

# High-Fidelity Geometric Modeling for Biomedical Applications

Zeyun Yu<sup>a\*</sup>, Michael J. Holst<sup>a</sup>, and J. Andrew McCammon<sup>b</sup>

<sup>a</sup>Department of Mathematics, University of California, San Diego, La Jolla, CA 92093

<sup>b</sup>Department of Chemistry & Biochemistry and Department of Pharmacology, University of California, San Diego, La Jolla, CA 92093

We describe a combination of algorithms for high fidelity geometric modeling and mesh generation. Although our methods and implementations are application-neutral, our primary target application is multiscale biomedical models that range in scales across the molecular, cellular, and organ levels. Our software toolchain implementing these algorithms is general in the sense that it can take as input a molecule in PDB/PQR forms, a 3D scalar volume, or a user-defined triangular surface mesh that may have very low quality. The main goal of our work presented is to generate high quality and smooth surface triangulations from the aforementioned inputs, and to reduce the mesh sizes by mesh coarsening. Tetrahedral meshes are also generated for finite element analysis in biomedical applications. Experiments on a number of bio-structures are demonstrated, showing that our approach possesses several desirable properties: feature-preservation, local adaptivity, high quality, and smoothness (for surface meshes). The availability of this software toolchain will give researchers in computational biomedicine and other modeling areas access to higher-fidelity geometric models.

Keywords: mesh generation, geometric modeling, adaptivity, feature-preserving, biomedical application

## 1. Introduction

The finite element method (FEM) is a powerful technique for finding the approximate solution of a partial differential equation (PDE) where the domain boundaries of a given problem are so complex that other approaches such as the finite difference method may have difficulties or even fail. Although the finite element method was originally developed in the 1950's mainly for civil or aeronautical engineering or material science problems, it has now become one of the fundamental numerical approaches for solving PDE's that arise in many other fields, including biomedical simulation, which is the focus of this article.

In the finite element method, a complex domain is discretized into a number of reasonably "good" elements, such that a set of basis functions can be defined on the elements to approximate the solution. The discretization of a domain into small elements is usually referred to as mesh generation [2,7,28]. The quality of a mesh can be measured by the associated angles – a mesh having angles close or equal to  $0^\circ$  or  $180^\circ$  is said to have

---

\*E-mail: zeyu@math.ucsd.edu

very low-quality. Although different types of meshes may be generated depending on the numerical solvers being employed, we restrict ourselves in this paper to triangular (surface) and tetrahedral (volumetric) meshes. In particular, we consider mesh generation for biomedical applications, taking as inputs a set of centers and radii of atoms in a molecule, a 3D scalar volume reconstructed with CT/MRI medical imaging techniques, or any user-defined triangular surface meshes that may have extremely low quality.

Mesh generation from a set of atoms requires good approximation of molecular surfaces. There are two primary ways of constructing such surfaces: one is based on the “hard sphere” model [37] and the other is based on the level set of a “soft” Gaussian function [24]. 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: *van der Waals surface* (VDW), *solvent accessible surface* (SAS), and *solvent excluded surface* (SES) [17,24,31,37]. The molecular surface can be represented analytically by a list of seamless spherical patches [17,46] and triangular meshes can be generated using such tools as *MSMS* [39]. In contrast, the “soft” model treats each atom as a Gaussian-like smoothly decaying scalar function in  $\mathbb{R}^3$  [11,20,24]. The molecular surfaces are then approximated by appropriate level sets (or isocontours) of the Gaussian function [11,20]. As we shall see shortly, mesh generation from the other two sources, namely, a 3D scalar volume or a user-defined surface mesh, can be thought of as a special case or subroutine of the mesh generation pipeline starting from a molecule. For this reason, we shall consider mainly the mesh generation from a set of atoms given in a PQR [19] or PDB [9] format.

The goal of the present paper is to describe a set of algorithms and implementations to construct high quality surface triangulations and decent tetrahedral meshes for biomedical simulations. In general, a good mesh should have the following properties: (1) feature-preserving (meshes being as close as possible to the original surfaces), (2) adaptivity (meshes being made denser at areas of interest or high-curvature regions), and (3) high quality (triangles being as equilateral as possible). Among the existing mesh generation software toolkits, *Tetgen* is an example of an outstanding software tool that can generate high quality tetrahedral meshes from a triangulated surface, using constrained Delaunay tetrahedralizations [40,41]. But the major limitation of this tool is that the input surface mesh must be “good enough”, meaning that angles such as those close to  $0^\circ$  or  $180^\circ$  are not allowed. Another mesh generation tool, called *LBIE* [54], integrates the surface triangulation, tetrahedron generation and smoothing into a stand-alone software package. However, *LBIE* cannot directly handle a surface mesh – if the input is a mesh, it must be converted into a volume by, for example, the signed distance transformation [54,55]. In addition, the input volumes must be in a power of 2, due to the octree data structure used in this tool. Although the qualities of the interior tetrahedral meshes generated by *LBIE* are usually good, there are some “sharp” triangles on the surfaces and hence some tetrahedra near the surfaces can be poorly shaped.

In the present paper we describe a combination of algorithms and implementations to produce feature-preserving, adaptive, and high quality surface (triangular) and volumetric (tetrahedral) meshes for bio-modeling and other applications. Like *LBIE* [54], our method is based on the level set of a Gaussian kernel function that approximates the molecular surface of a given molecule. Because the surface meshes extracted by iso-contouring techniques always contain many “skinny” triangles, we develop a variant of the angle-

based approach [56] 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 are constructed using the constrained Delaunay triangulation as implemented in *Tetgen* [40,41].

## 2. Feature-Preserving Adaptive Mesh Generation and Processing

In this section we describe the algorithms and implementations 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 [19] or PDB files [9] 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 molecular mesh generation, the initial surface meshes are defined by the level sets of the summation of Gaussian kernel functions defined at each atom with center  $\mathbf{c}_i$  and radius  $r_i$ :

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

where the negative parameter  $B_i$  is called the *blobbyness* that controls the spread of characteristic function 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. Our experiments on a number of molecules show that the blobbyness at  $-0.5$  produces a good approximation in molecular simulations, although high-resolution molecular surfaces may require a blobbyness as small as  $-2.5$ .

The molecular surface is defined by the level set as  $\{\mathbf{x} \in \mathfrak{R}^3 : F(\mathbf{x}) = t_0\}$ , where  $t_0$  is the isovalue [11,24,55], which typically lies in the range from 2.0 to 4.0, based on our experiments. The surface (triangular) mesh can be constructed by either the marching cube method [32] or the dual contouring method [30]. In our toolchain we employ an implementation of the marching cube method. The basic idea of this method is to triangulate the cubes (or voxels) that intersect with the isosurface. Each cube contains 8 grid points, each of which could be either outside or inside the iso-surface, yielding a total of  $2^8 = 256$  cases to be considered. The key of the marching cube method is to triangulate each of such cubes according to a pre-given table containing these 256 cases. Fig.2(A) shows an example of the isosurface extracted using this method. From this example, we can see that: (a) the marching cube method can extract very smooth surfaces, but (b) many triangles are extremely “sharp” and could lead to poor approximation quality in finite element analysis. Therefore, we need to improve the mesh quality while retaining the smoothness of the initial triangular mesh.

## 2.2. Surface Mesh Quality Improvement

Mesh quality in 2D planes or 3D volumes is usually measured by the ratio of radii of the circumscribed to inscribed circles in each triangle or spheres in each tetrahedron. 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 [35]. 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 [23]. 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 surface shapes such as the curvature. Therefore, besides the mesh quality we mentioned above, several other criteria must be taken into consideration in particular for surface mesh improvement: (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

The local structure tensor has been used for anisotropic image smoothing [22,47] and protein secondary structure detection [52,53]. The idea of the local structure tensor is derived from the well-known principal component analysis (PCA) technique [29]. 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 [22,53]. 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:

$$T(\mathbf{v}) = \sum_{i=1}^{N_1} \begin{pmatrix} n_x^{(i)}n_x^{(i)} & n_x^{(i)}n_y^{(i)} & n_x^{(i)}n_z^{(i)} \\ n_y^{(i)}n_x^{(i)} & n_y^{(i)}n_y^{(i)} & n_y^{(i)}n_z^{(i)} \\ n_z^{(i)}n_x^{(i)} & n_z^{(i)}n_y^{(i)} & n_z^{(i)}n_z^{(i)} \end{pmatrix}, \quad (2)$$

where  $(n_x^{(i)}, n_y^{(i)}, n_z^{(i)})$  is the normal of the  $i^{th}$  neighbor of a vertex  $\mathbf{v}$  and  $N_1$  is the total number of neighbors. The normal of a vertex is defined by the weighted 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 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 mesh generator for molecules.

As discussed in [22,53], the eigenvalues and eigenvectors of the structure tensor indicate the overall distribution of the vectors within a neighborhood and categorize the features locally into different types: spheres, lines, and planes. In case of surface meshes, the types

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:  $\lambda_1 \approx \lambda_2 \gg \lambda_3 \approx 0$ .
- Planes:  $\lambda_1 \gg \lambda_2 \approx \lambda_3 \approx 0$ .

### 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  $\mathbf{x}$  be the vertex that we want to move for quality improvement. The (incident) neighbors are denoted by  $\mathbf{v}_i, i = 1, \dots, N_2$ , where  $N_2$  is the total number of the neighbors. The idea of the angle-based method described in [56] is to move  $\mathbf{x}$  to a new position  $\bar{\mathbf{x}}$  such that  $\bar{\mathbf{x}}$  maximizes the angles  $\angle(\mathbf{x}, \mathbf{v}_i, \mathbf{v}_{i-1})$  and  $\angle(\mathbf{x}, \mathbf{v}_{i-1}, \mathbf{v}_i)$  for  $i = 1, \dots, N_2$ , where  $\mathbf{v}_0 := \mathbf{v}_{N_2}$ . Let  $\mathbf{x}_i$  be the projection of  $\mathbf{x}$  to the bisector of the angles  $\angle(\mathbf{v}_{i-1}, \mathbf{v}_i, \mathbf{v}_{i+1})$  for  $i = 1, \dots, N_2$  and  $\mathbf{v}_{N_2+1} := \mathbf{v}_1$ , then  $\bar{\mathbf{x}}$  is approximated in [56] by  $\bar{\mathbf{x}} = (\mathbf{x}_1 + \mathbf{x}_2 + \dots + \mathbf{x}_{N_2})/N_2$ . This method has been extended in [48] 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 [56], in order to deal with surface mesh processing. First,  $\mathbf{x}$  is projected onto the bisecting plane  $\Pi$ , instead of the bisecting line, of  $\angle(\mathbf{v}_{i-1}, \mathbf{v}_i, \mathbf{v}_{i+1})$ , as shown in Fig.3. Note that the plane  $\Pi$  is orthogonal to the plane formed by  $(\mathbf{v}_{i-1}, \mathbf{v}_i, \mathbf{v}_{i+1})$ . Secondly, the average of all the projections  $\{\mathbf{x}_i, i = 1, \dots, N_2\}$  is weighted by a decreasing function of the angle  $\angle(\mathbf{v}_{i-1}, \mathbf{v}_i, \mathbf{v}_{i+1})$ :

$$\bar{\mathbf{x}} = \frac{1}{\sum_{i=1}^{N_2} (\alpha_i + 1)} \sum_{i=1}^{N_2} (\alpha_i + 1) \mathbf{x}_i, \quad (3)$$

where  $\alpha_i$  is the dot product of the normalized vectors of  $\overrightarrow{\mathbf{v}_i \mathbf{v}_{i-1}}$  and  $\overrightarrow{\mathbf{v}_i \mathbf{v}_{i+1}}$ . The use of weighted average is due to the fact that a smaller angle  $\angle(\mathbf{v}_{i-1}, \mathbf{v}_i, \mathbf{v}_{i+1})$  is more sensitive to the position change of  $\mathbf{x}$ . We shall demonstrate in Section 4.1 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  $\bar{\mathbf{x}}$  onto each of the eigenvectors of the tensor calculated at the old position  $\mathbf{x}$ . The projected values are weighted by the corresponding eigenvalues as follows. 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  $\mathbf{x}$ . The modified vertex  $\hat{\mathbf{x}}$  is calculated as follows:

$$\hat{\mathbf{x}} = \mathbf{x} + \sum_{k=1}^3 \frac{1}{1 + \lambda_k} ((\bar{\mathbf{x}} - \mathbf{x}) \cdot \vec{e}_k) \vec{e}_k, \quad (4)$$

where  $\bar{\mathbf{x}}$  is calculated in Equation 3. The use of eigenvalues as a weighted term in the above equation is essential to preserve surface 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 [55]. In [55], 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 should 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  $0.02^\circ$  and  $179.10^\circ$  respectively. These angles become  $14.11^\circ$  and  $135.65^\circ$  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 triangles for very large molecules and densely discretized volumes. Such a mesh often poses a big challenge in terms of both time and memory consumptions in numerical simulation. 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 reconnected 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 concept of mesh coarsening is straightforward – we choose a node, delete it, and then re-triangulate the region containing the incident neighbors. 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 to quantify the features. Let  $\mathbf{x}$  denote the node being considered for deletion and the neighboring nodes be  $\mathbf{v}_i, i = 1, \dots, N_2$ , where  $N_2$  is the total number of the incident neighbors. The maximal length of the incident edges at  $\mathbf{x}$  is denoted by  $L(\mathbf{x}) = \max_{i=1}^{N_2} \{d(\mathbf{x}, \mathbf{v}_i)\}$  where  $d(\cdot)$  is the Euclidean distance. Apparently  $L(\mathbf{x})$  indicates the sparseness of the mesh at  $\mathbf{x}$ . Let  $\lambda_1(\mathbf{x}), \lambda_2(\mathbf{x}), \lambda_3(\mathbf{x})$  be the eigenvalues of the local structure tensor calculated at  $\mathbf{x}$ , satisfying  $\lambda_1(\mathbf{x}) \geq \lambda_2(\mathbf{x}) \geq \lambda_3(\mathbf{x})$ . Then the node  $\mathbf{x}$  is deleted if and only if the following condition holds:

$$L(\mathbf{x})^\alpha \left( \frac{\lambda_2(\mathbf{x})}{\lambda_1(\mathbf{x})} \right)^\beta < T_0, \quad (5)$$

where  $\alpha$  and  $\beta$  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  $\alpha$  and  $\beta$  chosen. When  $\alpha$  and  $\beta$  are fixed, larger  $T_0$  will cause more nodes to be deleted. For the example illustrated in Fig.2(C), the coarsened mesh consists of about 8,846 nodes and 17,688 triangles, 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 other neighbors. This strategy is called *edge contraction* as employed in [55]. 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, and as a result much effort needs to be made to flip/swap the edges in order to balance the degrees of all the nodes. In our implementation, we always choose 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  $0.42^\circ$  and  $177.85^\circ$  after the coarsening, compared to  $14.11^\circ$  and  $135.65^\circ$  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 the example in Fig.2, the minimal and maximal angles become  $20.22^\circ$  and  $130.71^\circ$  respectively, after 8 iterations of quality improvements on the coarsened mesh.

### 2.3.2. Normal-based Anisotropic Surface Smoothing

Mesh coarsening can greatly reduce the mesh size to a user-specified order. However, the nodes on the “holes” are usually not co-planar; hence re-triangulating the “holes” often results in bumpy surface meshes, as we can see in Fig.2(C). The bumpiness can be reduced or removed by smoothing the surface meshes. Note that the mesh smoothing seen below is different from the mesh quality improvement we discussed in Section 2.2, although in some literature, mesh quality improvement is also referred to as mesh smoothing.

There are vertex-based and normal-based approaches for mesh smoothing. The first approach is to move the vertices based on certain criteria, such as the Laplacian operator [44] and mean curvature flows [18,33]. In contrast, the normal-based approach smooths meshes using the normal information of neighboring triangles, which turns out to preserve sharp features and prevent volume shrinkages [14] better than the vertex-based approach. In our mesh generator, we utilize the normal-based approach.

A number of feature-preserving filters seen in signal/image processing have been extended to surface mesh smoothing, such as the bilateral filter [21], mean [34] and median [49] filters. The idea that we shall describe is based on the well-known anisotropic image filter, called the Perona-Malik method [36], which was extended to anisotropic vector diffusion in [51]. Let us first describe the isotropic smoothing at a vertex  $\mathbf{x}$ . Our goal is to move  $\mathbf{x}$  such that the normal vectors smoothly change among all the incident triangles

and their neighboring ones (dashed triangles in Fig.4). Without loss of generality, let us consider one of the incident triangles (colored in yellow in Fig.4), whose normal vector is denoted by  $\mathbf{n}_i$ . For the mesh to be smooth, the normal vector  $\mathbf{n}_i$  could be adjusted by the average of the normal vectors of the adjacent three triangles (colored in blue in Fig.4), that is,  $\tilde{\mathbf{n}}_i = (\mathbf{n}_{i1} + \mathbf{n}_{i2} + \mathbf{n}_{i3})/3$ . To adjust the normal vector from  $\mathbf{n}_i$  to  $\tilde{\mathbf{n}}_i$ , we rotate the vertex  $\mathbf{x}$  around the edge  $\mathbf{e}_i$  by an angle of  $\theta_i = \angle(\mathbf{n}_i, \tilde{\mathbf{n}}_i)$ . The new position is denoted by  $R(\mathbf{x}; \mathbf{e}_i, \theta_i)$ . By summing up all the adjustments, we have the following smoothing scheme at  $\mathbf{x}$ :

$$\tilde{\mathbf{x}} = \frac{1}{\sum_{i=1}^{N_2} w_i} \sum_{i=1}^{N_2} w_i R(\mathbf{x}; \mathbf{e}_i, \theta_i) \quad (6)$$

where  $w_i$  is the area of the  $i^{\text{th}}$  triangle incident to  $\mathbf{x}$ .

As we mentioned earlier, the above scheme is isotropic, meaning that the smoothing is independent of the features. The consequence of an 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 [36,51] into the calculation of  $\tilde{\mathbf{n}}_i$  as follows:

$$\tilde{\mathbf{n}}_i = \frac{1}{\sum_{j=1}^3 e^{\kappa(\mathbf{n}_i \cdot \mathbf{n}_{ij})}} \sum_{j=1}^3 e^{\kappa(\mathbf{n}_i \cdot \mathbf{n}_{ij})} \mathbf{n}_{ij} \quad (7)$$

where  $\kappa$  is a user-defined positive parameter. All normal vectors  $\mathbf{n}_i$  and  $\mathbf{n}_{ij}$  are assumed to be normalized. The weighting function used in Equation 7 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* [40,41] 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 40 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 spacial 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

### 3.1. Mesh Generation for Biomolecules

The example we demonstrate here is the mouse acetylcholinesterase (mAChE) monomer, which has a total of 8,362 atoms. The blobbyness of the Gaussian kernel function is

set as  $-0.5$  and the 3D volume is discretized by  $0.67\text{\AA}$  per grid yielding a volume of  $126 \times 111 \times 148$  grids. The initial molecular surface containing 51,260 nodes and 102,516 triangles is extracted using the marching cube method [32] at the isovalue of 3.5. After the mesh coarsening, where the threshold  $T_0$  in Equation 5 is set as 0.2, the mesh size is reduced to 15,207 nodes and 30,410 triangles, as shown in Fig.5(A). Note that the active site has been colored differently and meshed more densely than elsewhere, which is preferable in finite element methods in order to achieve a higher accuracy around the regions of interest. A closer look at the active site, rotated by  $90^\circ$ , is shown in Fig.5(D). The quality of the entire mesh is significantly improved:  $25.4^\circ$  (minimal angle) and  $126.3^\circ$  (maximal angle), in contrast to the minimal angle ( $0.0^\circ$ ) and maximal angle ( $179.4^\circ$ ) in the original mesh generated by the marching cube method.

The adaptive tetrahedral meshes are computed by constrained Delaunay triangulation using *Tetgen* [40,41], based on the surface mesh in Fig.5(A). Fig.5(B) and (C) show the cross sections of the interior and exterior tetrahedral meshes of the mAChE molecule respectively. Since all atoms are forced to lie on the interior mesh nodes, the interior mesh is more uniform and denser than the exterior mesh. The bounding sphere for the exterior mesh is usually chosen to be about 40 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. Fig.5(E) and (F) show a closer look at the interior and exterior meshes, respectively, near the active site. The entire mesh has a total of 90,163 nodes and 564,026 tetrahedra. The computational costs are 24 seconds (molecular surface extraction from PDB), 58 seconds (surface mesh post-processing), and 50 seconds (tetrahedral mesh generation) on a single Intel Xeon 2.0GHz CPU.

### 3.2. Mesh Generation for Other Applications

In addition to biomolecular mesh generation, our meshing tool can also take other forms of inputs such as an arbitrary 3D scalar volume (e.g., medical imaging data) or a user-defined arbitrary triangular mesh that has very low quality or is too large for numerical simulations. To demonstrate the generality of our mesh tool, we show here several other examples: the neuromuscular junction given as a triangulated surface mesh [15,45], a human head model given as a scalar volume (mimicked by the signed distance function) [54], and an actual CT medical image of human cardiovascular system. Fig.6(A) shows the surface mesh after the post-processing as discussed in Section 2. In order to simulate the ionic diffusion from the pre-synaptic active zone to the post-synaptic area, the interior tetrahedral mesh is generated in the synaptic domain as shown in Fig.6(B) and a closer look at the mesh is given in Fig.6(C). This mesh contains 125,231 nodes and 779,967 tetrahedra. The interior and exterior tetrahedral meshes of the human head model are shown in Fig.6(D) and (E) respectively. This mesh consists of 32,754 nodes and 200,988 tetrahedra. In Fig.6(F) we show an example of human cardiovascular system. The mesh is generated directly from the segmented blood vessels [5] using the marching cube method. The region indicated by arrow is enlarged and shown on the left-bottom. The mesh after quality improvement and mesh smoothing by our methods demonstrates much more regular angles and smoother surfaces as shown in Fig.6(G).

## 4. Discussion

### 4.1. Mesh Quality Analysis

As we saw in previous section, the mesh quality of surface meshes can be greatly improved using the angle-based approach as described in Section 2.2.2. Compared to the approach in [56], the weighted vertex movement we formulate in Equation 3 proves to be more efficient in mesh quality improvement. In Fig.7, we compare the performance of the original angle-based method presented in [56] and the modified method described in Section 2.2.2. Both algorithms are executed for ten iterations (the horizontal axis) on the original surface mesh of the mACH 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 [56]. 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.

The minimal and maximal angles, used as quality measurement of surface meshes, are significantly improved after our mesh post-processing methods. These criteria, however, may fail in measuring tetrahedral mesh quality as the well-known “slivers” may exist in tetrahedral meshes (a tetrahedron is called a sliver if it has “decent” edge lengths but almost zero volume). For this reason, we use the dihedral angles within a tetrahedron to measure the quality of a tetrahedral mesh. Fig.8 shows the normalized histogram of the dihedral angles calculated from the entire tetrahedral meshes of the three examples we showed above. Due to the way by which *Tetgen* generates the tetrahedral meshes and removes the “slivers”, all three examples show almost the same distribution of the dihedral angles. Note that the numbers on the horizontal axis are given as follows: 1 for  $\{0^\circ - 5^\circ\}$ , 2 for  $\{5^\circ - 10^\circ\}$ , 3 to 9 for every  $10^\circ$  of  $\{10^\circ - 80^\circ\}$ , 10 for  $\{80^\circ - 110^\circ\}$ , 11 to 16 for every  $10^\circ$  of  $\{110^\circ - 170^\circ\}$ , 17 for  $\{170^\circ - 175^\circ\}$ , and 18 for  $\{175^\circ - 180^\circ\}$ .

### 4.2. Multilevel Mesh Generation and Multigrid-based Solvers

By coarsening the surface triangular meshes using different parameters  $T_0$  in equation 5, we can achieve a set of meshes with different levels of detail. Fig.9 illustrates the meshes of the human head data at three resolutions:  $T_0 = 0.001, 0.03$  and  $0.1$ . We can see that, as  $T_0$  increases, the overall mesh size decreases but the mesh points are selectively deleted so that important features (with high curvature) can be preserved.

Having access to a hierarchy of meshes can 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 multigrid algorithms developed for the Poisson equation in the 1970s [12,25]. 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 [26,13]:

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 the cost of finding the 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 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 [10] 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 resolutions 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 [10], we can calculate the restriction operator as follows:

$$I_f^c = \begin{pmatrix} \phi_1^c(\mathbf{v}_1) & \phi_1^c(\mathbf{v}_2) & \cdots & \phi_1^c(\mathbf{v}_{N_f}) \\ \phi_2^c(\mathbf{v}_1) & \phi_2^c(\mathbf{v}_2) & \cdots & \phi_2^c(\mathbf{v}_{N_f}) \\ \vdots & \vdots & \vdots & \vdots \\ \phi_{N_c}^c(\mathbf{v}_1) & \phi_{N_c}^c(\mathbf{v}_2) & \cdots & \phi_{N_c}^c(\mathbf{v}_{N_f}) \end{pmatrix}_{N_c \times N_f}, \quad (8)$$

where  $\phi_i^c(\mathbf{v}_j)$  is the  $i^{\text{th}}$  interpolation function, defined on the coarse mesh and valued at the  $j^{\text{th}}$  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 [8].

## 5. Conclusion

We presented a general mesh generation toolchain that has been demonstrated on a number of examples to produce smooth, adaptive, and high quality meshes with features well preserved. Our mesh generator consists of several surface mesh generation and

post-processing algorithms, including angle-based mesh quality improvement, adaptive feature-preserving mesh coarsening, and normal-based surface mesh smoothing. Tetrahedral meshes are constructed by constrained Delaunay triangulation as implemented in *Tetgen* [40,41]. Although this paper focused on the molecular mesh generation having the center and radius of each atom as inputs, our toolchain can also generate and process meshes from arbitrary scalar volumes (e.g., reconstructed 3D medical or cellular data) or low-quality surface meshes provided by users, as demonstrated in Fig. 6. One of our future directions is to extend the triangular and tetrahedral meshing toolchain presented here to other types such as quadrilateral/hexahedral meshes. We also plan to work on quality improvement of tetrahedral meshes, to further improve the tetrahedral meshes as generated by *Tetgen*.

The approaches described have been implemented in ANSI-C and incorporated into the Finite Element ToolKit (*FETK*) software suite [27] (<http://www.fetk.org>) as one of its major components, called *GAMER* (Geometry-preserving Adaptive MeshER). 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. Built on top of this highly portable software toolkit are two widely used application-oriented molecular modeling tools: the APBS [6] and the SMOL [16,43]. The *GAMER* together with other libraries in *FETK* will be made available in source form under the GNU Lesser General Public License.

**Acknowledgments** Z.Yu was supported in part by NSF Award 0411723, DOE Award DE-FG02-05ER25707, and NBCR (<http://nbcrc.sdsc.edu/>). M.Holst was supported in part by NSF Awards 0411723, 0511766, and 0225630, and DOE Awards DE-FG02-05ER25707 and DE-FG02-04ER25620. J.A.McCammon were supported in part by the NSF (MCB-0506593), NIH (GM31749), HHMI, CTBP, NBCR, W.M. Keck Foundation, and Accelrys, Inc.. We thank Chandrajit Bajaj for providing the head and heart data and Joel Stiles for the NMJ data.

## REFERENCES

1. M. Adams, Evaluation of three unstructured multigrid methods on 3D finite element problems in solid mechanics, *Int. J. Numer. Meth. Eng.* 55 (2002) 519-534.
2. M. Ainsworth, J.T. Oden, *A Posteriori Error Estimation in Finite Element Analysis*, Wiley-Interscience, 2000.
3. B. Aksoylu, S. Bond, M. Holst, An Odyssey into Local Refinement and Multilevel Preconditioning III: Implementation and Numerical Experiments, *SIAM J. Sci. Comput.* 25 (2003) 478-498.
4. B. Aksoylu, M. Holst, Optimality of multilevel preconditioners for local mesh refinement in three dimensions, *SIAM J. Numer. Anal.* 44 (2006) 1005-1025.
5. C.L. Bajaj, S. Goswami, Z. Yu, Y. Zhang, Y. Bazilevs, and T. Hughes, Patient specific heart models from high resolution CT, *Proceedings of International Symposium on Computational Modeling of Objects Represented in Images (CompIMAGE)*, Portugal, 2006.
6. N.A. Baker, D. Sept, S. Joseph, M. Holst, J.A. McCammon, Electrostatics of nanosys-

- tems: application to microtubules and the ribosome, *Proc. Natl. Acad. Sci.* 98 (2001) 10037-10041.
7. R. Bank, M. Holst, A New Paradigm for Parallel Adaptive Meshing Algorithms, *SIAM Review* 45 (2003) 291-323.
  8. Randolph E. Bank, R. Kent Smith, An Algebraic Multilevel Multigraph Algorithm, *SISC* 23 (2002) 1572-1592.
  9. H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, et al., The Protein Data Bank, *Nucleic Acids Research* 28 (2000) 235-242.
  10. M.L. Bittencourt, C.C. Douglas, R. A. Feijo, Non-nested multigrid methods for linear problems, *Numer. Meth. PDE* 17 (2001) 313-331.
  11. J.F. Blinn, A generalization of algebraic surface drawing, *ACM Transactions on Graphics* 1 (1982) 235-256.
  12. A. Brandt, Multi-level adaptive solutions to boundary-value problems, *Math. Comp.* 31 (1977) 333-390.
  13. W.L. Briggs, V.E. Henson, S.F. McCormick, A multigrid tutorial (2nd ed.), (Society for Industrial and Applied Mathematics, Philadelphia, PA), 2000.
  14. C.-Y. Chena, K.-Y. Cheng, A sharpness dependent filter for mesh smoothing, *Computer Aided Geometric Design* 22 (2005) 376-391.
  15. Y.H. Cheng, J.K. Suen, Z. Radić, S.D. Bond, M.J. Holst, J.A. McCammon, Continuum simulations of acetylcholine diffusion with reaction-determined boundaries in neuromuscular junction models, *Biophys. Chem.* 127 (2007) 129-139.
  16. Y.H. Cheng, J.K. Suen, D.Q. Zhang, S.D. Bond, Y.J. Zhang, Y.H. Song, N.A. Baker, C.L. Bajaj, M.J. Holst, J. A. McCammon, Finite element analysis of the time-dependent smoluchowski equation for acetylcholinesterase reaction rate calculations, *Biophys. J.* 92 (2007) 3397-3406.
  17. M.L. Connolly, Analytical molecular surface calculation, *J. Appl. Cryst.* 16 (1983) 548-558.
  18. M. Desbrun, M. Meyer, P. Schroder, A.H. Barr, Implicit fairing of irregular meshes using diffusion and curvature flow, *Proc. of SIGGRAPH'99* (1999) 317-324.
  19. T.J. Dolinsky, J.E. Nielsen, J.A. McCammon, N.A. Baker, Implicit fairing of irregular meshes using diffusion and curvature flow, *Nucleic Acids Research* 32 (2004) 665-667.
  20. B.S. Duncan, A.J. Olson, Shape analysis of molecular surfaces, *Biopolymers* 33 (1993) 231-238.
  21. S. Fleishman, I. Drori, D. Cohen-Or, Bilateral mesh denoising, *Proc. of SIGGRAPH'03* (2003) 950-953.
  22. J.-J. Fernandez, S. Li, An improved algorithm for anisotropic nonlinear diffusion for denoising cryo-tomograms, *Journal of Structural Biology* 144 (2003) 152-161.
  23. L.A. Freitag, C.F. Ollivier-Gooch, Tetrahedral Mesh Improvement Using Swapping and Smoothing, *Int. J. Numerical Methods in Engineering* 40 (1997) 3979-4002.
  24. J.A. Grant, B.T. Pickup, A Gaussian description of molecular shape, *J. Phys. Chem.* 99 (1995) 3503-3510.
  25. W. Hackbusch, A Fast Iterative Method solving Poisson's Equation in a General Region, *Proceedings of the Conference on the Numerical Treatment of Differential Equations (Lecture Notes in Mathematics, Springer-Verlag)* 631 (1978) 51-62.
  26. W. Hackbusch, Multi-grid Methods and Applications, *Multi-grid Methods and Appli-*

- cations (Springer-Verlag, Berlin, Germany), (1985).
27. M. Holst, Adaptive numerical treatment of elliptic systems on manifolds, *Advances in Computational Mathematics* 15 (2001) 139-191.
  28. K.H. Huebner, D. Dewhurst, D.E. Smith, T.G. Byrom, *The Finite Element Method for Engineers* (Wiley-IEEE), (2001).
  29. I. Jolliffe, *Principal Component Analysis* (Springer-Verlag, NY), (1986).
  30. T. Ju, F. Losasso, S. Schaefer, J. Warren, Dual contouring of hermite data, In *Proc. of SIGGRAPH* (2002) pages 339-346.
  31. B. Lee, F.M. Richards, The interpretation of protein structures: estimation of static accessibility, *J. Mol. Biol.* 55 (1971) 379-400.
  32. W.E. Lorensen, H. E. Cline, Marching cubes: a high resolution 3D surface construction algorithm, *Computer Graphics* 21 (1987) 163-169.
  33. M. Meyer, M. Desbrun, P. Schroder, A.H. Barr, Discrete differential-geometry operators for triangulated 2-manifolds, *Proc. of Visualization and Mathematics*, 2002.
  34. Y. Ohtake, A.G. Belyaev, I.A. Bogaevski, Mesh regularization and adaptive smoothing, *Computer-Aided Design* 33(4) (2001) 789-800.
  35. P.P. Pebay, T.J. Baker, Analysis of triangle quality measures, *Mathematics of Computation* 72(244) (2003) 1817-1839.
  36. P. Perona, J. Malik, Scale-space and edge detection using anisotropic diffusion, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 12 (1990) 629-639.
  37. F.M. Richards, Areas, volumes, packing, and protein structure, *Ann. Rev. Biophys. Bioeng.* 6 (1977) 151-156.
  38. M.F. Sanner, Python: a programming language for software integration and development, *J. Mol. Graphics Mod.* 17 (1999) 57-61.
  39. M.F. Sanner, A.J. Olson, J.C. Spehner, Reduced surface: an efficient way to compute molecular surfaces, *Biopolymers* 38 (1996) 305-320.
  40. H. Si, TetGen: a quality tetrahedral mesh generator and three-dimensional Delaunay triangulator, Technical Report 9, Weierstrass Institute for Applied Analysis and Stochastics (software downloading: <http://tetgen.berlios.de>), (2004).
  41. H. Si, K. Gartner, Meshing piecewise linear complexes by constrained delaunay tetrahedralizations, In *Proceedings of the 14th International Meshing Roundtable*, (2005).
  42. Y. Song, Y. Zhang, C.L. Bajaj, N.A. Baker, Continuum diffusion reaction rate calculations of wild type and mutant mouse acetylcholinesterase: adaptive finite element analysis, *Biophys. J.* 87 (2004) 1558-1566.
  43. Y. Song, Y. Zhang, T. Shen, C.L. Bajaj, J.A. McCammon, N.A. Baker, Finite element solution of the steady-state Smoluchowski equation for rate constant calculations, *Biophys. J.* 86 (2004) 2017-2029.
  44. G. Taubin, A signal processing approach to fair surface design, *Proc. of SIGGRAPH'95* (1995) 351-358.
  45. K. S. Tai, S. D. Bond, H. R. Macmillan, N. A. Baker, M. J. Holst, J. A. McCammon, Finite element simulations of acetylcholine diffusion in neuromuscular junctions, *Biophys. J.* 84 (2003) 2234-41.
  46. M. Totrov, R. Abagyan, The contour-buildup algorithm to calculate the analytical molecular surface, *Journal of Structural Biology* 116 (1996) 138-143.
  47. J. Weickert, *Anisotropic Diffusion In Image Processing*, ECMI Series (Teubner-Verlag,

- Stuttgart), (1998).
48. H. Xu, T.S. Newman, 2D FE Quad Mesh Smoothing via Angle-Based Optimization, In Proc., 5th Intl Conf. on Computational Science (2005) pages 9-16.
  49. H. Yagou, Y. Ohtake, A. Belyaev, Mesh Smoothing via Mean and Median Filtering Applied to Face Normals, Proceedings of the Geometric Modeling and Processing Theory and Applications (GMP'02) (2002) 124-131.
  50. I.T. Young, L.J. Vliet, Recursive implementation of the Gaussian filter, Signal Processing 44 (1995) 139-151.
  51. Z. Yu, C. Bajaj, A segmentation-free approach for skeletonization of gray-scale images via anisotropic vector diffusion, In Proc. Intl Conf. Computer Vision and Pattern Recognition (2004) pages 415-420.
  52. Z. Yu, 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 (2006) pages 2513-2516.
  53. Z. Yu, C. Bajaj, Computational Approaches for Automatic Structural Analysis of Large Bio-molecular Complexes, IEEE/ACM Transactions on Computational Biology and Bioinformatics (in press), (2007).
  54. Y. Zhang, C. Bajaj, 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 (2005) 5083-5106.
  55. Y. Zhang, G. Xu, 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 (2006) 510-530.
  56. T. Zhou, K. Shimada, An angle-based approach to two-dimensional mesh smoothing, In Proceedings of the Ninth International Meshing Roundtable (2000) pages 373-384.

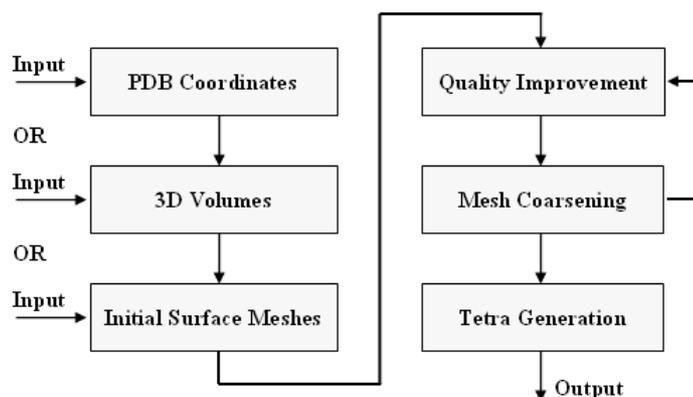


Figure 1. Illustration of our mesh generation toolchain.

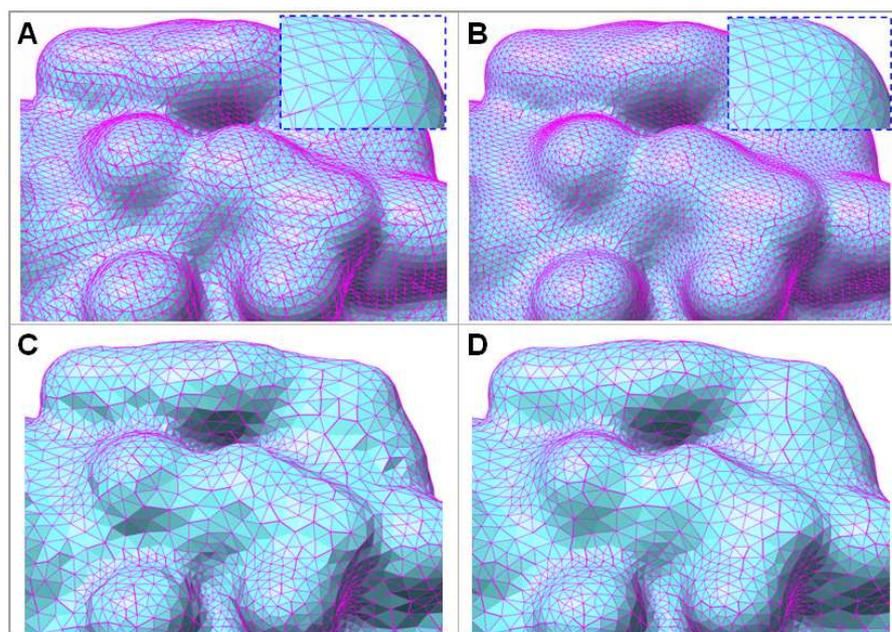


Figure 2. Illustration of the surface generation and post-processing. The PDB used is ICID with 1381 atoms. (A) The 3D volume is computed using the Gaussian blurring approach (Equation 1). Shown here is part of the molecular surface triangulation generated by the marching cube method. (B) The surface mesh after two iterations of quality improvement. The small windows shown in (A) and (B) are a closer look at the right-top corner of the mesh. (C) After coarsening ( $T_0 = 0.2$  in Equation 5), the mesh size is reduced by about seven times. (D) After normal-based surface smoothing (2 iterations).

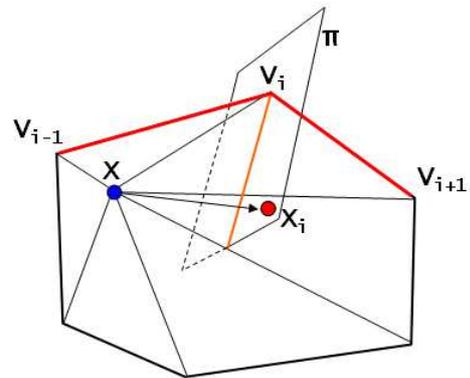


Figure 3. Illustration of the angle-based quality improvement algorithm.

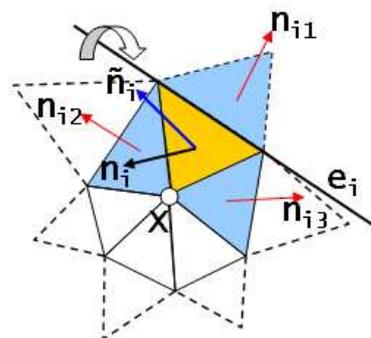


Figure 4. Illustration of normal-based surface mesh smoothing.

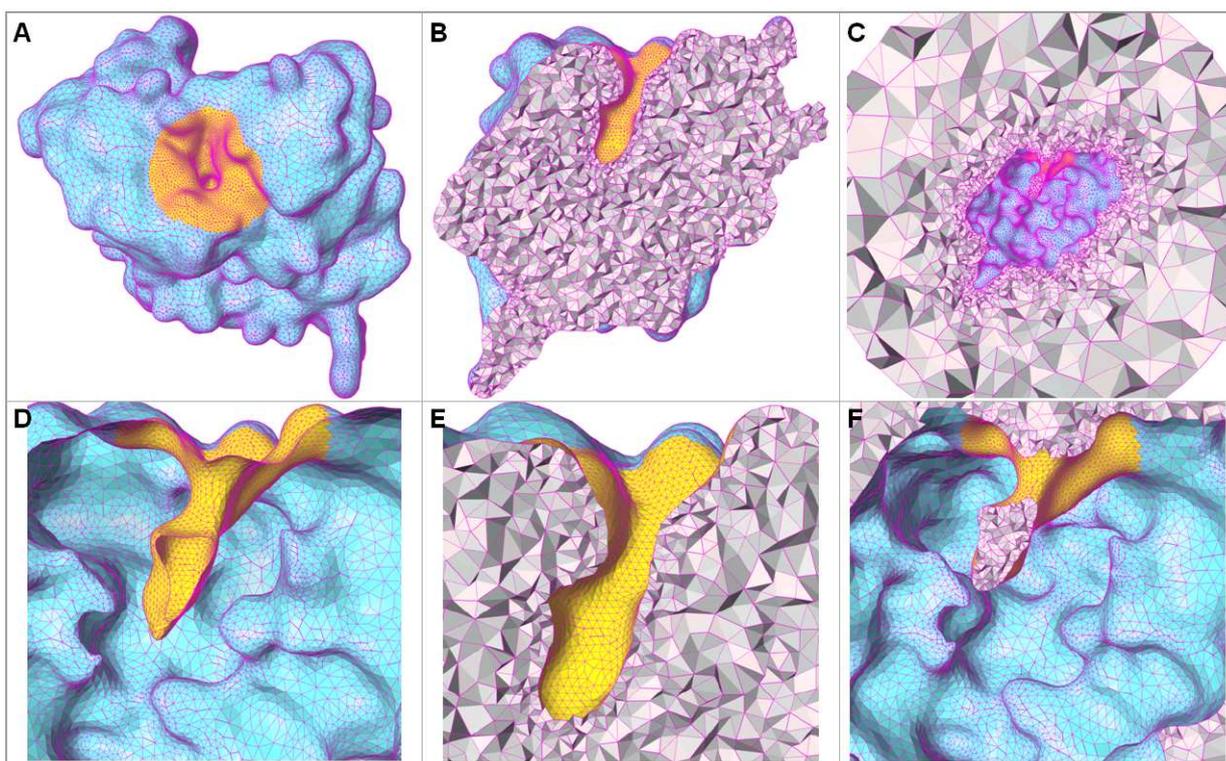


Figure 5. Illustration of mesh generation on the mAChE molecule. (A) A 3D volume is generated from the molecule's PQR file and then the surface mesh is constructed by the marching cube method and then processed by the methods described above. The region in yellow is the active site that has denser sampling rate than elsewhere. (B) and (C) show the cross sections of the interior and exterior tetrahedral meshes generated by *Tetgen* based on the surface mesh in (A). (D) An interior and closer look at the active site, rotated by  $90^\circ$ . (E) and (F) are closer views of (B) and (C), respectively, near the active site.

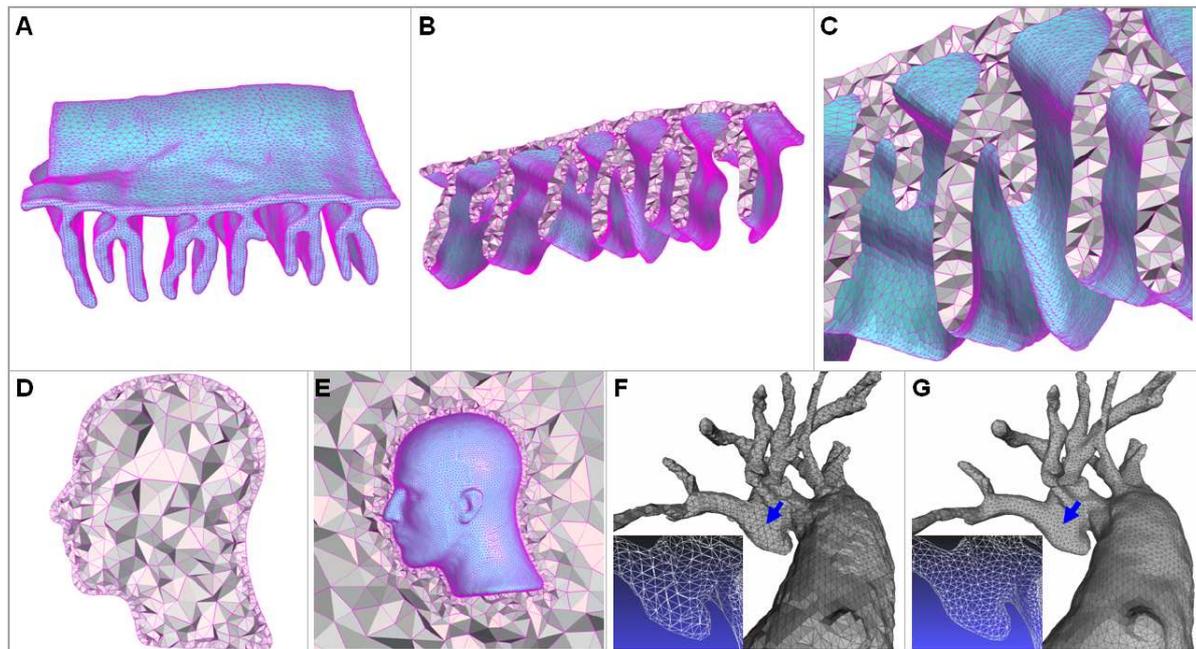


Figure 6. Illustration of mesh coarsening on other data at different scales. (A) The neuromuscular junction is given as a triangulated surface mesh. Shown here is the surface mesh after the post-processing by our methods. (B) The interior tetrahedral mesh is generated in the synaptic domain. (C) A closer look at the mesh around the third finger to the right. (D) and (E) The interior and exterior tetrahedral meshes are generated from the 3D volume of the human head model using our methods. No mesh coarsening is applied here due to the relatively small mesh size. (F) An example of human cardiovascular system. The mesh is generated directly from the segmented blood vessels using the marching cube method. The region indicated by arrow is enlarged and shown on the left-bottom. (G) Same as (F) but with mesh quality improvement and mesh smoothing by our methods.

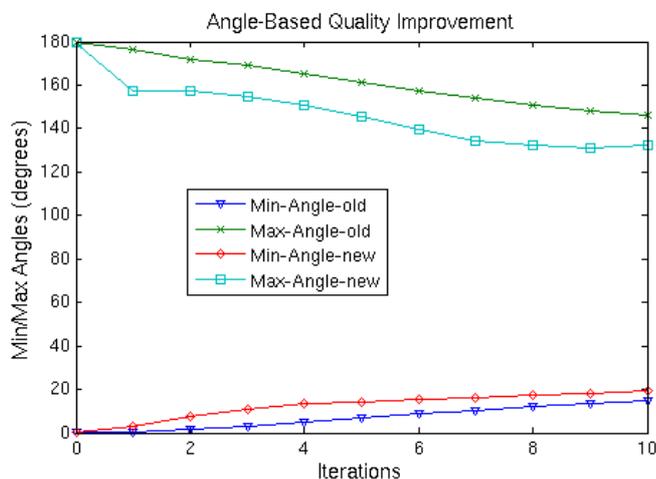


Figure 7. Comparisons between the original and modified angle-based methods for mesh quality improvement. The molecule being studied is mAChE monomer. Ten iterations of mesh smoothing are executed for both methods. The two curves on the bottom represent the minimal angles improved with respect to the 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.

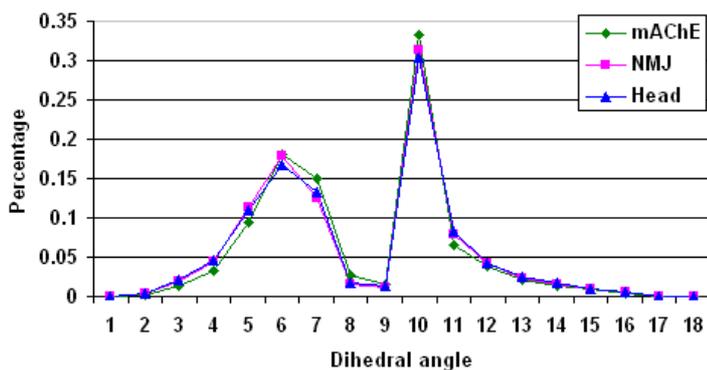


Figure 8. Quality analysis of our mesh generation toolchain. Three examples are considered: mAChE, NMJ, and Head. The vertical axis is the normalized histogram of the dihedral angles calculated from the entire tetrahedral meshes of the three examples. Due to the way by which *Tetgen* generates the tetrahedral meshes and removes the “slivers”, all three examples show almost the same distribution of the dihedral angles. See text for the correspondence between the numbers on the horizontal axis and the dihedral angles.

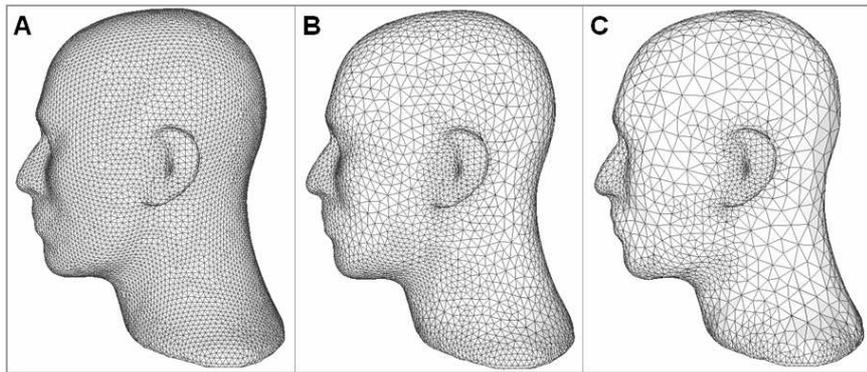


Figure 9. Hierarchical mesh generation by coarsening (experiments on the human head data). The parameter  $T_0$  is described in equation 5. The original surface mesh has 12,362 nodes and 24,720 triangles. With different coarsening parameters, we have: (A) Nodes = 12,012 and triangles = 24,020 for  $T_0 = 0.001$ . (B) Nodes = 5,513 and triangles = 11,022 for  $T_0 = 0.03$ . (C) Nodes = 3,242 and triangles = 6,480 for  $T_0 = 0.1$ .