# Tool Assisted Mesh Generation Based on a Tissue-Growth Model

A.V.Smirnov
West Virginia University
Morgantown, WV 26506-6106
http://smirnov.mae.wvu.edu/

August 1, 2002

**Abstract**

A heuristic mesh generation technique is proposed, which is based on the model of forced particle motion, edgewise cell-splitting algorithm, and a moving tool concept.

## 1 Background

There are three factors in mesh generation that are important from the user perspective: quality, speed and convenience. Usually mesh generation software tries to reach a compromise between the tree. When it comes to unstructured tetrahedral meshes, there is a wealth of methods to achieve a high mesh quality based on Delaunay tessellations [1, 5, 4]. Most of these use local mesh readjustments, by introducing extra vertexes and reassigning cell-node connectivities to preserve small aspect ratios in the mesh [2, 4]. This reassignment is a computationally intensive procedure, and the introduction of extra vertexes can lead to undesirable mesh non-uniformities. The criteria for mesh readjustment are usually given by circumradius-to-smallest-edge-length ratios and are designed to avoid small angles in the tetrahedra. However, these criteria are often not sufficient to completely avoid poorly shaped tetrahedra with extremely small

volumes [3]. Therefore, additional techniques are often used to improve the quality of the mesh, which leads to further slowdown of the mesh generation.

User convenience is determined by the complexity of operations required to define the 3D geometry, which is usually built from simpler planar elements. The complexity here arises from the need to specify the connectivities between these elements. Another problem is conforming the user defined 2D surface- and the generated 3D volume-meshes, which is not always easily solved.

In this study a heuristic approach to generate 3D tetrahedral meshes is proposed, based on the idea of a tool, an edge-wise cell-splitting, and a simple particle dynamics model.

# 2 Method

If a traditional unstructured mesh generation procedure can be compared to a crystal growth, where each new vertex is introduced on the top of the already assembled structure, thereby expanding the mesh-front, the current approach is based on on a tissue growth model. In this model each vertex of already existing mesh is experiencing pressure forces from inside the mesh. These forces can be repulsive forces from the inside vertexes, or the pressure forces of a hypothetical gas, filling-in the interior of the mesh. In either case the surface vertexes of the mesh will tend to move outwards, stretching the edges connecting them to the inside neighbors, whose positions are kept fixed (in another scenario the immediate inside neighbor vertexes are allowed to move as well). As the length of an edge reaches a certain threshold it is split in half and a new vertex is introduced in it's center (Fig.1). Following this, all cells containing that edge are split in half, with the new vertex becoming the common vertex for the newly formed cells around that edge. This edgewise splitting can occur for both internal and boundary edges, leading to a foam-like growth of the cells.

If the progress is allowed to proceed unconstrained it would lead to rather unpredictable shapes of the resulting mesh. To keep the situation under control a special *tool*-object is introduced, which is a single-connected convex domain, which in simplest case can have a spherical or a polyhedral shape. The tool encloses the growing part of the mesh. Only vertexes located inside the tool are allowed to move. Boundary vertexes

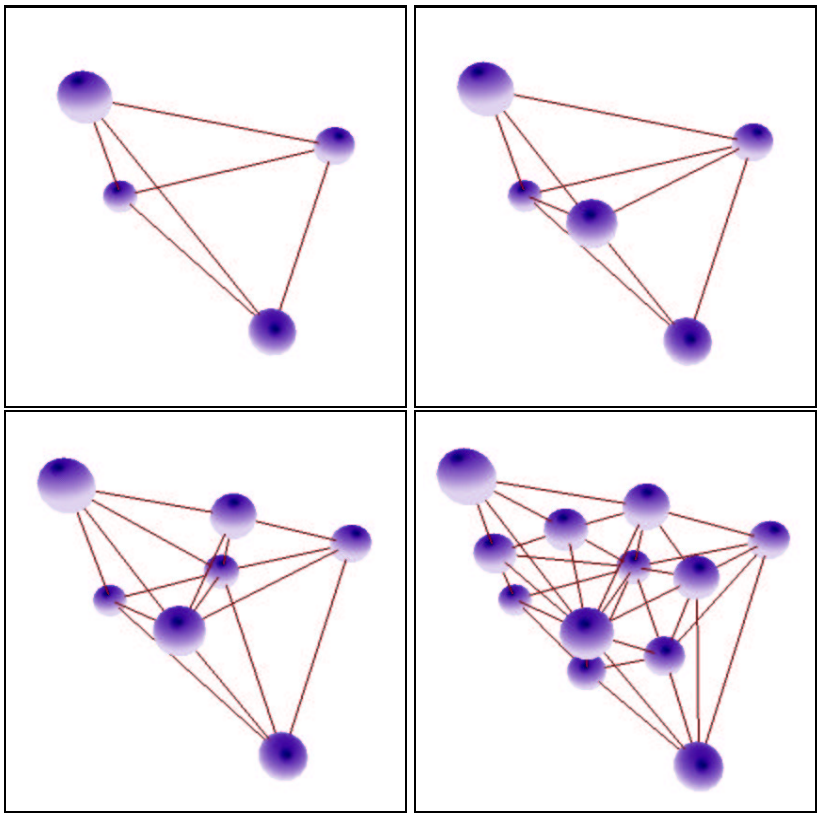

Figure 1: Cell-growth by edge splitting.

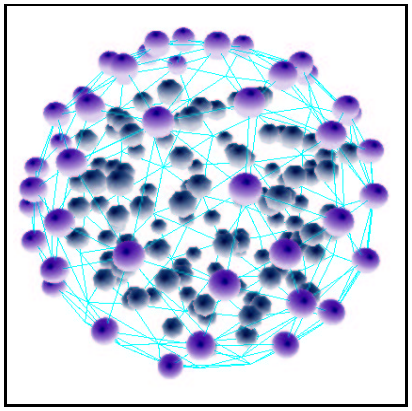

Figure 2: Dividing cells are filling-in the interior of a sphere.

of the mesh inside the tool experience attractive force directed toward the surface of the tool (alternatively it could be a repulsive force acting from some point inside the tool). In this way the local mesh boundary takes the shape of the tool, as more boundary cells are generated while the edges stretch and new vertexes are introduced into the mesh (Fig. 2).

In order to prevent the formation of badly shaped tetrahedra the process is accompanied by a periodic elliptic smoothing procedure, when each vortex is placed at the center-of-mass of the enclosing polyhedron. This procedure is similar to that used in FEM assembly operation and involves looping over the cells to calculate cell-volumes and assemble them into nodal-volumes. Another loop through the vertexes is then required to move them to the new positions. The smoother is applied only to the vertexes within a certain relaxation radius from the tool. The relaxation region can be slightly bigger than the radius of the tool, so that sufficiently many mesh-nodes are relaxed to new positions. A similar elliptic smoother is applied to the vertexes that reach the surface of the tool. Here a special care should be taken that the vertexes actually remain on the tool-surface while they are relaxed. After the relaxation is done the tool is moved to a new position.

If the motion of the tool is incremental, so that each new position is inside the previous area enclosed by the tool, the new vertex and cell lists belonging to the tool can be compiled locally without searching through all the mesh. It can be done with a recursive procedure, which would require

a cell-cell connectivity information, i.e. a list of cell-neighbors for each cell. This information is automatically recorded during cell splitting. In this way the algorithm is made strictly local, in that the speed of execution does not depend on the size of already assembled mesh, and all the operations are confined to the area inside the tool.

The length of the edge-splitting threshold is selected as a fraction of a tool-size. This enables an easy control of the mesh refinement level and sets the limit on the minimum and maximum cell sizes in the mesh. Together with the elliptic smoother, this produces a rather uniform mesh.

Because the mesh in this process grows from inside-out it rarely experiences a conformation problem of joining with other parts of the mesh inside the domain or at the boundaries, which is often the case with surface-inward mesh generation techniques. Nevertheless, mechanisms of using a *sewing*-tool to patch different parts of the mesh can be implemented. Another useful option is to use a *destructive* tool, which will remove cells from already existing mesh.

A further advantage of this method is that it does not require specification of two dimensional surfaces in order to produce the mesh. Instead it requires procedures for tool selection, orientation and motion. Tool motion, being one-dimensional, can be easily programmed in a separate tool-script file. This file contains a one dimensional skeleton of the geometry and the scenario of tool shapes, motions and transformations along the skeleton. This information is substantially simpler to encode than the 2D surface geometry of the domain. Another possibility is to integrate the tool control into a user interface and use it for manual tool manipulation. In this case the mesh generation will be accomplished in a way similar to conventional 2D drawing packages. This will provide a more intuitive and simpler user interface than the one based on surface generation and patching.

The example in figure 3 shows the surface edges of a tetrahedral mesh, representing bifurcating blood vessels or airways. The mesh was generated using a spherical tool moving along prescribed paths with tool-radius changing along the way.

# 3 Conclusions

The proposed tool-assisted mesh generation technique simplifies mesh generation and has a potential for creating more intuitive user interfaces.

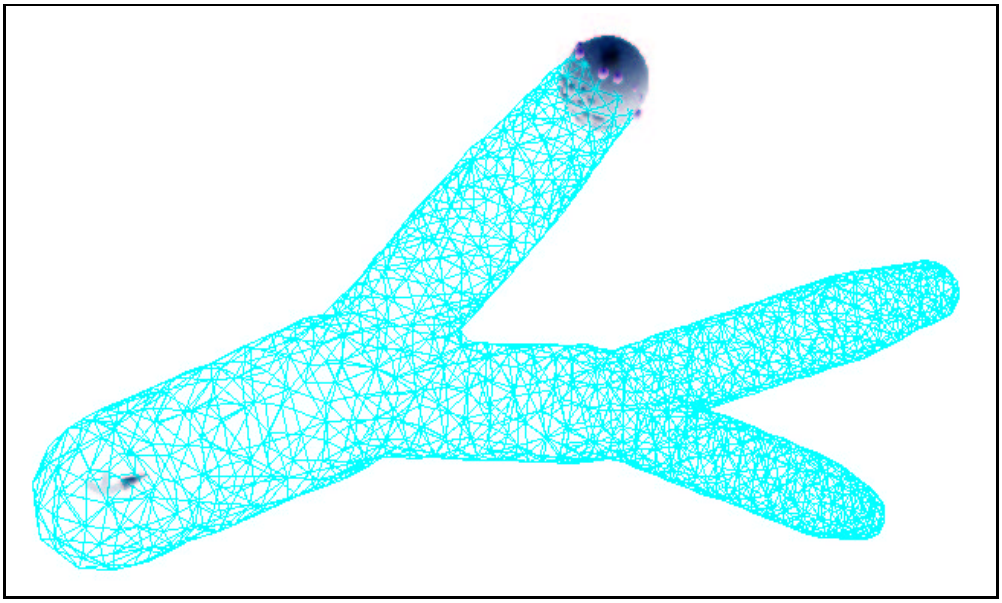

Figure 3: Meshing a bifurcating duct with a spherical tool.

# References


[1] T.D. Dey, C.L. Bajaj, and K. Sugihara. On good triangulations in three dimensions. *International Journal of Computational Geometry*, 2(1):75–95, 1992.

[2] W.H. Frey. Selective refinement: A new strategy for automatic node placement in graded triangular meshes. *International Journal for Numerical Methods in Engineering*, 24(11):2183–2200, 1987.

[3] J.R. Shewchuk. Tetrahedral mesh generation by delaunay refinement. In *Fourteenth Annual Symposium on Computational Geometry*, pages 86–95, Minneapolis, Minnesota, 1998. Association for Computing Machinery.

[4] D.F. Watson. Computing the n-dimensional Delaunay Tessellation with Application to Voronoi Polytopes. *Computer Journal*, 24(2):167–172, 1981.

[5] N.P. Weatherill. Delaunay triangulation in computational fluid dynamics. *Computers in Mathematics with Applications*, 24(5/6):129–150, 1992.