FEATURE-PRESERVING ADAPTIVE MESH GENERATION FOR MOLECULAR MODELING

ZEYUN YU[§], MICHAEL HOLST[§], AND ANDREW MCCAMMON[†]

[§]*Department of Mathematics,* [†]*Department of Chemistry and Biochemistry*
University of California, San Diego
9500 Gilman Drive, Dept. 0112, La Jolla, CA 92093-0112 USA
Email: zeyu@math.ucsd.edu*,* mholst@math.ucsd.edu*,* jmccammon@ucsd.edu

Abstract. In this article we describe a mesh generation toolchain for molecular modeling. We take as inputs the centers and radii of all atoms of a molecule and the toolchain outputs both surface (triangular) and volumetric (tetrahedral) meshes. Experiments on a number of molecules are demonstrated, showing that our toolchain possesses a number of desirable properties: feature-preservation, local adaptivity, high quality (in various metrics), and smoothness (for surface meshes). Our tool is also applicable to other types of inputs such as scalar 3D volumes and triangular surface meshes with low quality, and hence can be used for generation/improvment of meshes in a broad range of applications.

Contents

1. Introduction

Mesh generation has been demonstrated to be very useful in the finite element method and has been extensively studied in both applied mathematics [6] and computational engineering [2, 26]. Although different types of meshes may be generated depending on the numerical solvers to be employed, we restrict ourselves in this paper to triangular (surface) and tetrahedral (volumetric) meshes. In particular, we consider mesh generation for molecular applications, namely, meshes that are generated from a set of centers and radii of atoms in a molecule and are then used for solving various types of partial differential equations arising in molecular modeling.

Molecular mesh generation requires good approximation of molecular surfaces. There are two primary ways to construct such surfaces: one is based on the "hard sphere" model [35] and the other is based on the level set of a "soft" Gaussian function [22]. In the first model, a molecule is treated as a list of "hard" spheres with different radii, from which three types of surfaces can be extracted. The *van*

Date: July 10, 2007.

der Waals surface (or *envelope*) is defined as the union of the spheres with their intersecting portions removed. An alternative way to define the molecular surface is by rolling a sphere over the van der Waals surface, where the loci of the probing sphere gives rise to the so-called *solvent accessible surface* (SAS) [29]. The radius of the probe sphere is chosen as the size of the solvent molecules (usually water). Another widely used surface is known as *solvent excluded surface* (SES), which is defined as the "inward-facing" part of the probing sphere as it rolls over the molecules [14, 22, 35]. The molecular surface can be represented analytically by a list of seamless spherical patches [14, 42] and triangular meshes can be generated using such tools as *MSMS* [37].

In contrast, the "soft" model treats each atom as a Gaussian-like smoothly decaying scalar function in $\Re^3$ that approximates the electron density of the atom [10, 17, 22]. The molecular surfaces, including the van der Waals surface, SAS and SES, are then approximated by appropriate level sets (or isocontours) of the Gaussian function [10, 17]. In addition to the high smoothness, the Gaussian model provides an analytical, and more realistic, representation to the electron density of a molecule [17]. The isocontouring and triangulation of a 3D scalar function are well studied in computer graphics, and two primary techniques for extracting isocontours are the marching cube [30] and the dual contouring method [28].

The goal of the present paper is to describe a toolchain designed to generate high quality tetrahedral meshes for molecular simulations. In general, a good mesh should have the following properties: (1) feature-preserving, (2) adaptivity, and (3) high quality. *Tetgen* is an example of an outstanding software tool that can generate high quality tetrahedral meshes from a triangulated surface, using constrained Delaunay tetrahedralizations [38, 39]. It can be used together with the surface triangulation tools such as *MSMS* [37]. But the major weakness of this tool is that the input surface mesh must be "good enough", meaning that angles such as those close to $0°$ or $180°$ are not allowed. Otherwise, *Tetgen* must insert a large number of additional nodes or the algorithms fail. Unfortunately many surface triangulation tools such as *MSMS* [37] often generate meshes with very poor quality in standard metrics in spite of their graphically smooth surfaces. These types of surface meshes usually do not work well with tools like *Tetgen* without significant mesh pre-processing. To this end, improving surface mesh quality will be our main goal in the present paper.

While *Tetgen* cannot be used alone to generate tetrahedral meshes from a set of atoms, another excellent tool called *LBIE* integrates the surface triangulation, tetrahedron generation and smoothing [50] into a stand-alone software package, reading PDB files or 3D volumes as its inputs. *LBIE* is based on the "soft" model, meaning that it extracts the molecular surfaces using the level sets (or isocontours) of the Gaussian kernel summations of all atoms in a molecule. An octree-based data structure is employed to construct both the surface and volumetric meshes that preserve important geometric features and guarantee adaptive triangles (tetrahedra) as well. However, the drawbacks of *LBIE* are as follows. (1) The meshes can only be constructed from a 3D scalar volume. If the input is a surface mesh, the mesh must be converted into a volume by, for example, the signed distance transformation [50] before it can be used by *LBIE*. In addition, each dimension of the volumes generated or given must be in a power of 2, due to the octree data structure used in this tool. In the worst case where a volume is slightly bigger than 2^n in each dimension, a

2^{n+1} volume must be generated by zero-padding the original volume, which results in lower efficiency in computational time and memory. (2) Although the qualities of the interior tetrahedral meshes generated by *LBIE* are usually good, there are many "sharp" triangles on the surfaces due to the contouring method used in *LBIE*, and hence many tetrahedra near the surfaces can be poorly shaped. A new version of *LBIE* has been introduced in [51] by applying surface and tetrahedral mesh smoothing techniques (such as edge contraction and weighted averaging). While the surface mesh smoothing works well, the tetrahedral mesh smoothing turns out to be slow and sometimes the algorithm fails.

In the present paper we introduce a new mesh generation tool that aims to produce feature-preserving, adaptive, and high quality surface (triangular) and volumetric (tetrahedral) meshes for molecular modeling and other applications. Like *LBIE* [50], our method is based on the level set of a Gaussian kernel function that approximates the molecular surface of the given molecule. The surface meshes extracted by the iso-contouring techniques always contain many "skinny" triangles that may cause poor approximation quality in finite element methods. To this end, we develop a variant of the angle-based approach [52] to improve the mesh quality while retaining the smoothness of the iso-contours. In our toolchain we also develop an adaptive mesh coarsening technique, by which the resulting meshes are made finer in regions of high curvatures or by some other user-specified criteria and coarser elsewhere. Along with the mesh coarsening, a normal-based surface mesh smoothing technique is employed to guarantee the smoothness of the meshes while they are being coarsened. Once a good surface mesh is generated, the tetrahedra (interior and exterior of the molecular surfaces) are constructed using the constrained Delaunay triangulation as implemented in *Tetgen* [38, 39].

The remainder of this paper is organized as follows. In Section 2 we present the algorithmic details of each step of our mesh generation toolchain. In Section 3 we show the mesh generation results on a number of experimental examples and analyze the mesh qualities with and without our mesh processing tools. We then discuss several interesting issues in Section 4, followed by our conclusion of this paper in Section 5.

2. Methods

In this section we describe the algorithms that we use to construct the triangular surface and tetrahedral volumetric meshes. For the particular purpose of molecular modeling, the inputs to our toolchain are a list of centers and radii of atoms (e.g., PQR files [16] or PDB files [8] with radii defined by users), although our tool can also read as inputs an arbitrary 3D scalar volume or a triangulated surface mesh with very low quality. Fig. 1 shows the pipeline of our mesh generation toolchain. In the following subsections, we shall explain each step in detail.

2.1. **Initial Surface Meshing.**

In our mesh generation toolchain, the initial surface meshes are defined by the level sets of the Gaussian kernel function that is computed from a list of centers and radii of the atoms as follows:

$$F(\mathbf{x}) = \sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{e}^{\mathbf{B_i}\left(\frac{\|\mathbf{x}-\mathbf{x_i}\|^2}{\mathbf{r_i^2}}-\mathbf{1}\right)}, \tag{1}$$

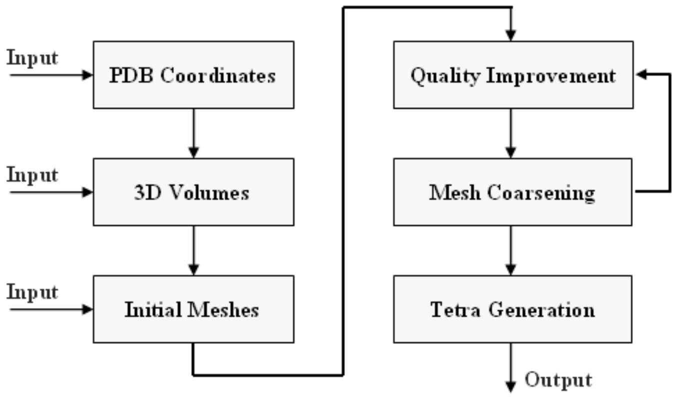


FIGURE 1. Illustration of our mesh generation toolchain.

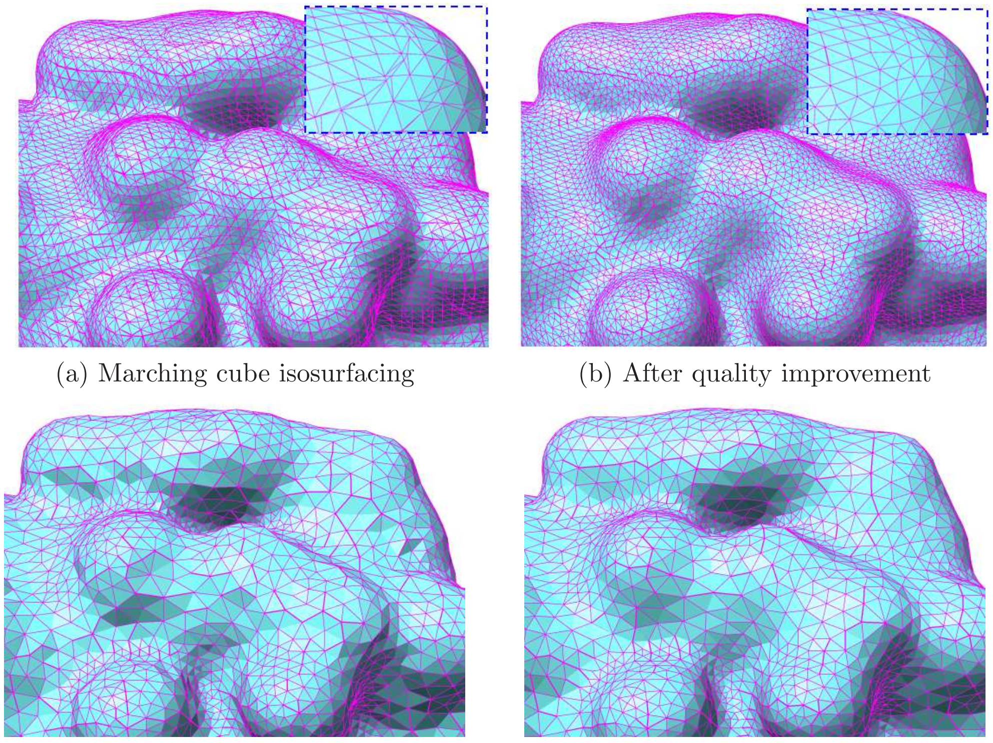
where the negative parameter B_i is called the *blobbyness* that controls the spread of electron density of each atom. We usually treat the blobbyness as a constant parameter (denoted by B_0) for all atoms. When B_0 goes to zero, $F(\mathbf{x})$ becomes constant and all features disappear. On the other hand, a blobbyness of -2.5 is said to be small enough for producing a volume at atomic resolutions [36]. Our experiments on a number of molecules show that the blobbyness at -0.5 produces a good resolution for molecular simulations.

An alternate way to define the Gaussian kernel function is by putting the blobbyness and radius of each atom together as a constant, as formulated in the following equation [51]:

$$G(\mathbf{x}) = \sum_{i=1}^{N} e^{C_0(\|\mathbf{x}-\mathbf{x}_i\|^2 - r_i^2)}, \tag{2}$$

where $C_0 = \frac{B_i}{r_i^2}$ is a constant for all atoms $i = 1, \cdots, N$. From a computational point of view, the formulation in Equation 2 might be more beneficial due to the constant decay rate C_0 that may lead to a fast Gaussian summation using the recursive scheme as described in [46]. We shall discuss this issue in detail in Section 4.

Once the Gaussian kernel function is computed and summed up for all atoms, the molecular surface is then defined by the level set as $F(\mathbf{x}) = t_0$, where t_0 is the isovalue and usually set as 1.0 as pointed out in several previous articles [10, 22, 51]. The surface (triangular) mesh can be constructed by either one of the two primary isocontouring techniques − the marching cube method [30] and dual contouring method [28]. In our mesh generation toolchain we employ an implementation of the marching cube method. Fig. 2(a) shows an example of the isosurface extracted using this method. To illustrate the details better, shown here is only a small portion of the molecular surface (PDB-ID: 1CID). More results on this molecule will be given in Section 3. From this example, we can see that: (1) the isocontouring technique can extract very smooth surfaces, but (2) many triangles are extremely “sharp” and could lead to poor approximation quality in finite element analysis.



(a) Marching cube isosurfacing (b) After quality improvement

(c) After coarsening (d) After surface smoothing

FIGURE 2. Illustration of the surface generation and post-processing. The PDB used is ICID with 1381 atoms. (a) The 3D volume is generated using the Gaussian blurring approach (Equation 1). The blobbyness is set as -0.5. The dimension of the volume obtained is $130 \times 132 \times 176$. Shown here is part of the molecular surface triangulation using the marching cube method at the isovalue of 1.0. There are $62,454$ nodes and $124,904$ triangles in total. The minimal and maximal angles are $0.02°$ and $179.10°$ respectively. (b) The surface mesh after two iterations of running the quality improvement algorithm. The minimal and maximal angles improved become $14.11°$ and $135.65°$ respectively. The small windows shown in (a) and (b) are a closer look at the right-top corner of the mesh. (c) After coarsening, there are $8,846$ nodes and $17,688$ triangles left. The minimal and maximal angles are $0.42°$ and $177.85°$ before quality improvement and $20.22°$ and $130.71°$ after quality improvement (8 iterations). (d) After normal-based surface smoothing (2 iterations).

Therefore, we need to improve the mesh quality (i.e., remove the "bad" triangles) while preserving the smoothness of the initial triangular mesh.

2.2. **Surface Mesh Improvement.**

Mesh quality in 2D planes or 3D volumes is usually measured by the ratio of the radii of the circumscribed to inscribed circles in each triangle. There are a number of other quality metrics such as those based on the edges, angles, matrix norms, or a combination of these quantities, but they are more or less equivalent to each other [33]. The mesh quality can be improved by three major techniques: inserting/deleting vertices, swapping edges/faces, and moving the vertices without changing the mesh topology [21]. The last one is the main strategy we use to improve the mesh quality in our toolchain.

For surface meshes, however, moving the vertices may change the shape of the surface. Therefore, besides the quality measure we mentioned above, several other criteria must be taken into consideration in particular for surface mesh improvements: (1) Important features (e.g., sharp boundaries, concavities, holes, etc.) on the original surface should be preserved as much as possible and (2) the surface should be kept smooth while the vertices are moved. In the following we shall first describe the local structure tensor that is used to quantify surface features. We then present an angle-based method for mesh quality improvement.

2.2.1. *Local Structure Tensor.* Local structure tensor has been used for anisotropic image smoothing [18, 43] and protein secondary structure detection [48, 49]. The idea of the local structure tensor is derived from the well-known principal component analysis (PCA) technique [27]. It basically captures the principal orientations of a set of vectors in space. For volumetric data, the local structure tensor is constructed from the gradient vectors [18, 49]. For triangulated surface meshes, we compute the local structure tensor based on a set of normal vectors in a neighborhood of a vertex as follows:

$$T(\mathbf{v}) = \sum_{i=1}^{N} \begin{pmatrix} \mathbf{n}_x^{(i)}\mathbf{n}_x^{(i)} & \mathbf{n}_x^{(i)}\mathbf{n}_y^{(i)} & \mathbf{n}_x^{(i)}\mathbf{n}_z^{(i)} \\ \mathbf{n}_y^{(i)}\mathbf{n}_x^{(i)} & \mathbf{n}_y^{(i)}\mathbf{n}_y^{(i)} & \mathbf{n}_y^{(i)}\mathbf{n}_z^{(i)} \\ \mathbf{n}_z^{(i)}\mathbf{n}_x^{(i)} & \mathbf{n}_z^{(i)}\mathbf{n}_y^{(i)} & \mathbf{n}_z^{(i)}\mathbf{n}_z^{(i)} \end{pmatrix}, \tag{3}$$

where $(\mathbf{n}_x^{(i)}, \mathbf{n}_y^{(i)}, \mathbf{n}_z^{(i)})$ is the normal of the i^{th} neighbor of a vertex $\mathbf{v}$ and N is the total number of neighbors. The normal of a vertex is defined by the average of the normals of all its incident triangles. From this definition, we can see that the order of the neighbors can be arbitrary because the local structure tensor is simply the sum of the matrices corresponding to all the neighbors. In addition, the directions of the normal vectors do not affect the local structure tensor. It is worthwhile pointing out that the neighbors considered may be more than the incident ones of a vertex. They could extend by several “layers” and the number of the “layers” considered depends on how large the local features are. We usually consider 2 to 3 “layers” around a vertex, to compute the local structure tensor in our molecular mesh generator.

As illustrated in [18, 49], the eigenvalues and eigenvectors of the structure tensor indicate the overall distribution of the vectors within a local window and categorize the features locally into different types: spheres, lines, and planes. In case of surface meshes, the type of features are slightly different. Let the eigenvalues be $\lambda_1, \lambda_2, \lambda_3$ and $\lambda_1 \geq \lambda_2 \geq \lambda_3$. Then we have the following cases:

- Spheres and saddles: $\lambda_1 \approx \lambda_2 \approx \lambda_3 > 0$.
- Ridges and valleys: $\lambda_1 \approx \lambda_2 >> \lambda_3 \approx 0$.
- Planes: $\lambda_1 >> \lambda_2 \approx \lambda_3 \approx 0$.

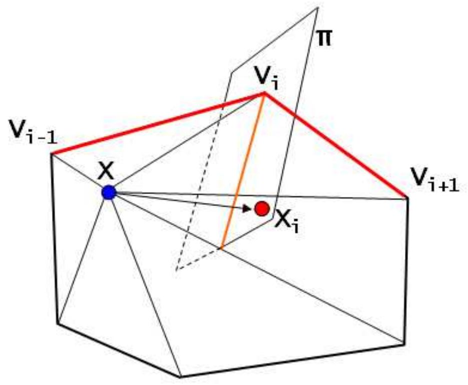


FIGURE 3. Illustration of the angle-based quality improvement algorithm.

2.2.2. *Angle-based Mesh Quality Improvement.* The quality of a mesh can be improved by maximizing the minimal angles. As illustrated in Fig. 3, let X be the vertex that we want to move for quality improvement. The (incident) neighbors are denoted by $V_i, i = 1, \cdots, n$, where n is the total number of the neighbors. The idea of the angle-based method described in [52] is to move X to a new position $\bar{X}$ such that $\bar{X}$ maximizes the angles $\angle(X, V_i, V_{i-1})$ and $\angle(X, V_{i-1}, V_i)$ for $i = 1, \cdots, M$ with $V_0 := V_M$. Let X_i be the projection of X to the bisector of the angles $\angle(V_{i-1}, V_i, V_{i+1})$ for $i = 1, \cdots, M$ with $V_{M+1} := V_1$, then $\bar{X}$ is approximated in [52] by $\bar{X} = (X_1 + X_2 + \cdots, +X_M)/M$. This method has been extended in [44] for improving quadrilateral mesh quality, but only two-dimensional meshes were discussed in the above two papers.

We make some modifications of the 2D angle-based method as described in [52], in order to deal with surface mesh processing. First, X is projected onto the bisecting plane Π, instead of the bisecting line, of $\angle(V_{i-1}, V_i, V_{i+1})$, as shown in Fig. 3. Note that the plane Π is orthogonal to the plane formed by (V_{i-1}, V_i, V_{i+1}). Secondly, the average of X_i is weighted by a decreasing function of the angle $\angle(V_{i-1}, V_i, V_{i+1})$:

$$\bar{X} = \frac{1}{\sum_{i=1}^{M}(\alpha_i + 1)} \sum_{i=1}^{M} (\alpha_i + 1) X_i, \tag{4}$$

where α_i is the dot product of the normalized vectors of $\overrightarrow{V_i V_{i-1}}$ and $\overrightarrow{V_i V_{i+1}}$. The use of weighted average is due to the fact that a smaller angle $\angle(V_{i-1}, V_i, V_{i+1})$ is more sensitive to the position change of $\bar{X}$. We shall demonstrate in Section 3.3 how this modification can improve the performance of the angle-based method.

The above two modifications are usually sufficient to improve the mesh quality. However, as we mentioned earlier, dealing with surface meshes requires additional care − while the mesh quality is being improved, the geometric features should be preserved as much as possible. To this end, we take the advantage of the local structure tensor by mapping the new position $\bar{X}$ to each of the eigenvectors of the tensor calculated at the old position X and scaling the mapped vectors with the corresponding eigenvalues. Let $\vec{e}_1, \vec{e}_2, \vec{e}_3$ denote the eigenvectors and $\lambda_1, \lambda_2, \lambda_3$

be the corresponding eigenvalues of the local structure tensor valued at X. The modified vertex $\hat{X}$ is calculated as follows:

$$\hat{X} = X + \sum_{k=1}^{3} \frac{1}{1+\lambda_k}((\bar{X} - X) \cdot \vec{e}_k)\vec{e}_k, \tag{5}$$

where $\bar{X}$ is calculated in Equation 4. The use of the eigenvalues as a weighted term in the above equation is essential to preserve the features (with high curvatures) and to keep the improved surface mesh as close as possible to the original mesh by encouraging the vertices to move along the eigen-direction with small eigenvalues (or in other words, with low curvatures). It is worthwhile noting that our tensor-based approach can preserve the features better than the normal-tangent decomposition method as described in [51]. In [51], a vertex moves in the normal and tangent directions independently but the movement within the tangent plane is treated isotropically. This could cause a problem in case of ridge- or valley-like features, where the vertex is encouraged to move only along the direction of ridges or valleys but not the other two (one is normal direction and the other lies on the tangent plane). Our approach, however, can readily handle these cases − the eigenvalues are small along the ridges or valleys (hence moving in this direction is preferred) but large in the other two directions (movements are therefore suppressed). Fig. 2(b) shows the surface mesh with quality improved , compared to the original mesh as shown in Fig. 2(a). Before quality improvement, the minimal and maximal angles are 0.02° and 179.10° respectively. These angles become 14.11° and 135.65° after the improvement (two iterations).

2.3. **Surface Mesh Coarsening.** The surface meshes extracted by isocontouring techniques (e.g., the marching cube method) often contain a large number of elements and are almost uniform everywhere. For instance, the meshing of 1CID, a molecule of 1381 atoms, results in 62,454 nodes and 124,904 triangles in total. The size of the mesh (only for surfaces!) could be over one million nodes for very large molecules and densely discretized volumes. Such a dense mesh may cause computational difficulties in terms of both time and memory on typical modern computers. Adaptive meshes are therefore preferred where fine meshes only occur in regions containing features. In this section we first describe a feature-preserving mesh coarsening approach. The coarsened meshes are in general less smooth due to the deletions of nodes and recalculated triangles. A normal-based surface smoothing technique is also described in order to improve the smoothness of the coarsened mesh.

2.3.1. *Feature-Preserving Mesh Coarsening.* The idea of mesh coarsening is straightforward − choose a node to be deleted and re-triangulate the region containing the incident neighbors. In the following we shall describe these two steps.

The nodes to be deleted should be selected according to a feature-preserving criterion: nodes in "flat" or low-curvature regions should be deleted with higher probability. For this reason, we again utilize the local structure tensor as a way to quantify the features. Let X denote the node being considered for deletion and the neighboring nodes be $V_i, i = 1, \cdots, n$, where n is the total number of the neighbors. The maximal length of the incident edges at X is denoted by $L(X) = \max_{i=1}^{n}\{d(X, V_i)\}$ where $d(.)$ is the Euclidean distance. Apparently $L(X)$ indicates the sparseness of the mesh at X. Let $\lambda_1(X), \lambda_2(X), \lambda_3(X)$ be the eigenvalues of the local structure tensor calculated at X, satisfying $\lambda_1(X) \geq \lambda_2(X) \geq \lambda_3(X)$. Then the node X is deleted if and only if the following condition holds:

$$L(X)^{\alpha} \left(\frac{\lambda_2(X)}{\lambda_1(X)} \right)^{\beta} < T_0, \tag{6}$$

where α and β are chosen to balance between the sparseness and the curvature of the mesh. In our experiments, they both are set as 1.0 by default. The threshold T_0 is user-defined and also dependent on the values of α and β chosen. When α and β are fixed, larger T_0 will cause more nodes to be deleted. By default, it is set as 0.2 in our experiments. For the example illustrated in Fig. 2(c), the coarsened mesh consists of about $8,846$ nodes and $17,688$ triangles, which is about seven times smaller than the original mesh as shown in Fig. 2(b).

Once a node is selected for deletion, all of its associated edges and triangles should be deleted as well. As a result, there is a "hole" surrounded by the neighbors. To fill this "hole", one way is to choose one neighbor and connect it to all the other neighbors. This strategy is called *edge contraction* as employed in [51]. The consequence of several iterations by this method is that some nodes may have a large number of incident edges while others have very few. This "unbalanced" triangulation clearly worsens the mesh quality as often seen in *LBIE* [51]. Much effort may have to be made to flip/swap the edges in order to balance the degrees of all the nodes. In our implementation, we always choose the two neighbors that have fewest degrees (number of incident edges) and connect them unless they are adjacent on the "ring" of the "hole". The connection between the nodes chosen increases their degrees by one and splits the "hole" into two smaller "holes", each of which is then treated recursively in the same way. This type of re-triangulation of the "holes" guarantees balanced degrees between all the neighbors (nodes) on the "holes". It is worth of noting that the mesh coarsening alone may worsen the mesh quality due to the re-triangulation. For the example shown in Fig. 2(b), the minimal and maximal angles are $0.42°$ and $177.85°$ after the coarsening, compared to $14.11°$ and $135.65°$ before the coarsening. Therefore, it is necessary to run the mesh coarsening along with the quality improvement algorithm as described in Section 2.2.2. Fortunately the mesh after coarsening is much smaller; hence running the quality improvement again does not take much extra time but it does significantly improve the mesh quality. For example in 2, the minimal and maximal angles become $20.22°$ and $130.71°$ respectively, after 8 iterations of quality improvements on the coarsened mesh.

2.3.2. *Normal-based Surface Smoothing.* Mesh coarsening can greatly reduce the mesh size to a user-specified order. However, the nodes on the "holes" are often not co-planar; hence the re-triangulation of the "holes" often results in a bumpy surface mesh, as we can see in Fig. 2(c). The bumpiness can be reduced or removed by smoothing the surface meshes.

There are vertex-based and normal-based approaches for mesh smoothing. The first approach is to adjust the position of each vertex based on certain criteria measured on the vertex such as the Laplacian operator [41] and mean curvature flows [15, 31]. In contrast, the normal-based approach smoothes meshes using the normal information of neighboring triangles, which turns out to preserve sharp features and prevent volume shrinkages [13] better than the vertex-based approach. In our mesh generator, we utilize the normal-based approach.

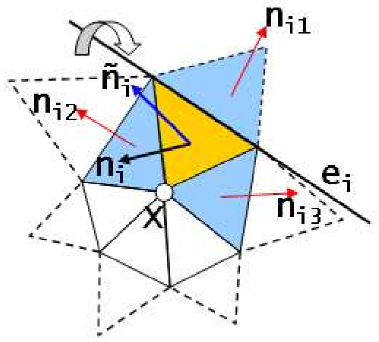


FIGURE 4. Illustration of normal-based surface mesh smoothing.

A number of feature-preserving filters seen in signal/image processing have been extended to surface mesh smoothing, such as the bilateral filter [20], mean [32] and median [45] filters. The idea that we shall describe is based on the well-known anisotropic image filter, called the Perona-Malik method [34], which was extended to anisotropic vector diffusion in [47]. Let us first describe the isotropic smoothing at a vertex X. Our goal is to move X such that all the incident triangles and their neighboring ones (dashed triangles in Fig. 4) have smooth normals. Without lose of generality, let us consider one of the incident triangles (colored in yellow in Fig. 4), whose normal is denoted by $\mathbf{n_i}$. For the mesh to be smooth, the normal $\mathbf{n_i}$ could be adjusted by the average of the normals of the adjacent three triangles (colored in blue in Fig. 4), that is, $\tilde{\mathbf{n}}_\mathbf{i} = (\mathbf{n_{i1}} + \mathbf{n_{i2}} + \mathbf{n_{i3}})/3$. To adjust the normal from $\mathbf{n_i}$ to $\tilde{\mathbf{n}}_\mathbf{i}$, we rotate the vertex X by the angle $\theta_i = \angle(\mathbf{n_i}, \tilde{\mathbf{n}}_\mathbf{i})$ about the edge $\mathbf{e_i}$. The new position is denoted by $R(X; \mathbf{e_i}, \theta_i)$. By summing up all the adjustments, we have the following smoothing scheme at X:

$$\tilde{X} = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i R(X; \mathbf{e_i}, \theta_i) \tag{7}$$

where w_i is the area of the i^{th} triangle incident to X.

As we mentioned earlier, the above scheme is isotropic, meaning that the smoothing is independent of the features. The consequence of the isotropic scheme is that sharp features often become weakened or even disappear after a certain number of iterations. To remedy this problem, we incorporate the anisotropic diffusion [34, 47] into the calculation of $\tilde{\mathbf{n}}_\mathbf{i}$ as follows:

$$\tilde{\mathbf{n}}_\mathbf{i} = \frac{1}{\sum_{j=1}^{3} e^{\kappa(\mathbf{n_i}\cdot\mathbf{n_{ij}})}} \sum_{j=1}^{3} e^{\kappa(\mathbf{n_i}\cdot\mathbf{n_{ij}})} \mathbf{n_{ij}} \tag{8}$$

where κ is a user-defined positive parameter. All normals $\mathbf{n_i}$ and $\mathbf{n_{ij}}$ are assumed to be normalized. The weighting function used in Equation 8 is decreasing with respect to the angle between $\mathbf{n_i}$ and $\mathbf{n_{ij}}$ such that sharp features where the angle between $\mathbf{n_i}$ and $\mathbf{n_{ij}}$ are large can be preserved. Fig. 2(d) shows the result after the normal-based mesh smoothing.

2.4. **Tetrahedron Generation.** Once the surface triangulation is generated with good quality, *Tetgen* [38, 39] can produce tetrahedral meshes with user-controlled quality. Besides the triangulated molecular surface, our toolchain will have three other outputs for a given molecule: the interior tetrahedral mesh, the exterior tetrahedral mesh, and both meshes together. For the interior tetrahedral mesh we force all atoms to be on the mesh nodes. The exterior mesh is generated between the molecular surface and a bounding sphere whose radius is set as about 20 times larger than the size of the molecule being considered (or some other user-specified radii). The size of a molecule is defined as follows. We first compute the average of all atom centers; then the maximal distance between the atoms to the averaged center is defined as the size of the molecule. In the following section, we shall see several experiments on both surface triangulation and tetrahedral generation.

3. Results

In this section we shall show the mesh generation results on a number of molecules. The atom coordinates are extracted from the PDB files [8] and the radius for each atom is given in a user-specified table. Our program can also read the PQR format in which the radius information is attached to each atom [16].

3.1. **Surface Generation and Processing.** Let us begin with the surface generation and processing. All examples shown below are generated using the following parameters unless otherwise specified. The blobbyness of the Gausian kernel function is set as -0.5. All 3D volumes are discretized based on the spacing distance of $0.5\AA$ per grid. The initial molecular surfaces are extracted using the marching cube method [30] at the isovalue of 1.0. The threshold T_0 for the mesh coarsening in Equation 6 is set as 0.2.

The first example that we studied is a small molecule of 1,381 atoms, taken from the Protein Data Bank (PDB-ID: 1CID). A volume of $130 \times 132 \times 176$ grids is generated using the Gaussian-based blurring technique as described in Section 2.1. Part of the initial surface was shown in Fig. 2(a). The exterior surface view of the entire molecule after quality improvement is shown in Fig. 5(a). As we mentioned earlier, there are $62,454$ nodes and $124,904$ triangles in this mesh. After mesh coarsening and normal-based smoothing combined with quality improvements, the mesh size is reduced to $8,846$ nodes and $17,688$ triangles. The mesh quality is also greatly improved, from $0.02°$ (minimal angle) and $179.10°$ (maximal angle) for initial mesh, to $20.22°$ (minimal angle) and $130.71°$ (maximal angle) for the processed mesh. Fig. 5(b) and (c) show the exterior and interior views of the processed mesh. Thanks to the feature-based adaptivity in our mesh tools, most important features in the original mesh are well preserved in spite of a significant size reduction.

Another example we investigated is the mouse acetylcholinesterase (mAChE) molecule which has a total of $8,362$ atoms yielding a volume of $162 \times 142 \times 192$ grids. The initial mesh contains $96,046$ nodes and $192,088$ triangles and is reduced to $19,795$ nodes and $39,586$ triangles after the mesh coarsening, as shown in Fig. 6(a) where the active site is indicated by a dashed rectangle near the center on the left and an interior and closer look at the active site is shown on the right. The mesh quality is also significantly improved $-$ $23.22°$ (minimal angle) and $125.93°$ (maximal angle), in contrast to the minimal angle ($0.02°$) and maximal angle ($179.55°$) in the original mesh.

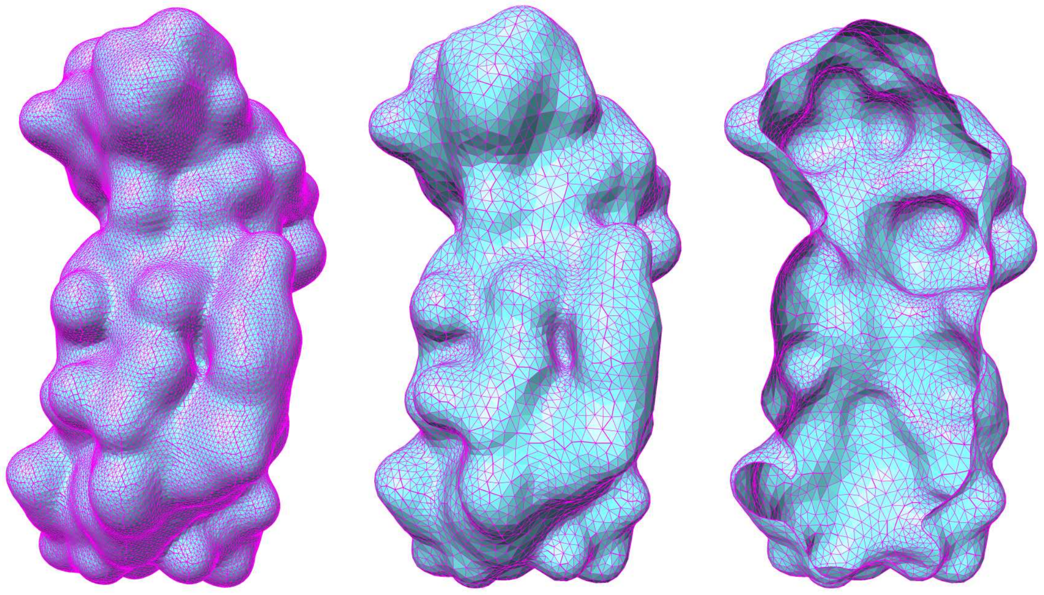
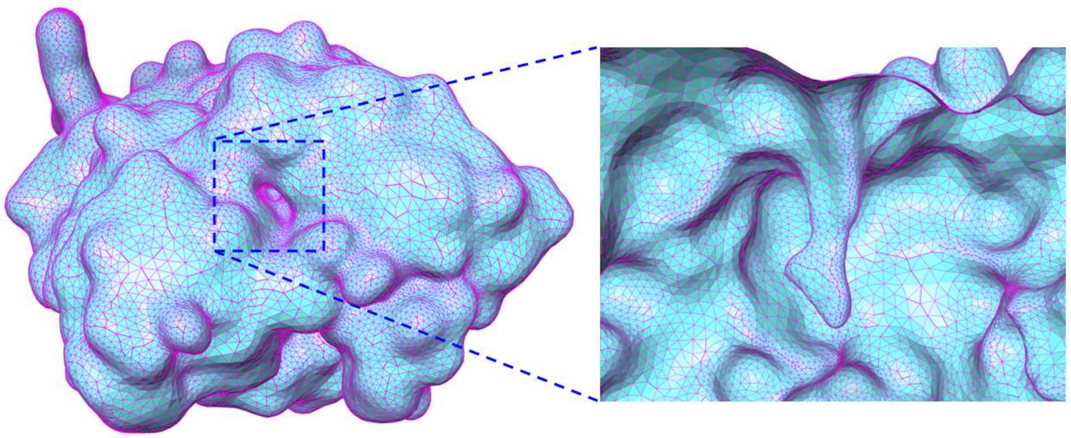


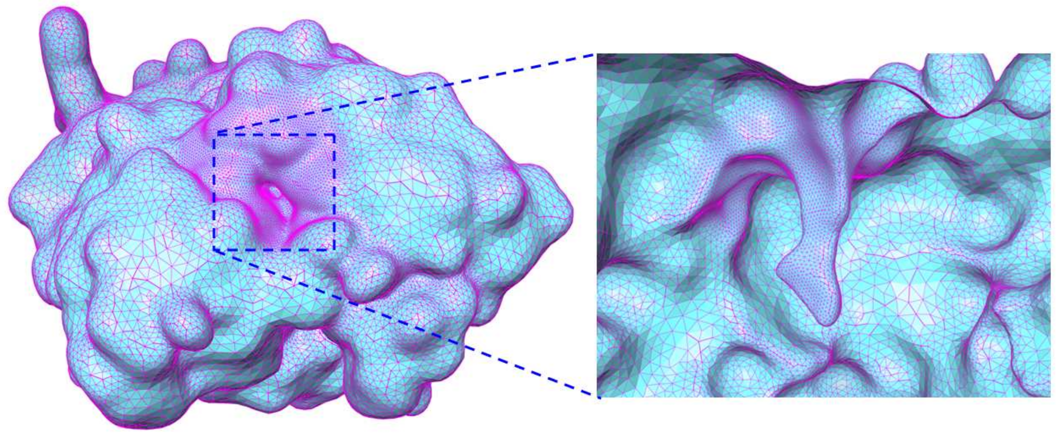
FIGURE 5. Illustration of the surface generation and post-processing. This molecule is taken from PDB (ID: 1CID) and has 1,381 atoms. (a) The mesh is generated using the marching cube method and the quality is improved based on the approach we described in Section 2.2.2. (b) The mesh is coarsened and smoothed, which reduces the mesh size by more than a factor of 7. But, most important features are still well preserved. (c) An interior view of the processed mesh.

Another motivation of studying the mAChE molecule here is to demonstrate the constrained mesh coarsening that turns out to be very useful in refining a region of particular interest. In molecular simulation, it is a common strategy to have a dense mesh near the active sites and a sparser mesh elsewhere, in order to maximize the simulation accuracy while still keeping a low computational cost. This can be easily done by restricting the mesh coarsening everywhere except in user-defined regions (e.g., the active sites). In case of the mAChE molecule we use a few spheres to define the boundary of the active site. The union of the spheres is where the mesh coarsening is prohibited. Fig. 6(b) shows the constrained mesh coarsening, where the active site has a much denser mesh than the rest of the molecular surface. The resulting mesh has $22,753$ nodes and $45,502$ triangles and the mesh quality is improved to $24.09°$ (minimal angle) and $125.94°$ (max angle).

In Fig. 7, we show the meshes generated using our tools on two large molecules. One is 1BVP trimer with $8,109$ atoms and the other is mAChE tetramer with $36,650$ atoms. The volume size for 1BVP is $206 \times 172 \times 216$. The original mesh has $150,182$ nodes and $300,360$ triangles. It was improved, coarsened, and smoothed using the methods described in Section 2. The processed mesh, as shown in Fig.



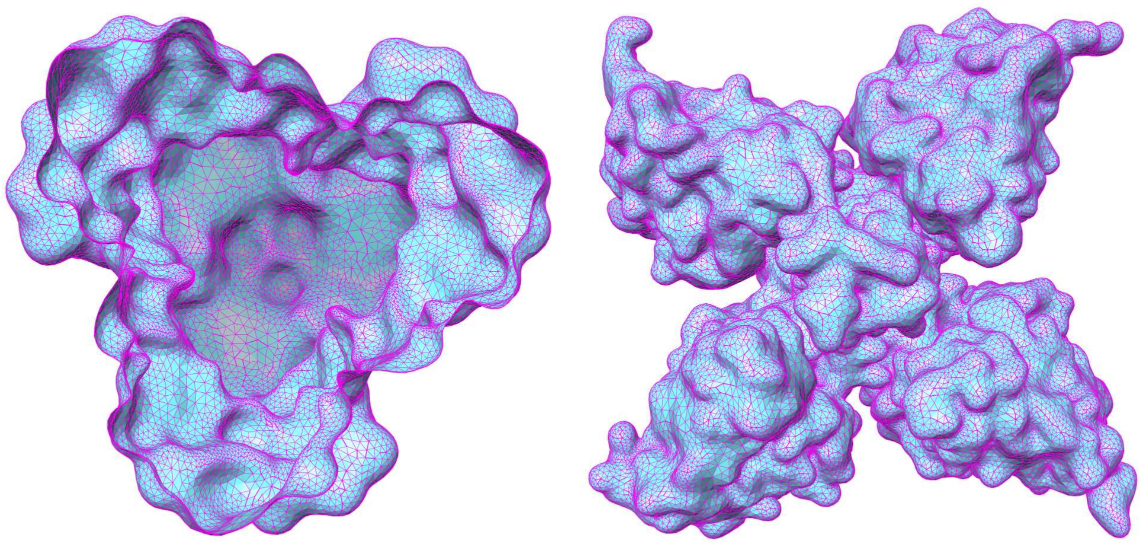
(a) Regular mesh coarsening



(b) Constrained mesh coarsening

Figure 6. Illustration of the regular and constrained mesh coarsening on the mAChE molecule. (a) The mesh is generated by a regular mesh process as presented in Section 2. The region indicated by a dashed rectangle is the active site that has the same sampling rate as the other regions. (b) No coarsening is applied near the active site; therefore, the active site has a much denser mesh than elsewhere. An interior and zoomed-in look at the active site, rotated by 90°, is shown in both (a) and (b).

7(a), has $27,658$ nodes and $55,312$ triangles. In addition, the mesh quality was improved from $0.02°$ (minimal angle) and $179.84°$ (maximal angle) to $21.63°$ (minimal angle) and $133.99°$ (maximal angle) respectively. The Gaussian blurring technique on mAChE tetramer generates a volume of $298 \times 250 \times 266$ grids. The original mesh has $444,362$ nodes and $888,748$ triangles and the size is reduced to $54,951$ nodes and $109,926$ triangles using our method. The mesh quality is also improved – the



(a) The 1BVP trimer (b) The mAChE tetramer

FIGURE 7. Mesh generation on two very large molecules. (a) An interior view of the 1BVP trimer with 8,109 atoms. The original mesh has 150,182 nodes and 300,360 triangles. Our mesh coarsening algorithm reduces the size to 27,658 nodes and 55,312 triangles. (b) An exterior view of the mAChE tetramer that has a total of 36,650 atoms. The original mesh has 444,362 nodes and 888,748 triangles and the size is reduced to 54,951 nodes and 109,926 triangles using our method.

minimal angle increases from 0.00° to 19.57° and the maximal angle decreases from 179.75° to 127.79°. An exterior view of the processed mesh is shown in Fig. 7(b).

3.2. **Adaptive Tetrahedral Generation.** Shown here are a few examples of the adaptive tetrahedral meshes generated using *Tetgen* [38, 39], based on the surface meshes triangulated and processed by the methods described above. Fig. 8(a) and (b) show the interior and exterior tetrahedral meshes of the molecule 1CID respectively. Since all 1,381 atoms are forced to be on the interior mesh nodes, the interior mesh looks more uniform and denser than the exterior mesh. The bounding sphere for the exterior mesh is usually chosen to be about 20 times larger than the size of the molecule being considered. But for better illustration, the sphere we show here is about twice as big as the molecule. The sphere is meshed into 4,350 nodes and 8,696 triangles. Fig. 8(c) shows a closer look at the exterior mesh near the molecular surface. The interior mesh has a total of 14,769 and 64,080 tetrahedra. The size of the exterior mesh is however slightly varying on the radius of the bounding sphere. If the sphere is about twice as big as the molecule, as shown in 8(b), the exterior mesh has 21,899 nodes and 94,736 tetrahedra. However, if the sphere is chosen to be 20 times bigger than the molecule, the exterior mesh would contain 22,318 nodes and 97,229 tetrahedra.

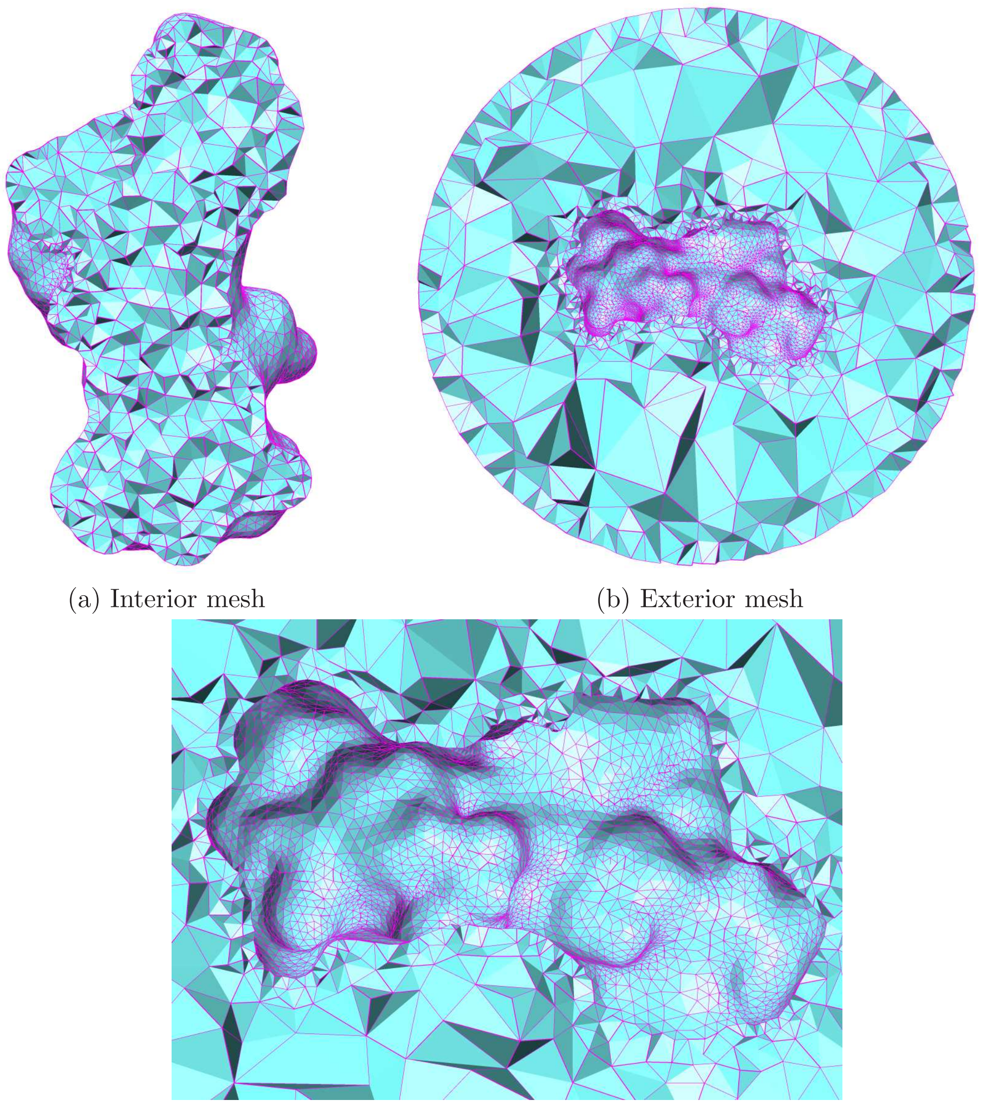
Fig. 9 shows the tetrahedralization of the mAChE molecule with finer meshes at the active site. The interior mesh and a close look at the active site are shown in

Fig. 9 (a) and (b) respectively. The interior mesh again has all atoms on the nodes and the whole mesh consists of $46,582$ nodes and $223,365$ tetrahedra. In contrast, the exterior tetrahedral mesh has a total of $42,765$ nodes and $181,942$ tetrahedra and, unlike the 1CID molecule, is smaller than the interior mesh. This is because a large number of interior nodes have to be created to guarantee that all $8,362$ atoms are on the mesh nodes. Fig. 9 (c) shows a cross-section of the exterior mesh generated and a closer look is shown in Fig. 9 (d).

We summarize the mesh generation results on a number of molecules in Table 1. All surface meshes are generated and processed using the parameters as given in Section 3.1. For each molecule, the first row of **Surf-N** and **Surf-T** stand respectively for the numbers of nodes and triangles in the initial surface mesh extracted using the marching cube method [30]. The **Surf-N** and **Surf-T** in the second row are the numbers of nodes and triangles after the mesh post-processing (quality improvement, coarsening, and smoothing). Note that all molecules shown here are coarsened only once except 1CID, 1BBH, mAChE tetramer that are coarsened twice. The first row of **Min-Ang** and **Max-Ang** for each molecule represents the minimal and maximal angles before quality improvement while the second row stands for the minimal and maximal angles after the mesh post-processing. The **Tet-N** and **Tet-T** in the first row of each molecule are the numbers of nodes and tetrahedra respectively in the interior tetrahedral mesh, while the **Tet-N** and **Tet-T** in the second row correspond to the numbers of nodes and tetrahedra respectively in the exterior tetrahedral mesh. The bounding spheres for all exterior tetrahedral meshes are 20 times larger than the size of the corresponding molecules.

3.3. **Mesh Quality Analysis.** As we saw above, the mesh quality can be greatly improved using the angle-based approach as described in Section 2.2.2. Compared to the approach in [52], the weighted vertex adjustment we formulate in Equation 4 proves to be more efficient in mesh quality improvement. In Fig. 10, we compare the performance of the original angle-based method presented in [52] and the modified method described in Section 2.2.2. We test both methods for ten iterations (the horizontal axis) on the original surface mesh of the mAChE monomer. The blue curve (the one on the bottom) and the green one on the top stand respectively for the minimal and maximal angles of the entire mesh with quality improved by the original method [52]. The other two curves are the minimal and maximal angles of the entire mesh processed by the weighted angle-based method as described in Section 2.2.2. We can see that the modified method demonstrates a better performance than the original method by producing larger minimal angles and smaller maximal angles in each iteration. Note that since no mesh coarsening was carried out in this experiment, the minimal and maximal angles here are slightly "worse" than those shown in Table 1 (see the entries of mAChE1 or mAChE2).

Although the minimal and maximal angles are good measures of the mesh quality, we cannot tell from these two values how the mesh quality is distributed in the entire meshes. For this reason, we believe that drawing a histogram that shows the distribution of all angles in a mesh would tell us more about the mesh quality. To this end, we divide the angle range from $0°$ to $180°$ into 18 intervals by every $10°$. The histogram in each interval $[i \times 10° - 10°, i \times 10°], i = 1, \cdots, 18$, stands for the percentage of the angles lying in that interval over the total number of angles in a mesh (note that each triangle in the mesh will be counted three times for its three angles). We consider four molecules as examples here: 1CID (1,381 atoms), 1TIM

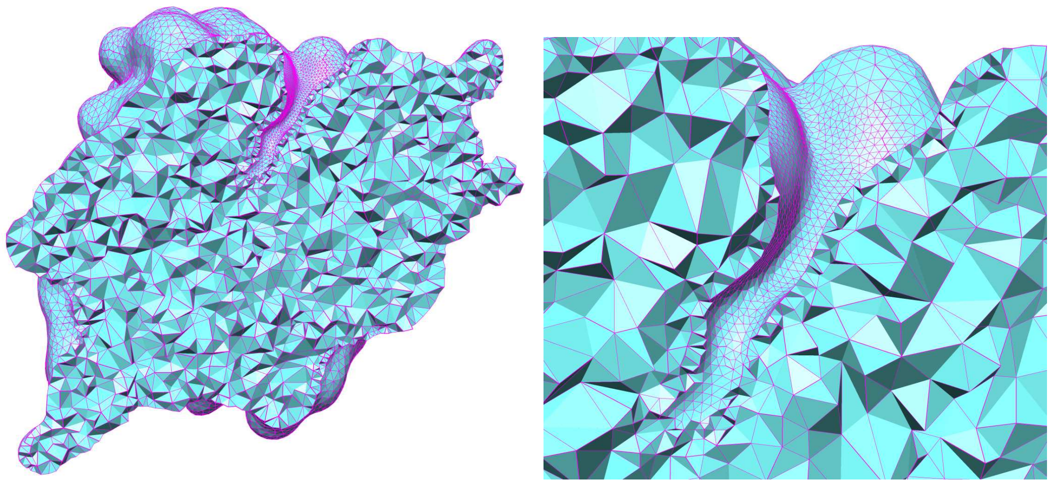


(a) Interior mesh (b) Exterior mesh

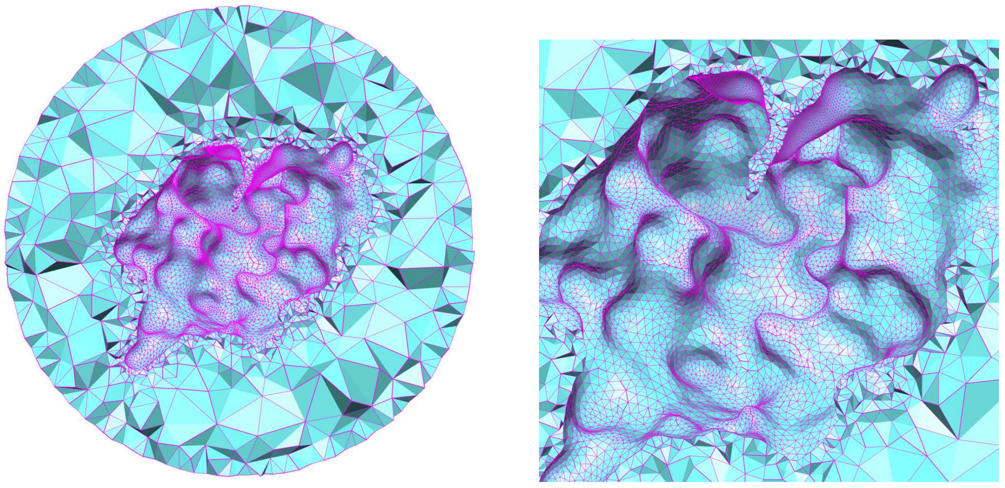
(c) A close view of (b)

FIGURE 8. Tetrahedral meshes generated using *Tetgen* for the molecule 1CID. (a) The interior mesh, where all atoms are forced to be on the mesh nodes. (b) The exterior mesh between the molecular surface and the bounding sphere. The size the sphere is about twice as big as the molecule. (c) A closer look at the exterior mesh near the molecular surface.

(3,740 atoms), mAChE monomer (8,362 atoms), and mAChE tetramer (36,650 atoms). It is interesting that the histograms of the initial meshes (i.e., quality not improved) for all four molecules are almost identical. Therefore we use only one curve to represent the histograms of all four initial meshes, as shown in blue (the thick and dashed one) in Fig. 11. The remaining four curves (the one for 1TIM is



(a) Interior mesh (b) A close look at the active site



(c) Exterior mesh (d) A close view of (c)

Figure 9. Tetrahedral meshes for the mAChE molecule. (a) The interior mesh consisting of 46, 582 nodes and 223, 365 tetrahedra. (b) A closer look at the active site. (c) The exterior mesh with 42, 765 and 181, 942 tetrahedra. (d) A closer view of (c).

almost identical to the histogram of mAChE monomer) are the histograms of the four molecules after the quality improvement using the method described in Section 2.2.2. From these curves we can see that about $85\% - 90\%$ of the angles lie in the range of $[40°, 80°]$, while in the original meshes there are only 50% of the angles in this range. Like the histograms of the initial meshes, the improved meshes also show very similar histograms, implying that the modified angle-based method we described is somewhat data-independent, at least for the meshes extracted from the Gaussian-blurred molecular maps when the same set of parameters are used.

Table 1. A Summary of Surface and Tetrahedral Mesh Generations

Molecules	Atom#	Surf-N	Surf-T	Min-Ang	Max-Ang	Tet-N	Tet-T	Timing
1CID		62,454	124,904	0.02°	179.10°	6,657	30,774	
	1,381	3,500	6,996	24.74°	119.01°	13,228	57,531	47s+8s
1BBH		73,156	146,308	0.02°	179.69°	12,261	57,326	
	1,934	6,370	12,736	21.67°	117.56°	17,660	75,571	68s+13s
1L3R		79,314	158,624	0.02°	179.55°	24,210	109,129	
	2,855	13,542	27,080	23.39°	128.25°	28,986	123,307	69s+24s
1TIM		90,912	181,820	0.02°	179.45°	31,161	140,826	
	3,740	17,332	34,660	22.49°	123.39°	35,194	149,759	90s+29s
1BVP		150,182	300,360	0.02°	179.84°	54,070	253,799	
	8,109	27,658	55,312	21.63°	133.99°	51,136	215,669	182s+48s
mAChE1		96,046	192,088	0.02°	179.55°	42,456	204,821	
	8,362	19,795	39,586	23.22°	125.93°	38,968	165,648	100s+36s
mAChE2		96,046	192,088	0.02°	179.55°	46,908	222,925	
	8,362	22,753	45,502	24.09°	125.94°	42,964	181,103	104s+40s
mAChE3		444,362	888,748	0.00°	179.75°	134,204	682,573	
	36,650	54,951	109,926	19.57°	127.79°	92,805	386,505	556s+105s

Surf-N and **Surf-T**: Numbers of surface mesh nodes and triangles, respectively.
Min-Ang and **Max-Ang**: Minimal and maximal angles of the surface meshes.
Tet-N and **Tet-T**: Numbers of nodes and tetrahedra, respectively.
Timing: First is the time in seconds for surface generation and post-processing.
Second is the time in seconds for tetrahedral mesh generation (both interior and exterior).
mAChE1: the mAChE monomer with regular mesh coarsening.
mAChE2: the mAChE monomer with constrained mesh coarsening.
mAChE3: the mAChE tetramer with regular mesh coarsening.

4. Discussion

4.1. **A Fast Gaussian-Blurring Implementation.**

As we mentioned in Section 2.1, the Gaussian blurring function in Equation 2 has an advantage, compared to Equation 1, that the decay rate C_0 is a constant for all atoms. This can lead to a fast Gaussian blurring implementation using the recursive scheme as described in [46]. Let us rewrite Equation 2 into the following standard Gaussian function:

$$G(\mathbf{x}) = \sum_{i=1}^{N} f_i(\mathbf{x}) \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{2\sigma^2}}, \tag{9}$$

where

$$f_i(\mathbf{x}) = \sqrt{\frac{-\pi}{C_0}} e^{-C_0 r_i^2} \tag{10}$$

stands for the function value (or signal magnitude) at i^{th} atom centered at $\mathbf{x}_i$ with radius r_i. The $\sigma = \sqrt{-\frac{1}{2C_0}}$ is the standard deviation of the Gaussian function. Given the negative constant C_0 and a list of centers $\mathbf{x}_i$ and radii r_i of atoms, the following steps can be implemented to achieve a fast Gaussian-blurring calculation:

(1) Compute σ and $f_i(\mathbf{x})$.

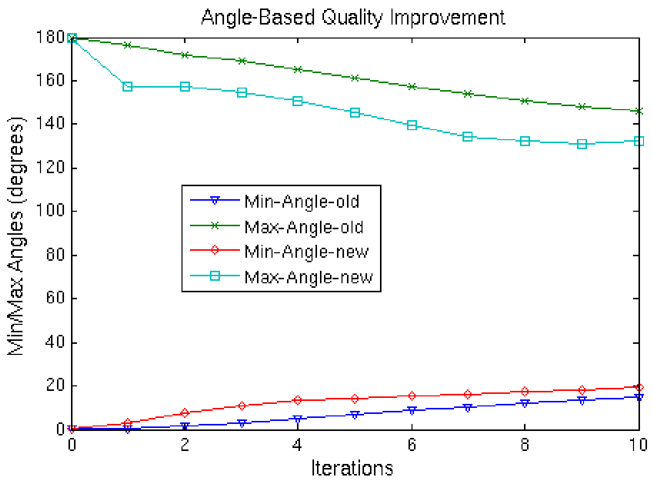


FIGURE 10. Comparisons between the original and modified angle-based methods for mesh quality improvement. The molecule being studied is mAChE monomer. The initial surface mesh is extracted using the parameters specified in Section 3.1. Ten iterations are executed for both methods. The two curves on the bottom represent the minimal angles improved with respect to the number of iterations. The other two curves are the maximal angles during the quality improvement. The blue (bottom) and green (top) curves stand for the results by the original angle-based method while the other two are the results by the modified method.

(2) Discretize the space into grids (same as we did in Section 2.1). For each atom $\mathbf{x}_i$, find the nearest grid $\mathbf{m}_i$. Approximate the function value at $\mathbf{m}_i$ with $f_i(\mathbf{x})$.
(3) Run the recursive Gaussian filtering [46] on the signals $f_i(\mathbf{x})$ at all grids $\mathbf{m}_i$ using the parameter σ.

It was shown in [46] that the numerical error of the recursive implementation of Gaussian filter is relatively less than 0.68% compared to the analytical Gaussian function. But the biggest gain of this scheme is that it can be even faster than the optimized FFT-based implementation. In case of the molecular maps as we outlined above, there is another type of error that comes from the approximation of $\mathbf{x}_i$ with the nearest grid point $\mathbf{m}_i$. However, this digitization error should be quite small when the space is densely sampled.

4.2. **Quality Improvements on User-Defined Meshes.** In addition to the PQR [16] or PDB [8] files as inputs, our mesh toolchain can also read and process a user-defined arbitrary triangular mesh that has very low quality or is too large for numerical simulations. As an example, we consider the mAChE monomer again and generate the mesh by the widely used MSMS tool [37]. Fig. 12(a) shows the mesh generated by MSMS. As we can see, this mesh looks a little "bumpy" on the surface and more seriously contains a lot of very "sharp" triangles. The minimal and maximal angles of this mesh are $0.00°$ and $176.92°$ respectively. There are only 47% of the total angles lying in the range of $[40°, 80°]$. The numbers of nodes and

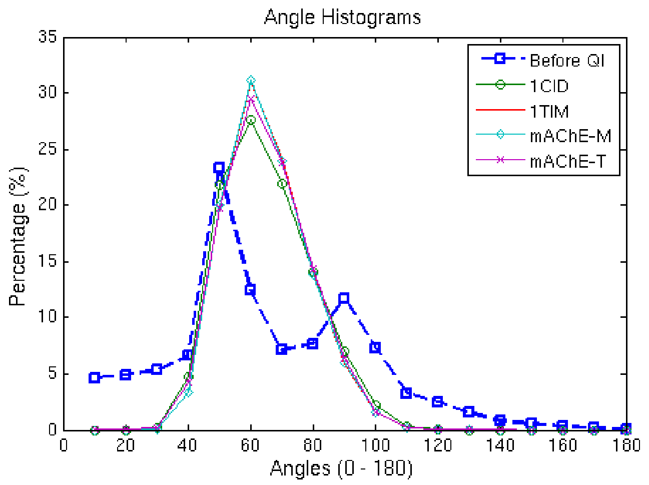
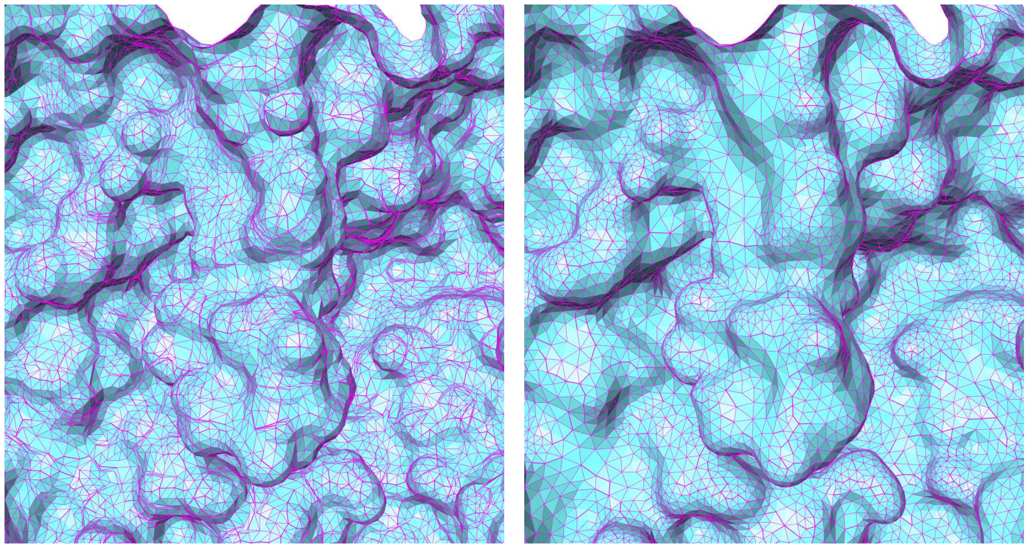


FIGURE 11. Histograms (in percentage) of the angles in the initial and processed meshes. Four molecules are considered: 1CID, 1TIM, mAChE monomer and mAChE tetramer. Since all four molecules demonstrate almost identical histograms before quality improvement (QI), we show only one curve (thick and dashed one in blue color) to represent the histogram for the initial meshes. The other four, shown in different colors, are the histograms of the processed meshes (1TIM has an almost identical histogram to that of mAChE monomer). Most angles in the processed meshes lie in the range of $[40^\circ, 80^\circ]$.

triangles of this mesh are 62, 402 and 124, 804 respectively. After our mesh quality improvement, coarsening, and smoothing, we reduce the mesh size by almost half to 35, 022 nodes and 70, 044 triangles. The mesh quality is significantly improved to 25.56° (minimal angle) and 125.46° (maximal angle). In contrast to the 47%, now we have 88% of the angles lying in the range of $[40^\circ, 80^\circ]$. The picture showing the same region of the mAChE monomer can be seen in Fig. 12(b).

4.3. **Hierarchical Mesh Generation and Multigrid-based Solvers.** By coarsening the surface triangular meshes using different parameters T_0 in Equation 6, we can achieve a hierarchy of meshes at different levels of details. Fig. 13 illustrates the meshes of the 1BVP trimeric molecule at four different levels of resolutions − $T_0 = 0.1, 0.3, 0.5$ and 0.7. We can see that, as the T_0 parameter increases, the mesh size decreases and some small features start to disappear. From our experiments, $T_0 = 0.2$ seems to be a good tradeoff to balance the accuracy and the mesh size (cost) and is what we used in the previously shown examples. An initial mesh, if it is too large, may be subject to coarsening for two or more times. This turns out to preserve features better than simply coarsening the mesh once with a high T_0.

Having access to a hierarchy of meshes can also be leveraged to build fast solvers for partial differential equations (PDEs) posed in the volume mesh around the biological structure, or posed on the surface mesh. It is well-known that the most efficient numerical methods for solving certain classes of PDEs (called *elliptic* PDEs) are based on multilevel or hierarchical methods, which are extensions of the original



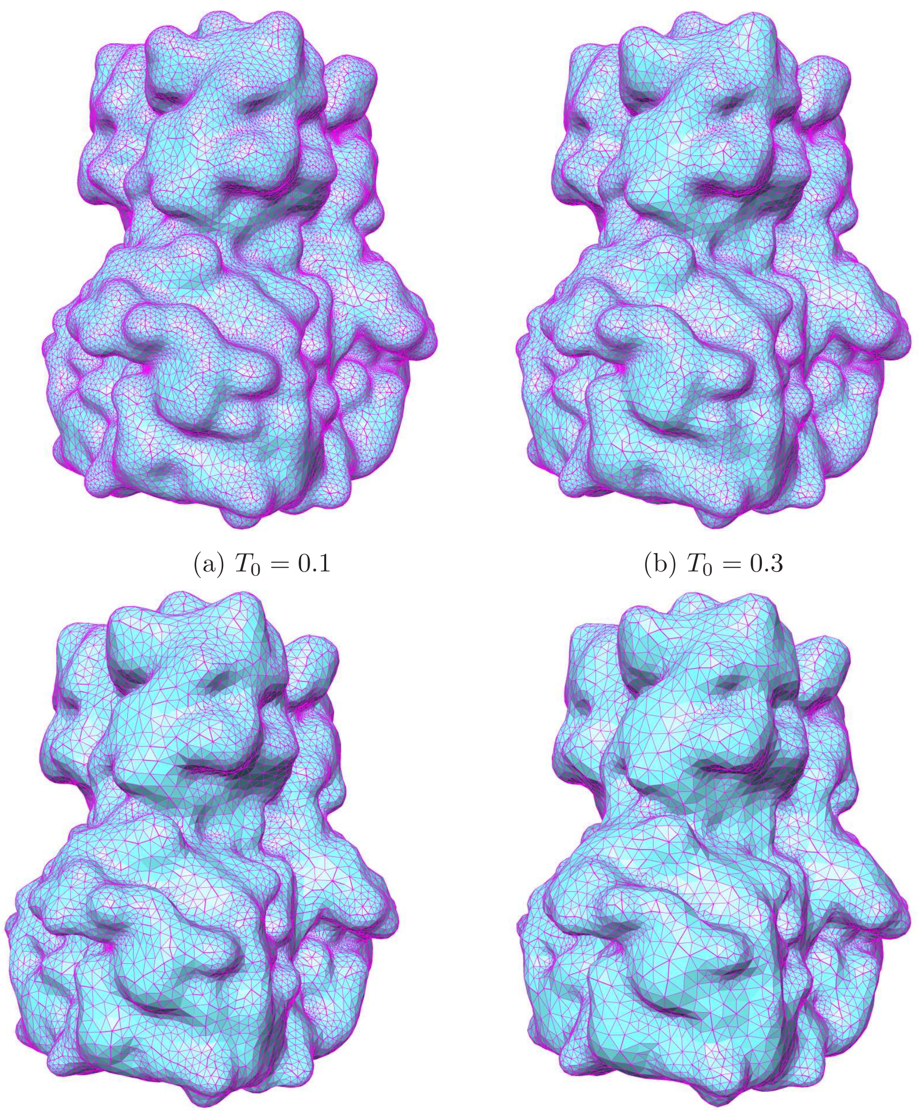
(a) Original mesh by MSMS (b) Mesh processed by our methods

FIGURE 12. Mesh quality improvement on the molecular surface generated by MSMS. (a) Original mesh generated by MSMS has $62,402$ nodes and $124,804$ triangles, respectively. The minimal and maximal angles of this mesh are $0.00°$ and $176.92°$ respectively. (b) After mesh coarsening and quality improvement, the mesh size is about half as large as the original one and the quality becomes much better: $25.56°$ (minimal angle) and $125.46°$ (maximal angle).

multigrid algorithms developed for the Poisson equation in the 1970s [11, 23]. The basic idea behind such multilevel methods for solving a linear system $Au = f$ is the recursive application of a simple two-level method having the following form (cf. [24, 12]):

(1) *Smoothing*. Perform iterative relaxations on $A^f u^f = f^f$ on the fine mesh using, for instance, the Gauss-Seidel method. This step has an effect of reducing high frequency errors on the fine mesh.
(2) *Restriction*. Compute the residual error $r^f = f^f - A^f u^f$ and down-sample it to the coarse mesh by $r^c = I_f^c r^f$, where I_f^c is called *restriction* operator.
(3) *Equation-Solving*. Solve the residual equation $A^c e^c = r^c$ on the coarse mesh to obtain the errors e^c. The matrix A^c is defined either by the finite element discretization on the coarse mesh or by the mesh-independent Galerkin formula $A^c = I_f^c A^f (I_f^c)^T$. *This step is accomplished by doing the two-level scheme recursively, until the coarse mesh is small enough so that direct solution of the coarse problem is negligible.*
(4) *Prolongation*. Interpolate the errors onto the fine mesh by $e^f = I_c^f e^c$, where $I_c^f = (I_f^c)^T$, and correct the errors at the fine grid by $u^f \leftarrow u^f + e^f$.

In the setting of nested uniform mesh refinement from an initial coarse mesh, one can show that for PDE problems like the Poisson equation, the cost of a single recursive application of the two-level scheme is linear in the dimension of the discrete problem on the finest mesh. Unfortunately, complex geometries and PDE



(a) $T_0 = 0.1$ (b) $T_0 = 0.3$

(c) $T_0 = 0.5$ (d) $T_0 = 0.7$

FIGURE 13. Hierarchical Mesh Generation by Coarsening (experiments on 1BVP). The parameter T_0 is described in Equation 6. The original mesh has $150,182$ nodes and $300,360$ triangles. With different coarsening parameters, we have: (a) Nodes = 26,842, Triangles = 53,680. (b) Nodes = 19,588, Triangles = 39,172. (c) Nodes = 12,882, Triangles = 25,760. (d) Nodes = 9,480, Triangles = 18,956. The typical parameter we use for controlling mesh coarsening is 0.2.

coefficients make it difficult to use either uniform mesh refinement or generate nested mesh hierarchies. If nesting is lost (necessary for example to retain the interpolatory property of surface meshes of varying resolution), then one must use generalizations of the basic multilevel idea; techniques include non-nested geometric multigrid methods [9] or algebraic multigrid methods [1]. Moreover, if local adaptivity is required to deal with geometric or solution singularities, then a different class of multilevel method must be employed to retain linear storage and solver complexity; these are based on stabilizations of the Hierarchical Basis Method (cf. [4, 3] for detailed discussions of this class of methods and further references).

The mesh hierarchies of varying resolution that our toolchain generates are well-positioned for leveraging such fast solver technology. The key objects that must be constructed to use most of these multilevel techniques is the restriction operator I_f^c. For example, following the work in [9], we can calculate the restriction operator as follows:

$$I_f^c = \begin{pmatrix} \phi_1^c(n_1) & \phi_1^c(n_2) & \cdots & \phi_1^c(n_{N_f}) \\ \phi_2^c(n_1) & \phi_2^c(n_2) & \cdots & \phi_2^c(n_{N_f}) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_{N_c}^c(n_1) & \phi_{N_c}^c(n_2) & \cdots & \phi_{N_c}^c(n_{N_f}) \end{pmatrix}_{N_c \times N_f}, \tag{11}$$

where $\phi_i^c(n_j)$ is the i^{th} interpolation function, defined on the coarse mesh and valued at the j^{th} node of the fine mesh. N_c and N_f are the numbers of nodes of the coarse and fine meshes respectively. Alternatively, I_f^c can be constructed essentially algebraically, using geometrical information about the mesh [1], or it can be constructed completely algebraically, using ideas from graph theory [7].

4.4. **Applications.** The toolchain described above has been implemented in ANSI-C and incorporated into the Finite Element ToolKit (*FeTK*) software suite [19, 25] as one of its major components, called *GAMer* (*G*eometry-preserving *A*daptive *M*esh*er*). The *FeTK* libraries and tools are developed by the Holst Research Group at UCSD and are designed to solve coupled systems of partial differential equations (PDE) and integral equations (IE) using parallel adaptive multilevel finite element methods. On top of this highly portable software toolkit are two widely used application-oriented molecular modeling tools: the APBS [5] and the SMOL [40]. APBS is an adaptive Poisson-Boltzmann solver, designed for simulating electrostatic properties of molecules in salty, aqueous media (http://apbs.sourceforge.net/). SMOL is designed to solve the Smoluchowski diffusion equation (for a description of this software, see http://mccammon.ucsd.edu/smol/). Both molecular modeling tools are built on top of *FeTK* [19, 25], where our high-quality mesh generator plays a critical role in providing the geometric domains for the PDE/IE solvers.

5. Conclusion

We presented a mesh generation toolchain that has been demonstrated on a number of molecules to generate smooth, adaptive, and high quality meshes with features well preserved. Along with the software package that will be made available in source form under the GNU General Public License, we also described in detail several mesh processing algorithms adapted from other related work, including the angle-based mesh quality improvement, adaptive feature-preserving mesh coarsening, and normal-based surface mesh smoothing. Although this paper focused on the molecular mesh generation having the center and radius of each atom as inputs, our toolchain can also be applied to generate and process meshes from arbitrary scalar volumes (e.g., reconstructed 3D medical or cellular data) or low-quality surface meshes provided by the user (e.g., molecular surfaces by MSMS [37]). One of our future directions is to extend the triangular and tetrahedral meshing toolchain presented here to other types such as quadrilateral/hexahedral meshes.

Acknowledgments

The first author was supported in part by NSF Award 0411723, DOE Award DE-FG02-05ER25707, and NBCR (http://nbcr.sdsc.edu/). The second author was supported in part by NSF Awards 0411723, 0511766, and 0225630, and DOE Awards DE-FG02-05ER25707 and DE-FG02-04ER25620. The third author was supported in part by the NSF (MCB-0506593), NIH (GM31749), HHMI, CTBP, NBCR, W.M. Keck Foundation, and Accelrys, Inc.. We would like to thank Chandrajit Bajaj, Randy Bank, Yuhui Cheng, Gary Huber, Benzhuo Lu, Yongjie Zhang, and Yongcheng Zhou for helpful discussions related to this project.

References

[1] M. Adams. Evaluation of three unstructured multigrid methods on 3D finite element problems in solid mechanics. *Int. J. Numer. Meth. Engng*, 55:519–534, 2002.
[2] M. Ainsworth and J.T. Oden. *A Posterori Error Estimation in Finite Element Analysis*. Wiley-Interscience, 2000.
[3] B. Aksoylu, S. Bond, and M. Holst. An adyssey into local refinement and multilevel preconditioning III: Implementation and numerical experiments. *SIAM J. Sci. Comput.*, 25(2):478–498, 2003.
[4] B. Aksoylu and M. Holst. Optimality of multilevel preconditioners for local mesh refinement in three dimensions. *SIAM J. Numer. Anal.*, 44(3):1005–1025, 2006.
[5] N.A. Baker, D. Sept, S. Joseph, M. Holst, and J.A. McCammon. Electrostatics of nanosystems: application to microtubules and the ribosome. *Proc. Natl. Acad. Sci.*, 98:10037–10041, 2001.
[6] R. Bank and M. Holst. A new paradigm for parallel adaptive meshing algorithms. *SIAM Review*, 45:291–323, 2003.
[7] Randolph E. Bank and R. Kent Smith. An algebraic multilevel multigraph algorithm. *SIAM J. Sci. Comput.*, 23:1572–1592, 2002.
[8] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The protein data bank. *Nucleic Acids Research*, 28:235–242, 2000.
[9] M.L. Bittencourt, C.C. Douglas, and R. A. Feijo. Non-nested multigrid methods for linear problems. *Numer. Meth. PDE*, 17:313–331, 2001.
[10] J.F. Blinn. A generalization of algegraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, 1982.
[11] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Math. Comp.*, 31:333–390, 1977.
[12] W.L. Briggs, V.E. Henson, and S.F. McCormick. *A multigrid tutorial (2nd ed.)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2000.
[13] C.-Y. Chena and K.-Y. Cheng. A sharpness dependent filter for mesh smoothing. *Computer Aided Geometric Design*, 22(5):376–391, 2005.
[14] M.L. Connolly. Analytical molecular surface calculation. *J. Appl. Cryst.*, 16(5):548558, 1983.
[15] M. Desbrun, M. Meyer, P. Schroder, and A.H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proc. of SIGGRAPH'99*, page 317324, 1999.
[16] T.J. Dolinsky, J.E. Nielsen, J.A. McCammon, and N.A. Baker. PDB2PQR: an automated pipeline for the setup, execution, and analysis of poisson-boltzmann electrostatics calculations. *Nucleic Acids Research*, 32:665–667, 2004.
[17] B.S. Duncan and A.J. Olson. Shape analysis of molecular surfaces. *Biopolymers*, 33:231–238, 1993.

[18] J.-J. Fernandez and S. Li. An improved algorithm for anisotropic nonlinear diffusion for denoising cryo-tomograms. *Journal of Structural Biology*, 144:152–161, 2003.
[19] FeTK. http://www.fetk.org.
[20] S. Fleishman, I. Drori, and D. Cohen-Or. Bilateral mesh denoising. In *Proc. of SIGGRAPH'03*, pages 950–953, 2003.
[21] L.A. Freitag and C.F. Ollivier-Gooch. Tetrahedral mesh improvement using swapping and smoothing. *Int. J. Numerical Methods in Engineering*, 40(21):39794002, 1997.
[22] J.A. Grant and B.T. Pickup. A gaussian description of molecular shape. *J. Phys. Chem.*, 99:3503–3510, 1995.
[23] W. Hackbusch. A fast iterative method solving Poisson's equation in a general region. In R. Bulirsch, R. D. Griegorieff, and J. Schröder, editors, *Proceedings of the Conference on the Numerical Treatment of Differential Equations, Oberwolfach, July 1976; Lecture Notes in Mathematics, Number 631*, pages 51–62, Berlin, 1978. Springer-Verlag.
[24] W. Hackbusch. *Multi-grid Methods and Applications*. Springer-Verlag, Berlin, Germany, 1985.
[25] M. Holst. Adaptive numerical treatment of elliptic systems on manifolds. *Advances in Computational Mathematics*, 15:139–191, 2001.
[26] K.H. Huebner, D. Dewhirst, D.E. Smith, and T.G. Byrom. *The Finite Element Method for Engineers*. Wiley-IEEE, 2001.
[27] I. Jollife. *Principal Component Analysis*. Springer-Verlag (NY), 1986.
[28] T. Ju, F. Losasso, S. Schaefer, and J.Warren. Dual contouring of hermite data. In *Proc. of SIGGRAPH'02*, pages 339–346, 2002.
[29] B. Lee and F.M. Richards. The interpretation of protein structures: estimation of static accessibility. *J. Mol. Biol.*, 55(3):379–400, 1971.
[30] W.E. Lorensen and H. E. Cline. Marching cubes: a high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.
[31] M. Meyer, M. Desbrun, P. Schroder, and A.H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Proc. of Visualization and Mathematics*, 2002.
[32] Y. Ohtake, A.G. Belyaev, and I.A. Bogaevski. Mesh regularization and adaptive smoothing. *Computer-Aided Design*, 33(4):789–800, 2001.
[33] P.P. Pebay and T.J. Baker. Analysis of triangle quality measures. In *Mathematics of Computation*, volume 72, pages 1817–1839, 2003.
[34] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.
[35] F.M. Richards. Areas, volumes, packing, and protein structure. *Ann. Rev. Biophys. Bioeng.*, 6:151–156, 1977.
[36] M.F. Sanner. Python: a programming language for software integration and development. *J. Mol. Graphics Mod.*, 17:57–61, 1999.
[37] M.F. Sanner, A.J. Olson, and J.C. Spehner. Reduced surface: an efficient way to compute molecular surfaces. *Biopolymers*, 38:305–320, 1996.
[38] H. Si. Tetgen: a quality tetrahedral mesh generator and three-dimensional delaunay triangulator. Technical Report 9, Weierstrass Institute for Applied Analysis and Stochastics, 2004. (software download: http://tetgen.berlios.de).
[39] H. Si and K. Gartner. Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *Proceedings of the 14th International Meshing Roundtable*, 2005.
[40] Y. Song, Y Zhang, T Shen, C.L. Bajaj, J.A. McCammon, and N.A. Baker. Finite element solution of the steady-state Smoluchowski equation for rate constant calculations. *Biophys. J.*, 86:2017–2029, 2004.
[41] G. Taubin. A signal processing approach to fair surface design. In *Proc. of SIGGRAPH'95*, pages 351–358, 1995.
[42] M. Totrov and R. Abagyan. The contour-buildup algorithm to calculate the analytical molecular surface. *Journal of Structural Biology*, 116:138–143, 1996.
[43] J. Weickert. *Anisotropic Diffusion In Image Processing*. ECMI Series, Teubner-Verlag, Stuttgart, 1998.
[44] H. Xu and T.S. Newman. 2d fe quad mesh smoothing via angle-based optimization. In *Proc., 5th Int'l Conf. on Computational Science*, pages 9–16, 2005.
[45] H. Yagou, Y. Ohtake, and A. Belyaev. Mesh smoothing via mean and median filtering applied to face normals. pages 124–131, 2002.

[46] I.T. Young and L.J. Vliet. Recursive implementation of the Gaussian filter. *Signal Processing*, 44:139–151, 1995.

[47] Z. Yu and C. Bajaj. A segmentation-free approach for skeletonization of gray-scale images via anisotropic vector diffusion. In *Proc. Int'l Conf. Computer Vision and Pattern Recognition*, pages 415–420, 2004.

[48] Z. Yu and C. Bajaj. A structure tensor approach for 3D image skeletonization: applications in protein secondary structural analysis. In *Proceedings of IEEE International Conference on Image Processing*, pages 2513–2516, 2006.

[49] Z. Yu and C. Bajaj. Computational approaches for automatic structural analysis of large bio-molecular complexes. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, in press, 2007.

[50] Y. Zhang, C. Bajaj, and B-S Sohn. 3D finite element meshing from imaging data. *Special issue of Computer Methods in Applied Mechanics and Engineering on Unstructured Mesh Generation*, 194(48-49):5083–5106, 2005.

[51] Y. Zhang, G. Xu, and C. Bajaj. Quality meshing of implicit solvation models of biomolecular structures. *The special issue of Computer Aided Geometric Design (CAGD) on Applications of Geometric Modeling in the Life Sciences*, 23(6):510–530, 2006.

[52] T. Zhou and K. Shimada. An angle-based approach to two-dimensional mesh smoothing. In *Proceedings of the Ninth International Meshing Roundtable*, pages 373–384, 2000.