Tetrahedral Discretization of Complex Volumetric Spaces: Implementation, Efficiency, Robustness and Interactive Control

Ashwini Patgawkar, Dinesh Shikhare, Satyashree Mahapatra, S. Gopalsamy and **S. P. Mudur**
National Centre for Software Technology,
Gulmohar Cross Rd. 9, Juhu, Mumbai 400049, India.
{ashwini,dinesh,satya,gopal,mudur}@ncst.ernet.in

Abstract

The problem of tetrahedral grid generation within volumes bounded by triangulated surfaces has received considerable attention in recent years due its significance in CFD *analysis dealing with complex geometric bodies. Tetrahedral discretization of volume is complex, and even more so when the bounding surfaces are complex geometric configurations of intersecting doubly curved surfaces, say, represented as* NURBS*, as is usually the case for aircraft surfaces. In addition, the* CFD *analyst has to be provided with both quantitative and qualitative controls over the grid generation process in terms of the number of tetrahedral elements to be generated, their size, shape, local density, variation and so on. The implementation related issues of efficiency, robustness, scalability and interactive control are thus understandably hard to handle. In this paper, we describe these issues in detail and present the solutions we have implemented in our grid generation system* – VolGrid.

1. Introduction

The computational fluid dynamic (CFD) analysis by finite element methods of 3D volumetric spaces, like the volume surrounding an aircraft, rely on the availability of volume grid generation tools. These tools enable rapid discretization of a given volume into a set of simple 3D volume elements, like tetrahedra or hexahedra. The motivation for discretization is that the simple elements are easy to process in a uniform way – the methods for processing complex entities do not change with a change in the geometry. The complexity of the analysis process increases only in a combinatoric fashion when a more complex geometric configurations are considered.

The CFD analysis codes have specific requirements of the grids both in quantative and qualitative terms. Typically this would include control over the number of elements, their size, their shape, their local density, smoothness in size variation etc. It is well known that grid quality affects both efficiency and accuracy of CFD solutions [4]. Providing explicit parameterized control over these qualitative and quantitative aspects of grid generation process is essential.

Apart from efficient algorithms for volume grid generation with the above requirements, the other issue is to provide a good 3D interface for defining/specifying the geometric surface configurations that make up the volumetric space of interest, for discretization, grid visualization and analysis. There are challenging problems in visualization of 3D geometries currently addressed by many researchers [5].

In this paper, we first discuss the important issues that need to be tackled when providing an efficient and user friendly volume grid generation tool and present how we have addressed them in our volume grid generation system VolGrid.

2. Tetrahedral Grid Generation Problem

Given a bounded volume defined in terms of oriented triangulated surfaces, the volume discretization problem is to decompose the volume into tetrahedral elements, such that the union of these elements spans the enclosed volume and intersection of any two tetrahedra is either null or a vertex or an edge or a triangular face shared between them. Such a grid or mesh is also required to retain the original points and their the connectivity on the boundary of the volume, and also not introduce any additional points on the boundary.

The quality of the grid is specified in terms of desirable aspect ratio of tetrahedral elements and smoothness of volume gradients along any line considered within the bounded region. The density of points along the boundary of the volume provides the primary point distribution control for the grid. For example, a dense distribution of points on some part of boundary surface will cause dense distribution of small tetrahedra near that region, and the density should gradually dilate while moving to a boundary having less dense distribution of points. In addition, a cluster of points can always be introduced in the interior region of the volume to get the desired density of points anywhere in the region of interest.

A tetrahedral grid generation method can be evaluated based on the following criteria:

- **Robustness:** A method should be able to discretize a volume, bounded by a surface configuration of any shape.

- **User control over grid quality:** The method must easily incorporate user controllable parameters for grid quality by explicit specification of the quality measures.

- **Control over clustering:** user specified clustering of points in the volume should be achievable.
- **Speed:** The method should be fast even for large number of points, i.e. the order of the algorithm should be a polynomial of a small degree.
- **Capacity to handle large data sets:** The data structure should be usable for large input data sets.
- **Parallelism:** A parallelisable algorithm is highly desirable due to ubiquity of multi-processor machines.

2.1. Selection of tetrahedral grid generation method

Many different tetrahedral grid generation techniques have been reported and surveyed in the literature [3]. The most popular methods in recent literature are advancing front method [9, 10], sphere/bubble packing method [12] and Delaunay-based methods [6, 15]. While the sphere-packing method is probably the most robust and models the desired qualities of grids explicitly in its formulation, its efficiency largely depends on the final optimization phase. For large grids it is bound to be slow. Advancing front method [9] relies on heuristics for introducing interior points, which fail in many special cases. A lot of special handling is required for a robust implementation of advancing front algorithm. Our experience has shown that in aircraft geometry such special special cases often arise. We have chosen a Delaunay-based method because it scores very high in its robustness. For any set of points a Delaunay tetrahedral descretization exists. The algorithm does not rely on too many heuristics for insertion and selection of interior points.

2.2. Associated Tools for Grid Generation

In a usable system, the basic grid generation algorithm must be supported by various tools for effective usage in the complete design cycle of a system such as an aircraft. The issues involved in building such a complementary suite of tools are listed below:

Data Preparation for Grid Generation: Surface grid generation tools generate surface triangulations on trimmed surface patches of complex geometries. These individual triangulated surface patches must be used to composite a single volume. Many geometry correction and "repair" problems need to be addressed.

Grid Analysis: The generated grid must be analysed for qualitative and quantitative requirements of CFD analysis system. Analysis capabilities must be integrated into grid generation code to refine the grid to obtain grid of desirable quality. Statistical analysis of the grids are often useful to determine overall quality of grids.

Grid Visualization: Various inspection techniques such as planar sections, sweeping planes, visiting specific regions in volumes are required besides quantitative analysis.

3. Basic Delaunay-based Algorithm

In this section we outline the basic Delaunay-based tetrahedral discretization algorithm in 3D [13]. Later sections will highlight the specific issues of efficiency, robustness and quality.

3.1. Delaunay Criterion in 3D

An n-dimensional domain can be decomposed systematically into a set of packed convex polyhedra. For a set of points $\{P_i\}, i = 1 \ldots N$, regions V_i can be assigned to P_i, such that V_i represents the space closer to P_i,

$$V_i = \{P : |P - P_i| < |P - P_j| \ \forall j \neq i\}. \qquad (1)$$

These regions are called Voronoi regions.

If the points sharing a common boundary in the Voronoi diagram are joined, then the result is Delaunay triangulation of the convex hull of the set of points $\{P_i\}, i = 1 \ldots N$. A Voronoi vertex is equidistant from the points which form the polyhedron and it is found to be the circumcentre of the polyhedron. Thus, if a circumsphere of a polyhedron is constructed, no other point will be contained in it.

3.2. Steps involved in 3D Triangulation

1. Input the boundary points $\{P_i\}, i = 1 \ldots N$, and the boundary point connectivities of the facets $\{F_j\}, j = 1 \ldots M$ i.e, the surface grid, which encloses a volume.

2. Compute the bounding box of the input points and construct a super-hexahedron out of it. This super-hexahedron is then split into five tetrahedra to give the initial triangulation for the Delaunay algorithm.

3. Insert the boundary points to obtain the tetrahedral discretization,$\{T_k\}, k = 1 \ldots K$, of the given set of points.

4. Creation of field grid points.

5. Ensure that the surface triangulation is contained in the volume triangulation. Recover missing facets, if any.

6. Identify the tetrahedra outside the domain of interest and remove them from the grid.

7. Remove the points added on the boundary, during the edge and facet recovery.

8. Smooth the grid.

4. Concerns and Issues

We first identify the implementation issues and then state the solutions adapted in our implementation.

Integrity issues: The input to the Delaunay algorithm should be a closed surface grid, which may or may not contain concavities. An essential requirement is that the grid should be boundary conformal. Also as mentioned earlier, the boundary facets should be undisturbed. During the point insertion process, local retetrahedralisation must not introduce intersecting tetrahedra or almost flat tetrahedra. Nor should there be any cavities.

Efficiency issues: The time critical steps mainly include the following:
(a) The point insertion stage is the most time-consuming one in the grid generation process, as it involves search for all those tetrahedra in the existing grid, that violate the Delaunay criteria.
(b) Connectivity queries need to be done during the local refinement for, say, determining the tetrahedra connected to a vertex or finding edges sharing a vertex, and so on.
Hence search and traversals must be efficiently carried out.

Quality issues: The minimum dihedral angles at the edges of tetrahedra, must be maximised to get a high quality mesh.

Adaptivity issues: Capability to adapt and refine grids in specific regions where CFD solver needs dense grids is necessary. Refinement is usually specified in terms of points to be inserted in an existing grid.

5. Data Structures

Accelleration techniques are needed for operations like querying for a point or set of points lying in a specified spatial neighbourhood, queries about topological neighbourhood and sharing information between vertices, edges, triangles, tetrahedra and meshes. VolGrid uses special data-structures and consistency checks at various points in the algorithm. The following data-structures have been implemented with substantial benefits:

Octree This spatial data-structure [11] is adopted to speed up queries about spatial neighbourhood. Insertion of every new point in Delaunay-based algorithm requires determination of a list of tetrahedra in the neighbourhood whose circumspheres include the point. This search is accelerated by the use of an octree of inserted grid points. The octree inherently has a tendency to occupy large space, and the growth must be arrested. We restrict its growth upto a predetermined depth, and leaf nodes at that depth are used as buckets for linear search. The choice of depth of octree is determined by space-speed tradeoff.

Radial-edge Data-structure All entities in VolGrid are constructed out of simplices. A 0-simplex is a vertex identified over a geometric point in 3D; a 1-simplex is an edge between two vertices; a 2-simplex is a facet constructed using three edges; and a 3-simplex is a tetrahedron having four facets (with 6 edges and 4 vertices).

Surface-grid is constructed using a collection of facets forming a manifold surface. A volume is composed using a collection of facets defining a closed manifold geometry. A volume-grid is a collection of tetrahedra spanning a given volume.

The organization of topological entities is as per Figure 1. The data structure implemented has been adapted from the work reported by Muuss *et. al* [8] and Bruzzone [2]. Entities are classified as top-level entities, component entities, and auxiliary entities. Vertex, edge, facet (triangle) and tetrahedron are termed as components entities. Top-level entites are surface-grid, volume and volume-grid are called top-level entities since the user is exposed only to these entites. The auxiliary entities are data-structures used only by internal algorithms.

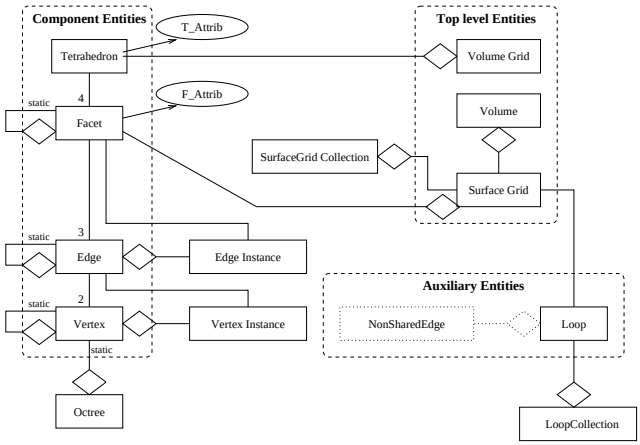


Figure 1. Topology Captured in Radial-edge Data-structure

Vertex is the only entity that directly refers to a geometric entity called Point. The other higher-level entities only refer to each other using pointers. We first examine what relationships exist in surface-grids and volume-grids, and also note the cardinality of each of the relationships.

The aim is to capture the connectivity relationships among vertices and edges, edges and facets, facets and tetrahedra, vertex and facets, vertex and tetrahedra, and edge and tetrahedra.

Note that the constant cardinality relationships can be trivially captured using fixed number of pointers to the associated entities. The relationships such as: “edges connected to a vertex” and “facets connected to an edge” are modeled using special entities called VertexInstance and EdgeInstance. A vertex keeps a list of VertexInstances, each keeping a pointer to an edge using the vertex, thus allowing a query such as “which edges are connected to a given vertex?” Similarly, an edge keeps a list of EdgeInstances, each keeping a pointer to the facet using that edge. This mechanism allows us to capture the other variable cardinality relationships like “which facets are connected to a vertex?” and “which tetrahedra are connected to an edge?” and so on.

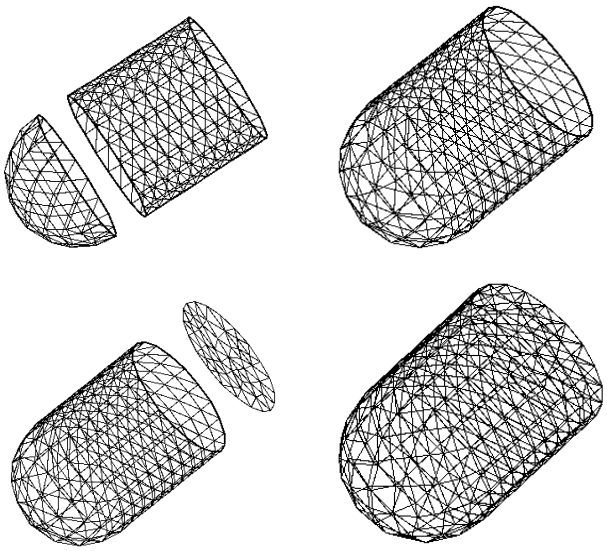


Figure 2. Composing a volume from surface triangulations.

6. Volume Data Preparation

The method of grid generation inside a volume requires a well defined volume bounded by one or more surfaces. VolGrid accepts bounding surfaces as surface grids. For example, to prepare a volume definition of an aircraft surface in an enclosure such as a huge sphere, the following procedure may be followed:

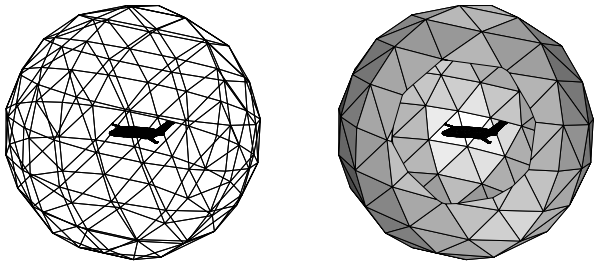


Figure 3. Aircraft placed inside a large sphere. The aircraft surface forms the inner boundary and the sphere forms the outer boundary of the volume

1. The parts of aircraft surface, represented by surface grids generated in a grid generation package, can be loaded independently and a single surface grid could be generated by merging these parts along their boundaries (see Figure 2). The gaps in the geometry should then be closed by forcing a local merging of geometry or by local triangulation.

2. Such a connected component then may be placed inside a large enclosure such as sphere or cylinder with closed ends, or any other appropriate shape. Tools for generating primitive shapes such as triangulated boxes, spheres, cylinders, hemispheres and ellipsoid are provided (see Figure 3).

3. The elements of the surface grids should be consistently oriented such that the normals of the elements point away from the domain of interest for grid generation.

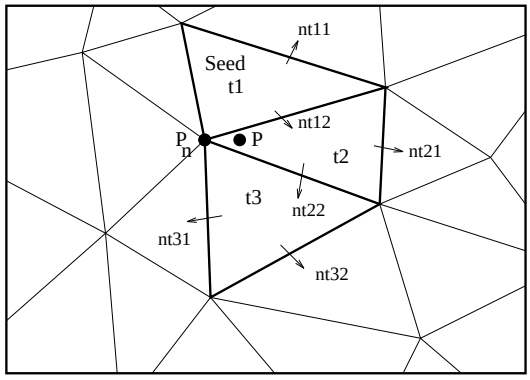


Figure 4. Point insertion

7. Implementation in VolGrid

7.1. Insertion of points

The Delaunay algorithm starts with the insertion of all the boundary points. To initiate the process, as already mentioned a super hexahedron split into five tetrahedra is created and then the boundary points are inserted iteratively. When a new point is to be inserted in the existing triangulation, the set of tetrahedra whose circumspheres contain the point, are found.

$$||x_c - x_n|| < r \tag{2}$$

x_c - coordinates of circumcentre

x_n - coordinates of point to be inserted

r - circumradius of the tetrahedron

These tetrahedra violate the Delaunay criterion and hence are removed to form a cavity, bounded by the facets, all of which are visible to the new point. These facets are then connected with the point to form the new tetrahedralisation.

Figure 4 illustrates the boundary point insertion stage, with a 2D analogy. The detailed explanation is as follows:

1. P is the point to be inserted in the grid. The nearest point, P_n is searched for, in the octree.

2. Query all the tetrahedra sharing P_n.

3. A tetrahedron, t_1, is found, whose circumsphere contains P. t_1 forms the seed for the recursive algorithm to obtain the cavity.

4. Visit neighbours of t_1 to find all other tetrahedra, whose circumspheres contain P. Here, t_1, t_2, t_3 form the

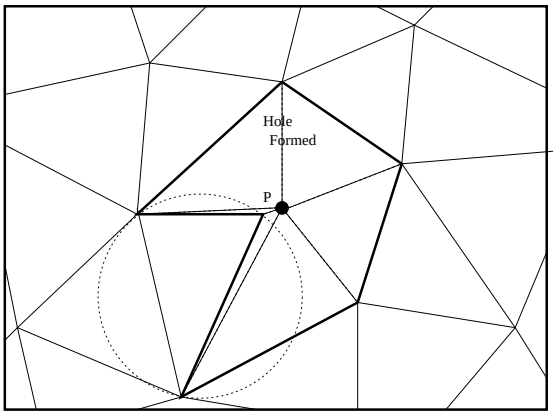


Figure 5. Creation of a flat tetrahedra due to the position of point along the plane of the facet

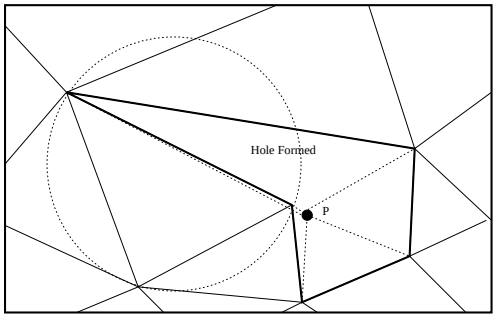


Figure 6. Creation of intersecting tetrahedra due to the position of point just below the plane of the facet

tetrahedra to be deleted, leaving behind a polyhedron, bounded by facets.

5. Connect P to the facets of this polyhedron and validate the new tetrahedralisation formed.

Validity and Integrity Checks

During the point insertion phase, not all points can be inserted into the grid due to various reasons. This could be due to floating point accuracy problems, or due to the internal geometry structure, that is formed in the triangulation process. The circumsphere check is a floating point operation and may result into flat or intersecting tetrahedra.

In Figure 5, the point, P is to be inserted in the grid. It lies almost along the plane of one of the facets of the polyhedral hole, that is formed by deleting all the tetrahedra, whose circumspheres contain P. This facet cannot be deleted as P lies just outside the circum-sphere of the tetrahedron containing it.

Another case, which needs to be handled is shown in Figure 6. The geometry of the hole may be star-shaped and the point, P may lie such that it may not be visible to one of the facets of the hole. This may lead to intersecting tetrahedra.

A point may lie very close to a facet, however the tetrahedron

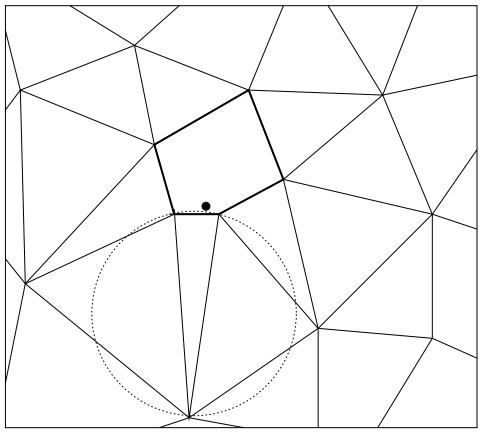


Figure 7. Creation of a flat tetrahedra due to the position of point just above the circumsphere of the tetrahedron

containing this facet may not get deleted because of its large circumsphere as shown in Figure 7.

In order to avoid such inconsistencies, certain checks need to be made to ensure a valid grid, during the local retetrahedralisation. These include the following:

1. Volume of a tetrahedron should not be zero (Figures 5 and 7).
2. Sum of the volumes of tetrahedra to be deleted should be equal to the sum of the volumes of new tetrahedra to be formed (Figure 6).
3. To avoid intersecting tetrahedra, a check should be done to find the location of the new point with respect to the facet, with which it forms the tetrahedron, i.e, whether point lies on/above/below the facet (Figure 6).

The points that cannot be inserted into the grid due to reasons discussed above, are kept aside till all other points are inserted. As the internal geometry gets modified with every point insertion, it is possible that the points thus set aside get included later. Hence, an attempt is made to insert these points again. A few points could still remain that cannot be inserted in the grid.

To facilitate the insertion of these "hard-to-insert" boundary points, it is necessary to modify the geometry around them. Hence, a dummy point is created in the vicinity of the non-inserted boundary points, which will disturb the local tetrahedral structure and may allow point insertion. The position of this point is at the centroid of the neighbourhood of the point to be inserted.This dummy point, not being part of the boundary, is removed from the grid at a later stage.

7.2. Creation of field points

The insertion of points in the interior is based on Weatherill and Hassan's approach [15]. The boundary of the domain is defined by the points and their connectivities, which reflect

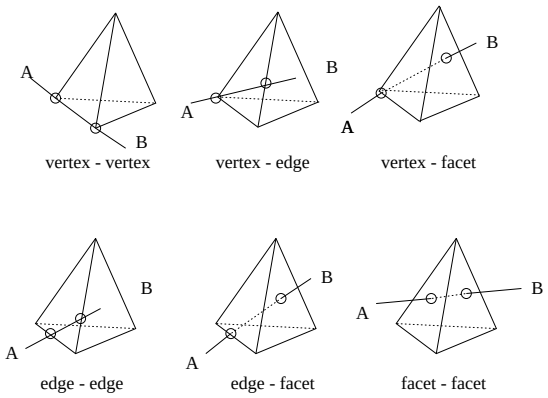


Figure 8. Possible cases of intersection of the missing edge with the tetrahedra

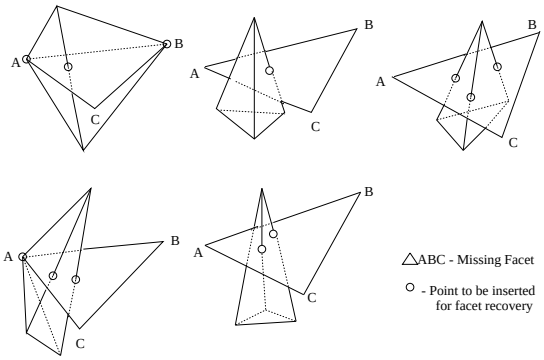


Figure 9. Penetration of tetrahedra through the missing facet

the appropriate variations in the surface slope and curvature. The field point creation method should ensure that the boundary point distribution is extended into the domain in a spatially smooth manner.

In this method a Point Distribution Function (PDF) is defined for each point as the average of the lengths of the connected edges.

$$dp_o = \frac{1}{M} \sum_{1}^{M} |r_i - r_o| \tag{3}$$

Any point which is placed within this distance forms a bad triangulation.

A point is inserted at the centroid of every tetrahedron T_j, if

$$d_m > \alpha dp_q \tag{4}$$

where,

d_m	distance of the point from the vertex forming the tetrahedron
dp_q	PDF of the centroid, obtained by interpolating the PDFs of the four vertices of the tetrahedron

If the point satisfies the Eq. 4, then it is inserted into the domain provided,

$$s_j > \beta dp_m \tag{5}$$

where,

s_j	distance of the point from other valid centroids
dp_m	PDF of the point
$alpha$	controls the gridpoint density by changing the allowable shape of the formed tetrahedra
β	influences on the regularity of the triangulation by not allowing points within a specified distance of each other to be inserted in the field

7.3. Boundary Conformity

Delaunay algorithm, by its property, triangulates the given set of points into its convex hull domain and hence, the boundaries of the input volume are not always preserved in the volume grid. In order to ensure that the boundary point connectivities of the surface grid exist in the volume grid too, it is necessary that:

1. all the boundary points be present in the volume grid (This is ensured in the point insertion stage.),
2. all the boundary edges exist in the volume grid,
3. the boundary facets are contained in the volume grid.

Some of these missing facets and hence, the edges, can be retreived by the edge swapping technique. Also, it is found that the absence of some of the boundary edges or facets is mainly due to the penetration of tetrahedra through the surface boundary. A few of these cases are shown in the figures 8 and 9. These can be recovered by inserting points at the intersections of these penetrating tetrahedra and the missing boundary edges/facets, as explained in the subsequent sections.

Some researchers, for example [1], have implemented this step immediately after the boundary point insertion stage and then headed for field point creation. However, there are certain cases, as shown in Figure 10, where, the field point insertion may spill the tetrahedra outside the boundary, thus, arising a need to put constraints on the field point insertion or redoing the boundary conformity step. To avoid this, we implement the boundary conformity stage after the insertion of field points. The various techniques which are carried out for edge/facet recovery in grid generation are stated below:

- Point insertion at the midpoint of the missing edge. This method was tried out, but edge/facet recovery is not always guaranteed, as the connections are done using Delaunay criterion. Also, this involves insertion of too many points [14].
- Only swapping of edges (as reported in [1]).
- Addition of points along the intersections of the missing elements and the grid elements. Here the added points are directly connected to the grid elements [15]. The implementation of this technique is discussed here, in the following section.

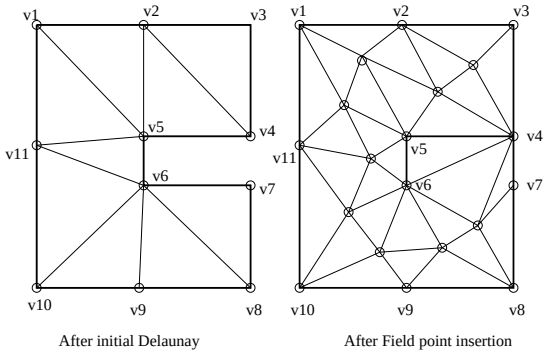


Figure 10. Spilling of tetrahedra outside boundary after field point insertion

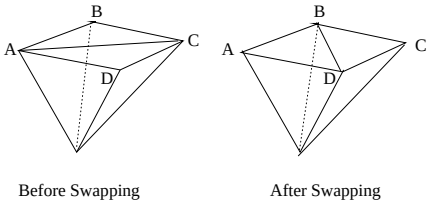


Figure 11. Face swapping

Edge Swapping

It is possible to make the boundary facets in the volume grid to be coincident with the boundary faces in the surface grid by swapping some edges in the boundary surface triangulation.

Edge swapping is not usually preferred as it alters the surface triangulation, which is not desired, as it may lead to a mismatch in case of merging of two volume grids. In this case, the swapped facets can be tagged and later, reverted back to the original geometry after the removal of exterior tetrahedra. Swapping of facets in the volume grid is an alternate solution, as shown in Figure 11. Here, the triangles ABD and BCD are a part of the the surface grid but, ACD and ABC exist in the volume grid. AC can be replaced by BD in the surface triangulation. Swapping helps in reducing a great amount of work to be carried out in the recovery of missing edges and facets.

Edge Recovery

After applying the edge swapping technique, there may be a few edges and hence, the facets, which may be missing in the volume grid. These can be recovered by adding points at the intersection of these with the tetrahedra. An edge may intersect multiple tetrahedra, hence,the intersection routine should be fast enough to determine all the intersection points and the positions of these points with respect to the tetrahedron. An edge may intersect the tetrahedron in various ways as shown in the Figures 8

The algorithm to compute all the intersection points for a missing edge E_{1-2} is as follows:

1. Find all the tetrahedra sharing vertex v_1 of E_{1-2}.

2. Find the first tetrahedron, t, which intersects E_{1-2}, either along the edge or along the facet or vertex.

3. If the intersection point, p_i, is a vertex then that means a part of the edge is already present in the grid and it is tagged as boundary edge. if the intersection vertex is v_2 of E_{1-2}, then the edge recovery is complete. Else, if it is a facet or an edge, then all the tetrahedra sharing this facet or edge are removed and the intersection point is connected to the facets of the polyhedron that remains after the tetrahedra deletion.

4. If $p_i \neq v_2$, then $v_1 = p_i$, and steps 1-3 are repeated.

The above steps are repeated until all the missing edges are recovered.

To facilitate the determination of intersection points, the sub-entities of a tetrahedron are given status numbers (see table). The intersection routine takes the start point and its status as input parameters and computes the intersection point and its status. Thus,the type of connections to be done, can be easily determined with the help of status numbers, i.e, if the point of intersection lies along the edge, then the tetrahedra sharing the edge need to be deleted. With every intersection, the edge is partially or fully recovered.

The edge starting from any of the vertex or edge or facet of the tetrahedron always intersects the facet opposite to it. And the resulting intersection is a vertex or edge of this opposite facet. Hence, a mapping of each of these sub-entities with its opposite facet is stored in the table. eg., $f(v_1 - v_2 - v_3)$ is the facet opposite to v_0 as shown in the following table:

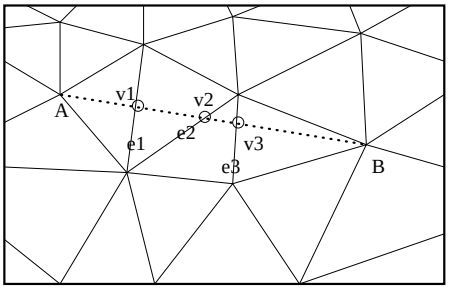
Element	Status	No. of Opposite facets	Opposite Face status
v_0	0	1	13
v_1	1	1	12
v_2	2	1	11
v_3	3	1	10
$e(v_0 - v_1)$	4	2	12, 13
$e(v_0 - v_2)$	5	2	11, 13
$e(v_0 - v_3)$	6	2	10, 13
$e(v_1 - v_2)$	7	2	11, 12
$e(v_1 - v_3)$	8	2	10, 12
$e(v_2 - v_3)$	9	2	10, 11
$f(v_0 - v_1 - v_2)$	10	3	11, 12, 13
$f(v_0 - v_1 - v_3)$	11	3	10, 12, 13
$f(v_0 - v_2 - v_3)$	12	3	10, 11, 13
$f(v_1 - v_2 - v_3)$	13	3	10, 11, 12

Boundary Facet Recovery

The presence of all the boundary edges does not imply the presence of all boundary facets in the volume grid. Some of these facets may be missing or may be present in the grid in the split form. The missing facets can be recovered in a similar way, as the missing edges, by identifying the tetrahedra which intersect it and then by adding the points at the intersection of the two.

The recovery mechanism goes as follows:

1. Query the tetrahedra connected to one of the vertices of the missing facet, f_i.



AB - Edge to be recovered

v1, v2, v3 - Points added on the Boundary for edge recovery.

e1, e2, e3 - Edges which intersect the edge AB

Figure 12. Insertion of points for edge recovery

2. Find the first tetrahedron,t which intersects f_i, either along the edge or in the interior of f_i.
3. Query the edges of t, and find the edges intersecting f_i. If the edges of t intersect at the vertex then, the facet formed by these vertices is found and tagged as a boundary facet. Else, intersection point is computed and inserted into the grid by making simple connections.
4. Visit the neighbour of t. Go to step 3.

The steps 2 and 3 are repeated till the complete facet is recovered.

7.4. Removal of tetrahedra outside the domain of interest

Since the delaunay algorithm tesselates in the convex hull of the given set of points, there are tetrahedra which are generated outside the domain of interest. These exterior tetrahedra include those in the concavities of the object and also in the region between the object boundary and the boundary of the super hexahedra. These should be identified and removed from the grid to obtain the tetrahedrulated version of the given volume.

Here, we make use of the orientation of the facets to identify the extraneous tetrahedra. The boundary facets are assumed to be oriented such that their normals point away from the object. Another property of the boundary facets is that the tetrahedra share-count should be one.

One of the two tetrahedra, sharing the same boundary facet is exterior to the domain. A first such external tetrahedron, t_e, is determined using facet orientation, which forms the seed for the removal algorithm. The others can then be found by walking through the neighbours of t_e recursively. A given volume may have more than one concavity, hence the above step needs to be repeated till all the boundary facets have a tetrahedra share-count of one.

7.5. Removal of added boundary points

The points, added along the boundary, during the boundary recovery stage, are not a part of the given input boundary,

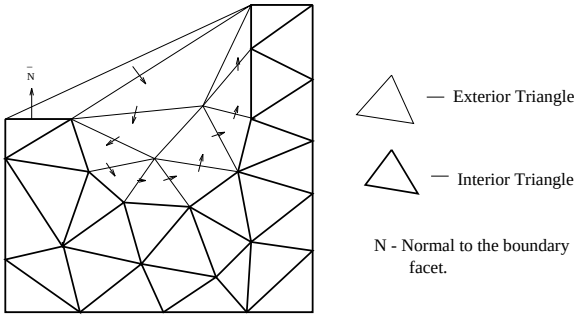
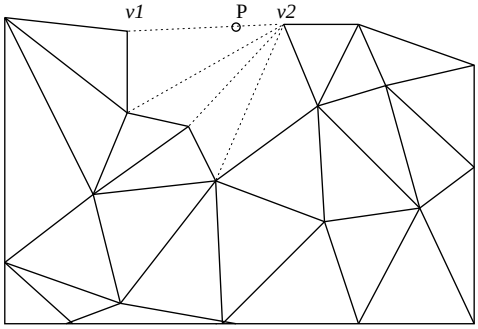


Figure 13. Removal of tetrahedra, generated outside the surface boundary



P - Point added along the boundary for boundary recovery

v1, v2 - Points along the boundary

Figure 14. Removal of point added on boundary for Boundary Recovery

and should be removed after the grid generation process is complete.

For this, the tetrahedra connected to it are found and deleted, leaving behind a concavity. The facets of this concave region are then connected in a simple manner to one of its boundary vertex to yield a valid triangulation. (see Figure 14)

Validity checks need to be done while doing the connections in order to avoid flat or intersecting tetrahedra.

7.6. Smoothing

The quality of the mesh is defined by the quality of its elements. Smoothing is a way to improve the grid quality by relocating the interior grid points. A common technique used to smooth the grid is the Laplacian technique, which repositions a point to the centroid of the kernel of its neighborhood.

$$c_p = \frac{1}{N} \sum_{i<N} p_i \tag{6}$$

where, c_p is the centroid and p_i are the neighbouring points Here, we have used Laplacian algorithm with a difference. [7] talks about closer ties between the face-areas and dihedral angles in a perfect tetrahedra. Hence the shape of tetrahedra can be determined by their dihedral angles. A dihedral angle is the angle between the two connected facets of a tetrahedron.

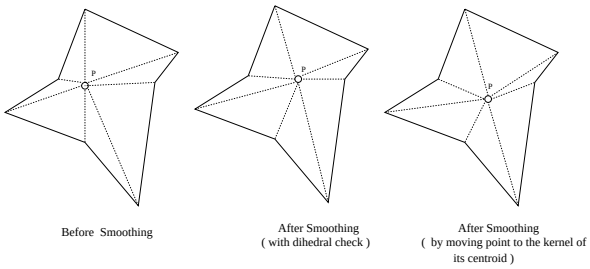


Figure 15. Comparison of smoothing techniques

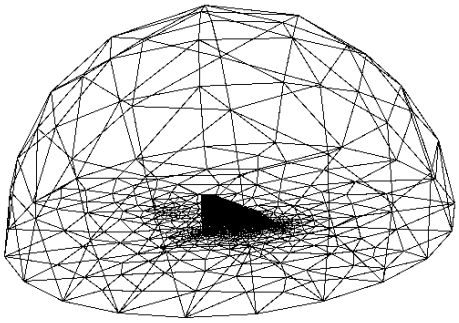


Figure 16. Wireframe viewing of volume consisting of aircraft wing inside a hemisphere

We opt to shift an interior point to the centroid of its neighbourhood provided it results into a better geometry, computed in terms of the dihedral angle. The minimum dihedral angle is computed before and after shifting. If the latter is better, then the point is repositioned at the centroid of its neighbourhood, else, its earlier position is retained.

Figure 15 shows a part of the grid before and after smoothing. It also compares the smoothing with dihedral check and that by shifting the point to the centroid of kernel of the connected geometry.

Validity checks need to be done to ensure that there are no intersecting or flat tetrahedra as described above in *Validity and Integrity Checks*.

8. Visualisation and Analysis

The tetrahedra generated could range from few hundred to millions in number, and hence complex to simply view them on a 2-D screen. VolGrid provides a number of techniques that help visualisation and analysis of the mesh, interactively.

8.1. Visualisation

Wireframe display: The grid can be viewed as a wire frame model. In this model, all edges of volume grid are drawn on the canvas with the user specified viewing parameters (see Figure 16).

Planar sections: Planar sections can be taken across the volume grid along a user specified plane. As clipping leads to incomplete elements, it does not serve the purpose. Thus the distribution of volume across the grid can be visualised,

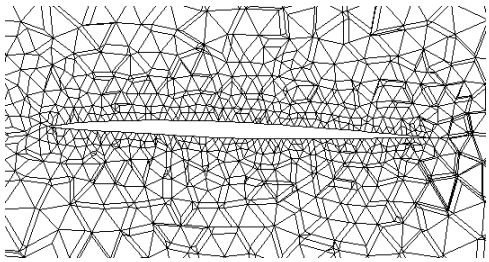


Figure 17. Planar section of tetrahdral grids

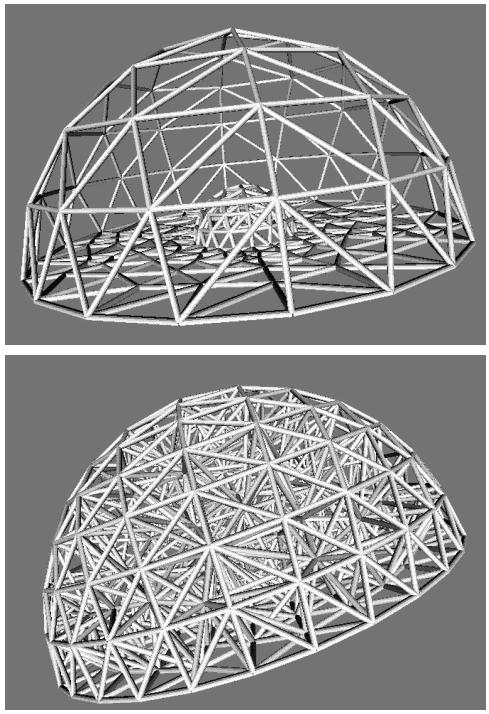


Figure 18. Inspection of surface grids and volume grids as VRML worlds

which is nothing but the volume gradient (see Figure 17).

Slicing: The difference between the planar sections and the slice is that, slicing maintains complete elements. Here, an isosurface cuts the volume grid, leaving behind a set of tetrahedra that are not intersected by the plane.

Viewing as a VRML world: VolGrid can export the grid as a VRML world. This can be viewed using any VRML viewer. Viewing as a VRML world is supported for the surface grid as well as volume grids. (See Figure 18).

8.2. Analysis

VolGrid incorporates surface grid analysis as well, in addition to the volume grid analysis. In both the cases, analysis is done by calculating properties across its elements, which define the grid quality.

The properties computed for analysing the surface grids are aspect ratio, area of an element, minimum and maximum angles between edges meeting at a vertex, number of elements, surface area of the surface grid, enclosed volume (if the surface grid is closed), number of boundary loops.

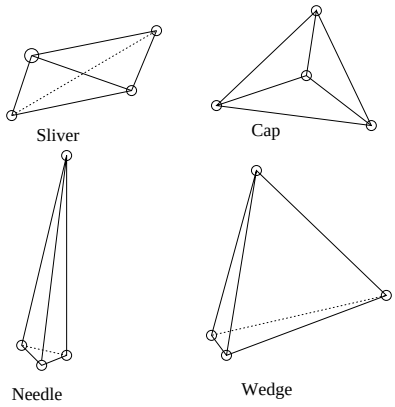


Figure 19. Poorly shaped tetrahedra

The volume grid analysis involves the following:

Aspect Ratio: The definition used for aspect ratio is "the ratio of length of the longest side to the length of the shortest side." An equilateral tetrahedron will have all sides of equal length. In an isotropic volume grid, the average aspect ratio should be as close to unity as possible.

Dihedral Angle: Figure 19 illustrates some tetrahedra having reasonably acceptable aspect ratios, however they are either flat, thin or slivers. Hence, the aspect ratio may not reflect the "quality" of the tetrahedra. The dihedral angle parameter represents the angle between two facets meeting at an edge and hence is a better measure. It is computed as the angle between the normals to the facets.

Solid Angle: Solid angle at a vertex of a tetrahedron is measured by placing the vertex at the center of a unit sphere and finding the area subtented on the surface of the sphere by the sides of tetrahedron meeting at the vertex. Solid angle is given as:

$$\theta = tan^{-1}\left[\frac{a \cdot b \times c}{1 + a \cdot b + b \cdot c + c \cdot a}\right] \qquad (7)$$

where, a, b and c are vectors corresponding to the edges meeting at the vertex.

Volume: Volume of a tetrahedron is given by one sixth of the scalar triple product of vectors meeting at any of the vertices of the tetrahedron.

$$V = \frac{1}{6}(a \cdot b \times c) \qquad (8)$$

Gradient of volume field: The change in the volumes of the tetrahedra may vary abruptly or smoothly across the grid. To determine the variation in volumes, we determine the gradient, which is the absolute difference between the volumes of neighbouring tetrahedra. The volume gradient analysis gives the maximum and minimum volume gradient values in the grid.

9. Future Work

VolGrid is a complete solution for building up of a volume data and achieving its tetrahedral decomposition, along with the tools for its visualisation and analysis. Major stress has been put on making the package user-friendly and robust.

Future pursuit will largely be toward achiving multi-million tetrahedra grids for entire aircraft geometries put inside wind-tunnel-like enclosures. The current work concentrates on improving the data-structures for accomodating large grids, multi-block strategies for unstructured tetrahedral grids, new grid visualization and analysis techniques.

Acknowledgments

This work has been funded by Aeronautical Development Agency (ADA), Bangalore. The authors acknowledge the help and suggestions received from the members of the CFD Group at ADA.

References

[1] H. Borouchaki, F. Hecht, E. Saltel, and P.-L. George. Reasonably efficient Delaunay based mesh generator in 3 dimensions. In *4th Annual Intl. Meshing Roundtable*, Oct. 1995. http://www.ce.cmu.edu/ sowen/Roundtable.agenda.html.

[2] E. Bruzzone and L. D. Floriani. Two data structures for building tetrahedralizations. *Visual Computer*, 6(5):266–283, 1990.

[3] D. A. Field. The legacy of automatic mesh generation from solid modeling. *Computer Aided Geometric Design*, 12:651–673, 1995.

[4] L. Freitag and C. Ollivier-Gooch. The effect of mesh quality on solution efficiency. In *Proceedings of the 6th International Meshing Roundtable*, October 1997.

[5] C. Gitlin and C. Johnson. Meshview: A tool for exploring 3d unstructured tetrahedral meshes. In *Proceedings of the 5th International Meshing Roundtable*, October 1996.

[6] N. Golias and T. Tsiboukis. An approach to refining three-dimensional tetrahedral meshes based on Delaunay transformations. *Intl. J. Numer. Meth. Eng.*, 37:793–812, 1994.

[7] D. McConnell. Hedronometry, 1993. (http://www.ics.uci.edu/ẽppstein/ junkyard/) Worcester Polytechnic Institute.

[8] M. MJ and B. LA. Combinatorial solid geometry, boundary representations and n-manifold geometry. In Rogers and Earnshaw, editors, *State of the art in Computer Graphics – Visualization and Modeling*. Springer-Verlag, New York, 1991.

[9] P. Möller and P. Hansbo. On advancing front mesh generation in three dimensions. *Intl. J. Numer. Meth. Eng.*, 38:3551–3569, 1995.

[10] L. R. and P. P. Generation of three-dimensional unstructured grids by the advancing-front method. *AIAA-88-0515*, 1988.

[11] H. Samet. *Applications of Spatial Data Structures: computer graphics, image processing and GIS*. Addison Wesley, 1990.

[12] K. Shimada. *Physically-Based Mesh Generation: Automated Triangulation of Surfaces and Volumes via Bubble Packing*. PhD thesis, ME Dept., MIT, 1993.

[13] D. Watson. Computing n-dimensional delaunay triangulation with applications to voronoi prototypes. *The Computer Journal*, 24(2), 1981.

[14] N. Weatherill. The integrity of geometrical boundaries in two dimensional Delaunay triangulation. *Com. in Appl. Num. Meth.*, 6:101–109, 1990.

[15] N. P. Weatherill and O. Hassan. Efficient three-dimensional delaunay triangulation with automoatic point creation and imposed boundary constraints. *Internation Journal for Numerical Methods in Engineering*, 37:2005–2039, 1994.