# Parallel Unstructured Mesh Generation by an Advancing Front Method

**Yasushi Ito, Alan M. Shih, Anil K. Erukala, and Bharat K. Soni**
Dept. of Mechanical Engineering,
University of Alabama at Birmingham, U.S.A.
*yito@uab.edu, ashih@uab.edu, anile@uab.edu, bsoni@uab.edu*

**Andrey N. Chernikov and Nikos P. Chrisochoides**
Dept. of Computer Science,
College of William and Mary, Williamsburg, VA, U.S.A.
*ancher@cs.wm.edu, nikos@cs.wm.edu*

**Kazuhiro Nakahashi**
Dept. of Aerospace Engineering,
Tohoku University, Sendai, Japan
*naka@ad.mech.tohoku.ac.jp*

**Abstract**

Mesh generation is a critical step in high fidelity computational simulations. High quality and high density meshes are required to accurately capture the complex physical phenomena. A parallel framework has been developed to generate large-scale meshes in a short period of time. A coarse volume mesh is generated first to provide the basis of block interfaces and partitioned into a number of sub-domains using METIS partitioning algorithms. A volume mesh is generated on each sub-domain in parallel using an advancing front method. Dynamic load balancing is achieved by evenly distributing work among the processors. All the sub-domains are combined to create a single volume mesh. The combined volume mesh can be smoothed to remove the artifacts in the interfaces between sub-domains. A void region is defined inside each sub-domain to reduce the data points during the smoothing operation. The scalability of the parallel mesh generation is evaluated to quantify the improvement.

**Keywords:** unstructured mesh generation; parallel; advancing front method

# 1. Introduction

Mesh generation is an essential step in computational simulations requiring high quality meshes to accurately capture the complex physical phenomena. Today's computational systems enable us to solve a problem with a mesh having tens of millions of nodes. Generating such a mesh on a single machine, however, is a challenging task in view of Central Processing Unit (CPU) time as well as memory requirements. Generating meshes in a parallel environment reduces huge amount of time required for large scale mesh generation.

Advancing front [11, 7] and Delaunay triangulation [2, 12] methods are widely used for generating tetrahedral meshes sequentially as well as in parallel. In parallel mesh generation, the domain to be meshed is usually decomposed into a number of sub-domains, and a mesh is generated on each sub-domain in parallel. The starting point of parallel mesh generation can be a surface mesh or a volume mesh. Löhner et al. [10] present a Parallel Advanced Front Technique (PAFT) for distributed memory machines. The method subdivides each submesh into an interior region and interface regions. Each processor reads a submesh and refines the interior region independently. Once all the interior regions are refined the processors refine the interface regions, and finally the corners. It appears from [10] that the order in which the interior and interface regions are refined does impact the performance and code complexity. The processors synchronize locally, because no new elements can be inserted in the interface and corner regions before the meshing of adjacent interior and interface regions, respectively.

Löhner [9] revisits the parallel AFT. An octree is employed in order to decompose the continuous geometry and maximize code re-use for parallelizing the AFT on shared memory computers. The new PAFT refines the interior octants only along the active front of the AFT. The octree is partitioned along the active front by partitioning the terminal octants. Clusters of terminal octants that maximize data locality are processed independently. Therefore, portions of the active front can be processed concurrently. The new PAFT meshes the domain in phases. At each phase of the AFT the active front expands and a new one is created. Then the octree is refined and re-partitioned again. This process continues until the whole domain is meshed. The PAFT synchronizes at the beginning of each phase in order to sequentially refine and re-partition the global octree, for the new active front, whose terminal octants will be refined in parallel. The PAFT method is suitable for shared memory machines, but it is inefficient on large-scale distributed memory parallel platforms for two reasons: 1) the global synchronization required at the beginning of each phase, and 2) the data movement for gathering (into a single node) and for scattering (back to all nodes) the global octree before and after its sequential refinement and re-partition.

De Cougny et al. [3] propose a Parallel Octree AFT (POAFT). The POAFT method, first, generates a distributed coarse-grain octree using a divide-and-conquer algorithm. The terminal octants and the geometric model of a domain define the sub-domains. The terminal octants of the octree are classified into

interior, interface, boundary, and complete. Interface octants have at least one adjacent octant which is not local. Boundary octants include mesh entities from the input surface mesh while complete octants contain no front faces. Before the boundary octants are refined and meshed a re-partitioning might take place if necessary. The meshing of boundary octants is a challenging task. Every processor applies a tree-based face removal procedure in order to connect the input surface mesh with the volume mesh of the interior octants. The face removal (from the active front) is a basic operation in AFT and is used to connect a front-face to a vertex which is drawn from a "neighborhood" of the front-face. The parallel face removal portion of the "neighborhood" might be on a remote processor and a target vertex cannot be found locally; in this case the face removal is postponed. This will create unmeshed regions between the terminal interface boundary octants and input surface mesh. The POAFT re-partitions the active terminal interface boundary octants and face removal is applied until there are no active front faces. This process is repeated until there are no unmeshed regions.

Both methods have the same two features. One is that the sub-domain limits or internal boundaries (in other words, interfaces between sub-domains) are specified beforehand so that no extra communication is required during the mesh generation in parallel. If a node is added on an interface, the information of the node should be transferred to the processor that is in charge of the neighboring sub-domain. This can result in heavy communication overhead. The other is that the starting point of the parallel mesh generation is a surface mesh. This feature is preferred to generate huge meshes in a few steps. Their parallel performance, however, may be sacrificed when the domain defined is divided into a number of sub-domains because there is no explicit information for interfaces of the sub-domains.

Computational simulations sometimes require several meshes for the same geometry and domain in the different sizes. Parallel mesh generation starting from a coarse volume mesh can be an alternative because this approach enables robust domain decomposition. In addition, the communication can be minimized between processors during the parallel mesh generation because all the sub-domains can be easily defined beforehand. In this approach, the surface meshes for the decomposed sub-domains are extracted and refined. After that, a tetrahedral mesh generation method is applied for each sub-domain.

In this paper, we propose an unstructured parallel mesh generation approach using an AFT starting with a coarse volume mesh. The steps involved in this framework are as follows: 1) prepare a coarse volume mesh to provide the basis of block interfaces, 2) perform domain decomposition, 3) extract boundary data as a surface mesh for each sub-domain and refine all the surface meshes, 4) generate a tetrahedral volume mesh for each sub-domain, 5) combine volume meshes, and smooth the elements near the artificial interfaces created in Step 3 as an option. METIS [8] is used for domain decomposition based on the evaluation of several domain partitioners [1]. During the parallel mesh generation, Message Passing Interface (MPI) [4] is used for inter-process communication. An advancing front method is used for the tetrahedral mesh generation to eas-

ily preserve the connectivity of the surface boundaries [7].

# 2. Parallel Mesh Generation

## 2.1. Creation of Sub-domains

In our approach, a coarse mesh is prepared as the starting point of the parallel mesh generation. For the efficient execution of computational simulations on high performance parallel computers, a good quality partitioning algorithm for tetrahedral meshes is required. Good quality partitions are nothing but partitions with almost equal number of mesh elements and a smaller number of faces on the interfaces so that the inter-processor communication required to perform the information interchange between adjacent elements is minimized. To achieve this, METIS, a software package based on a multilevel graph partitioning method, is employed.

METIS decomposes the original coarse volume mesh into a user-specified number of sub-domains $M$. Boundary data for the sub-domains can be extracted easily in the form of surface meshes. Note that boundary triangles on the interface between two sub-domains and boundary edges shared by more than three sub-domains should be identified in this stage to prevent the creation of duplicated faces. The boundary data have two types of faces: ones on the physical surface boundaries and the others on interface boundaries. A surface mesh extracted from a sub-domains defined by METIS sometimes contains edges shared by more than four triangles, which also should be identified.

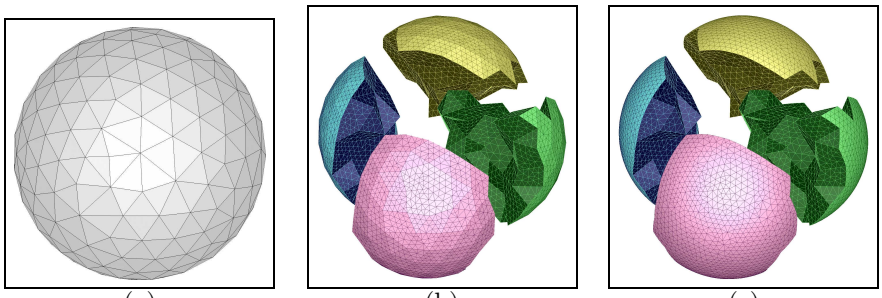
Basically, each edge of the surface meshes is divided by a certain number $n_d$, which controls the size of the final volume mesh. Too small edges, however, become much smaller edges if they exist and if every edge is divided by the same number. This situation should be avoided. At each node $i$, calculate the average length of connected edges $\bar{l}_i$. Since each edge $j$ has two nodes $i_1$ and $i_2$, the number of division can be calculated as follows:

$$n_{dj} = \frac{2n_d l_j}{\bar{l}_{i_1} + \bar{l}_{i_2}}, \tag{1}$$

where $l_j$ is the length of edge $j$. Each triangle is then subdivided using a Delaunay triangulation method in 2D as shown in Figure 1.

The fidelity of surface meshes to the original surface boundaries is often essential for numerical simulations. The nodes on the surface boundaries should be interpolated using a higher order method, such as a quadratic triangle shape function, based on the original coarse mesh or the initial geometry [5] (Figure 1c). Triangles on the surface boundaries are refined in 2D first and the locations of new nodes are interpolated using a higher order method to avoid creating invalid triangles. Then triangles on the internal boundaries are refined.

Because the mesh decomposition using METIS and the surface refinement by the Delaunay triangulation method do not take much time, only the master processor does all the tasks, while the slave processors are in blocked state.



(a) (b) (c)

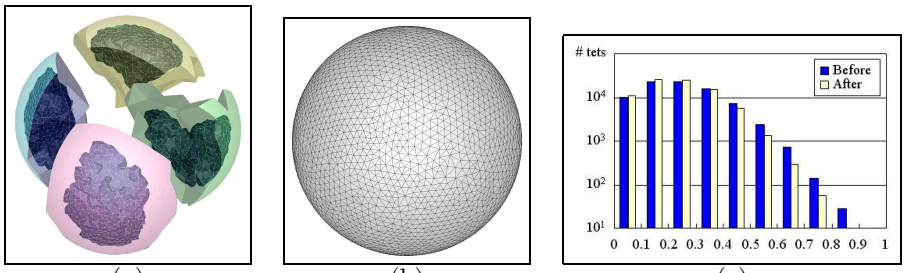
Figure 1: Sub-domains for a sphere: (a) an initial coarse tetrahedral mesh (1,457 tetrahedra); (b) refined surface meshes on sub-domains ($M = 4$, $n_d = 5$); (c) interpolation of nodes on the surface boundaries based on a quadratic triangle shape function

Volume mesh generation is performed in parallel using the refined surface meshes for the sub-domains using an AFT [7]. Sub-domain boundary data are stored on the hard disc drive once, and the master processor distributes loads to the slave processors using the MPI library. Although METIS creates fairly balanced sub-domains, the following estimated load of sub-domain $i$, which is the estimated number of tetrahedra to be created, is considered to ensure that the loads are almost equally distributed to each processor.

$$L_i = \frac{V_i n_{f_i}}{S_i}, \tag{2}$$

where $V_i$, $S_i$ and $n_{f_i}$ denote the volume of sub-domain $i$, the area and number of triangles of the boundary of sub-domain $i$, respectively. The sub-domains should be sorted in the ascending order of $L_i$, e.g., sub-domain 1 has the smallest $L$. Assume that we have $M$ sub-domains and $N$ processors. $M \gg N$ because the runtime required to generate a volume mesh using an AFT is generally proportional to at least $n \log n$ due to geometric search required in the process, where $n$ is the number of nodes or tetrahedra to be generated. Processor $p$ ($= 1$ to $N$) receives sub-domains $p, 2N - p + 1, p + 2N, 4N - p + 1, p + 4N, \ldots$

The barrier type of synchronization is used to achieve synchronization of the processors. In this synchronization method, each process performs its work until it reaches the barrier. It then waits for the other processes to finish their tasks. When the last process reaches the barrier, all processes are synchronized. After that all the processes are automatically released to continue their work. The master node performs synchronous communication operations to communicate with slave processes. Since the AFT preserves the connectivity of the boundary data, extra communication between the processors is not required during this parallel meshing process. The tetrahedral meshing method ensures the quality of resulting meshes by employing an angle-based node smoothing method [6] and face swapping based on the Delaunay property.




(a) (b) (c)

Figure 2: Mesh generation for a sphere: (a) Voids defined inside sub-domains to minimize required data size; (b) Final volume mesh (82,424 tetrahedra); (c) Skewness of tetrahedra in the sphere meshes before and after applying the angle-based node smoothing method

### 2.2. Combining volume meshes for all the sub-domains

After the generation of tetrahedral meshes for all the sub-domains, they should be glued if their size is not huge to create a single volume mesh as shown in Figure 2. If the internal boundary data are kept in the stage of the boundary refinement, the same nodes are identified by integer indexes, not by $xyz$ coordinates, which enables robust and fast search for the same nodes.

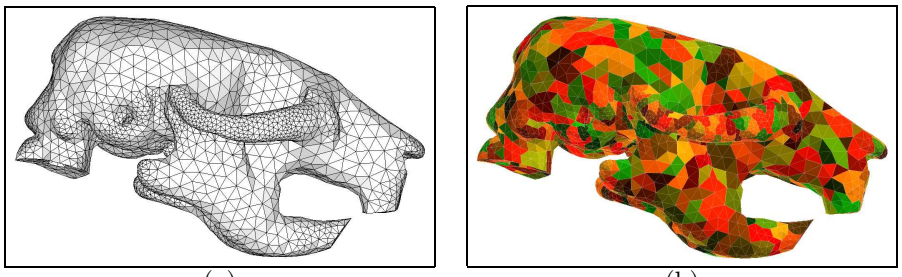
The tetrahedral elements, however, may not be in shape near the interfaces artificially defined in Section 2.1. Their quality can be improved using the same quality enhancement methods implemented in the tetrahedral mesh generator. We only apply the angle-based node smoothing method in this stage to minimize computational time. Since the elements far from the interfaces have already been smoothed enough during the stage of the volume mesh generation in parallel, they can be removed in this process as shown in Figure 2b to minimize the data required to input. The elements less than the 2nd level from the interfaces (zero level) are smoothed. The quality of the mesh before and after this local smoothing approach is shown in Figure 2c. The skewness of tetrahedron $i$ is defined as follows:

$$\varepsilon_i = \frac{V_{i\,\mathrm{opt}} - V_i}{V_{i\,\mathrm{opt}}}, \tag{3}$$

where $V_i$ is the volume of tetrahedron $i$ and $V_{i\,\mathrm{opt}}$ is the volume of an equilateral tetrahedron having a circumsphere with the same radius. If $\varepsilon_i$ is zero, the tetrahedron is equilateral.

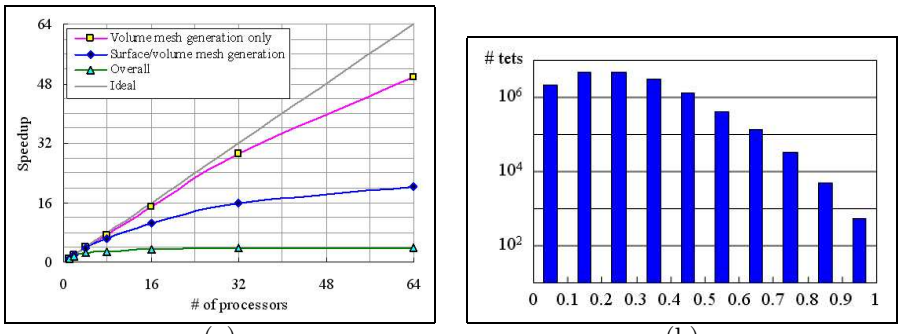
## 3. Applications

In this section, two examples are shown. For generating these meshes, a 128-processor IBM Linux cluster is used, each node of which has Dual Xeon 2.4GHz processors.



(a) (b)

Figure 3: A mouse skull model: (a) an initial coarse mesh; (b) sub-domains defined by METIS




(a) (b)

Figure 4: A mouse skull model: (a) speedups obtained for the generation of a volume mesh; (b) skewness of the tetrahedra

## 3.1. Mouse Skull Model

Figure 3 shows an initial volume mesh for a mouse skull model based on computed tomography (CT) data, which has 14,614 nodes and 67,900 tetrahedra. To demonstrate the scalability, the same finer volume mesh is created using the different numbers of processors as shown in Figure 4a. The original coarse volume mesh is decomposed into 2,048 sub-domains in each case using METIS (Figure 3b), and $n_d$ is set as 8. The final fine volume mesh has 2,863,992 nodes and 16,361,376 tetrahedra without applying the quality enhancement methods near the interface regions since the quality of the mesh is acceptable as shown in Figure 4b. In the sequential case, the required time for each process is 633 seconds for the surface refinement, 16,558 seconds (4.6 hours) for the volume mesh generation for the 2,048 sub-domains, and 6,599 seconds (1.8 hours) for gluing all the sub-domains and validating the quality of the final mesh.

The volume mesh generation in parallel achieves excellent scalability. The overall performance becomes worse when the number of processors is increased because only the master processor handles the surface refinement and the binding of the sub-domain meshes. The size of the final volume mesh is small in

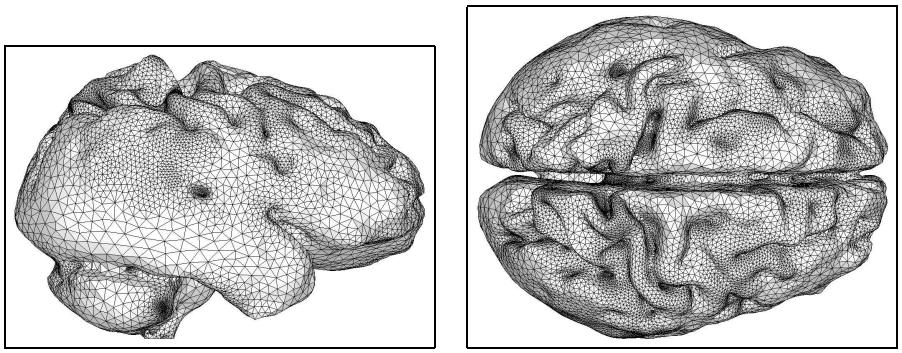


Figure 5: Side and top views of an initial mesh for a human brain model

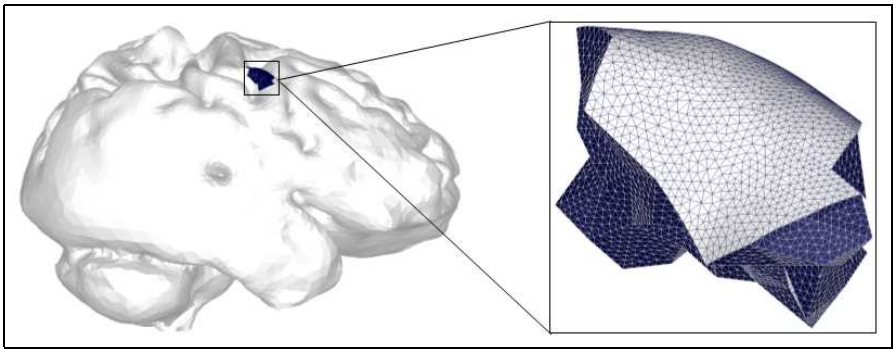


Figure 6: One of 2,048 sub-domains of a human brain model

this case, and the required time for the surface refinement is relatively high. The surface refinement in parallel, however, may be required. Since numerical simulations based on a huge mesh usually require the domain decomposition beforehand, the meshes for sub-domains can be kept separately.

## 3.2. Human Brain Model

Figure 5 illustrates an initial volume mesh for a human brain model based on CT data, which has 123,114 nodes and 652,464 tetrahedra. The volume mesh is divided into 2,048 sub-domains, one of which is shown in Figure 6 ($n_d = 8$). 64 processors in 32 nodes are used to generate a finer mesh. The overall runtime is 115 minutes. Figure 7 shows the overall performance of our mesh generation method. In this case, the final mesh, which has 22 million nodes and 132 million tetrahedra, is stored separately as the sub-domains. The volume mesh generation in parallel achieves excellent load balancing. However, the overall system including the subdivision of the surface boundaries does not work very well. This is due to the data transfer through the hard disc drive. Although only the master processor is in charge of the domain decomposition

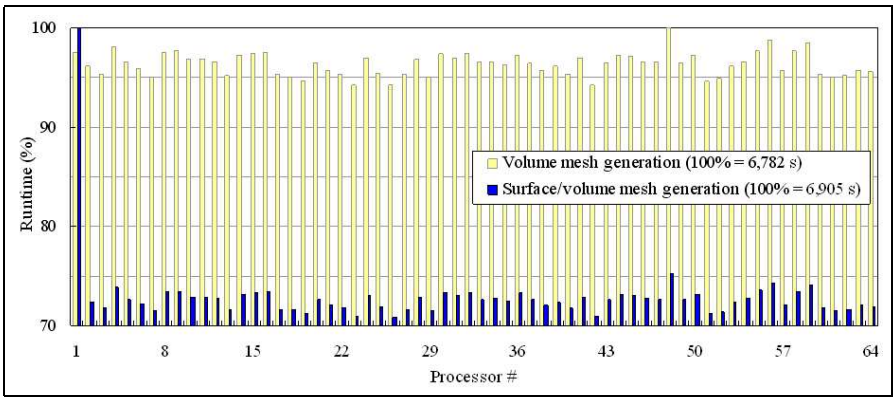



Figure 7: Overall performance in the human brain case ($N = 64$)

and the surface refinement at the first stage of the parallel mesh generation, the required computational time is short (less than 5 minutes in this case). When the master processor finishes the surface refinement, the information on the surface boundaries is stored on the hard disc drive once, and then each slave processors starts to create volume meshes. To avoid writing on the hard disc drive, the surface mesh refinement in parallel may also be required.

# 4. Conclusions

A parallel environment for large scale mesh generation is implemented. A coarse volume mesh is generated to provide the basis of block interfaces and it is decomposed into a number of sub-domains using METIS partitioning algorithms. The surface meshes of the sub-domains are extracted and refined using a Delaunay method in 2D. To achieve the geometric fidelity, the locations of nodes on the surface boundaries can be calculated using a second order interpolation method. A mesh is generated on each sub-domain in parallel using an advance front method, and then all the sub-domains are combined to create a single tetrahedral mesh. To remove the artifacts near the interface regions due to the domain decomposition, an angle-based node smoothing method can be applied as an option. A void region is defined in each sub-domain to reduce the data points during the process. Excellent load balance is achieved based on the number of faces, area and volume of each sub-domain. With the parallel mesh generation approach, turnaround time for the mesh generation process is significantly reduced and high quality meshes are obtained.

# 5. Acknowledgments

This research is supported in part by the NASA URETI Program No. NCC3-994, and the NSF ITR Adaptive Software Project No. ACI-0085969.

# References


[1] J. Antonio and I. Paz. Evaluation of parallel domain decomposition algorithms. In *1st National Computer Science Encounter, Workshop of Distributed and Parallel Systems*, pages 44–50, 1997.

[2] T. J. Baker. Shape reconstruction and volume meshing for complex solids. *Int. J. Numer. Meth. Eng.*, 32:665–675, 1991.

[3] H. L. De Cougny and M. S. Shephard. Parallel refinement and coarsening of tetrahedral meshes. *Int. J. Numer. Meth. Eng.*, 46:1101–1125, 1999.

[4] W. Gropp, E. Lusk, and A. Skjellum. *Portable Parallel Programming with the Message Passing Interface*. The MIT Press, 1994.

[5] Y. Ito and K. Nakahashi. Direct surface triangulation using stereolithography data. *AIAA J.*, 40:490–496, 2002.

[6] Y. Ito and K. Nakahashi. Improvements in the reliability and quality of unstructured hybrid mesh generation. *International Journal for Numerical Methods in Fluids*, 45:79–108, 2004.

[7] Y. Ito, A. M. Shih, and B. K. Soni. Reliable isotropic tetrahedral mesh generation based on an advancing front method. In *13th International Meshing Roundtable*, pages 95–105, 2004.

[8] G. Karypis and V. Kumar. *MeTiS: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices. Version 4.0*. Department of Computer Science, University of Minnesota, Sept. 1998.

[9] R. Löhner. A parallel advancing front grid generation scheme. *Int. J. Numer. Meth. Eng.*, 51:663–678, 2001.

[10] R. Löhner, J. Camberos, and M. Marshal. *Unstructured Scientific Computation on Scalable Multiprocessors (Eds. Piyush Mehrotra and Joel Saltz)*, chapter Parallel Unstructured Grid Generation, pages 31–64. MIT Press, 1990.

[11] R. Löhner and P. Parikh. Generation of three-dimensional unstructured grids by the advancing front method. *International Journal for Numerical Methods in Fluids*, 8:1135–1149, 1988.

[12] N. P. Weatherill and O. Hassan. Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints. *Int. J. Numer. Meth. Eng.*, 37:2005–2039, 1994.