.* **1999**, *21*, 87–115.

(3) Bates, P. W.; Wei, G. W.; Zhao, S. *J. Comput. Chem.* **2008**, *29*, 380–391.

(4) Duncan, B. S.; Olson, A. J. *Biopolymers* **1993**, *33*, 231–238.

(5) Blinn, J. F. *ACM Trans. Graph.* **1982**, *1*, 235–256.

(6) Mcgann, M. R.; Almond, H. R.; Nicholls, A.; Grant, A. J.; Brown, F. K. *Biopolymers* **2003**, *68*, 76–90.

(7) Grant, J. A.; Gallardo, M. A.; Pickup, B. T. *J. Comput. Chem.* **1996**, *17*, 1653–1666.

(8) J.Weiser; Shenkin, P.; Still, W. *J. Comput. Chem.* **1999**, *20*, 688–703.

(9) Yun, Z.; Matthew, P.; Richard, A. *J. Comput. Chem.* **2005**, *27*, 72–89.

(10) Connolly, M. L. *J. Appl. Crystallogr.* **1983**, *16*, 548–558.

(11) Connolly, M. L. *Science* **1983**, *221*, 709–713.

(12) Nicholls, A.; Bharadwaj, R.; Honig, B. *Biophys. J.* **1995**, *64*, 166–167.

(13) Vorobjev, Y. N.; Hermans, J. *Biophys. J.* **1997**, 722–732.

(14) Edelsbrunner, H.; Mücke, E. P. *ACM Trans. Graph.* **1994**, *13*, 43–72.

(15) Sanner, M.; Olson, A.; Spehner, J. *Biopolymers* **1996**, *38*, 305–320.

(16) Ryu, J.; Park, R.; Kim, D.-S. *Comput. Aided Des.* **2007**, *39*, 1042–1057.

(17) Zhang, Y.; Ch, G. X.; Bajaj, R. *Comput. Aided Geomet. Des.* **2006**, *23*, 510–530.

(18) Yu, Z.; Holst, M. J.; Andrew McCammon, J. *Finite Elem. Anal. Des.* **2008**, *44*, 715–723.

(19) Can, T.; Chen, C.-I.; Wang, Y.-F. *J. Mol. Graphics Modell.* **2006**, *25*, 442–454.

(20) Chavent, M.; Levy, B.; Maigret, B. *J. Mol. Graphics Modell.* **2008**, *27*, 209–216.

(21) Cheng, H.; Shi, X. *Comput. Geom.* **2009**, *42*, 196–206.

(22) Lorensen, W.; Cline, H. E. *Comput. Graph.* **1987**, *21*, 163–169.

(23) Ju, T.; Losasso, F.; Schaefer, S.; Warren, J. D. *ACM Trans. Graph.* **2002**, *21*, 339−346.

(24) Hilton, A.; Stoddart, A. J.; Illingworth, J.; Windeatt, T. *IEEE Int. Conf. Image Process.* **1996**, 381–384.

(25) Hartmann, E. *Vis. Comput.* **1998**, 95–108.

(26) Karkanis, T.; Stewart, J. *IEEE Comput. Graphics Appl.* **2001**, 60–69.

(27) Jenkins, M.; Traub, J. F. *Numer. Math.* **1970**, 253–263.

(28) de Berg, M.; van Kreveld, M.; Overmars, M.; Schwarzkopf, O. *Computational Geometry: Algorithms and Applications*, 2nd ed.; Springer-Verlag: New York, 2000; pp 45−61.

(29) Zhang, D.; McCammon, J. A. *PLoS Comput. Biol.* **2005**, 484–491.

(30) Jin, H.; Kelley, A. C.; Loakes, D.; Ramakrishnan, V. *Proc. Natl. Acad. Sci. U.S.A.* **2010**, 8593–8598.

(31) Kuhn, R. J.; Zhang, W.; Rossmann, M. G.; Sergei V. Pletnev, J. C.; Lenches, E.; Jones, C. T.; Mukhopadhyay, S.; Paul R. Chipman, E. G. S.; Baker, T. S.; Strauss, J. H. *Cell* **2002**, 715–725.

(32) Lu, B. Z.; Cheng, X. L.; Huang, J. F.; McCammon, J. A. *Comput. Phys. Commun.* **2010**, 1150–1160.

(33) Wang, C. X.; Wan, S. Z.; Xiang, Z. X.; Shi, Y. Y. *J. Phys. Chem. B* **1997**, *101*, 230–235.

(34) Lu, B. Z.; Wang, C. X.; Chen, W. Z.; Wan, S. Z.; Shi, Y. Y. *J. Phys. Chem. B* **2000**, *104*, 6877–6883.

(35) Wang, J.; Tan, C. H.; Tan, Y. H.; Lu, Q.; Luo, R. *Commun. Comput. Phys.* **2008**, *3*, 1010–1031.

CT100376G