A Survey of Unstructured Mesh Generation Technology

Steven J. Owen
Department of Civil and Environmental Engineering, Carngie Mellon University, Pittsburgh, PA.
and
ANSYS Inc., Canonsburg, PA.
steve.owen@ansys.com

Abstract

A brief survey of some of the fundamental algorithms in unstructured mesh generation is presented. Included is a discussion and categorization of triangle, tetrahedral, quadrilateral and hexahedral mesh generation methods currently in use in academia and industry. Also included is a brief discussion of smoothing, cleanup and refinement algorithms. An informal survey of currently available mesh generation software is also provided comparing some of their main features.

1. Introduction

Automatic unstructured mesh generation is a relatively new field. Within its short life span we have seen tremendous advances in many diverse fields. Once in a while, it is useful to step back from our own expertise and look at the entire picture of what is going on in the field. The purpose of this survey is to give some perspective to what the current trends are in mesh generation and outline some of the major technology areas, who is working in these fields and what software is available.

Probably the simplest approach is to first break down the technology based on the shape of element generated. We will consider triangle and quad generation methods in 2D and tetrahedral and hexahedral methods in 3D. Straddled between 2D and 3D, we have surface meshing, which has it's own set of issues. In addition we have another set of issues dealing with post processing of the mesh including smoothing, cleanup and refinement. Within each of these issues, have emerged a few clear categories of algorithms, which tend to dominate much of the literature and software. Not included in this survey are a wide variety of equally important related topics such as adaptive, anisotropic and parallel mesh generation as well as data structure and geometry management issues. Because of the immense scope of the field of unstructured mesh generation, I have limited this survey to include what I consider the more fundamental aspects of the field. Since I do not purport to be an expert in all fields of mesh generation, this will be at best, a cursory look at the main issues in each category.

1.1 Software Survey

As part of this paper, I conducted an informal survey of software vendors, research labs and educational institutions that develop mesh and grid generation software. The purpose was to get a broad picture of who was currently involved in developing software and what common algorithms were employed. The results of the survey are included as an appendix to this paper. They are also posted on the World Wide Web[1]. From the over 100 surveys mailed, approximately 80 responded. While the emphasis of the survey was unstructured, many unstructured codes are also included.

The survey is certainly not a complete list of all those developing software, but it does illustrate the wide range of mesh generation technology currently available. Included are simple research codes used by only a few people, to commercial codes integrated within complex analysis packages.

1.2 Structured vs. Unstructured

This survey paper focuses on unstructured meshing technology. There is a large group of literature[2,3] and software[4] that deals with structured meshing commonly referred to as "grid generation". Strictly speaking, a structured mesh can be recognized by all interior nodes of the mesh having an equal number of adjacent elements. For our purposes, the mesh generated by a structured grid generator is typically all quad or hexahedral. Algorithms employed generally involve complex iterative smoothing techniques that attempt to align elements with boundaries or physical domains. Where non-trivial boundaries are required, "block-structured" techniques can be employed which allow the user to break the domain up into topological blocks. Structured grid generators are most commonly used within the CFD field, where strict alignment of elements can be required by the analysis code or necessary to capture physical phenomenon.

Unstructured mesh generation, on the other hand, relaxes the node valence requirement, allowing any number of elements to meet at a single node. Triangle and Tetrahedral meshes are most commonly thought of when referring to unstructured meshing, although quadrilateral and hexahedral meshes can also be unstructured. While there is certainly some overlap between structured and unstructured mesh generation technologies, the main feature which distinguish the two fields are the unique iterative smoothing algorithms employed by structured grid generators.

2.0 Tri/Tetrahedral Meshing

Triangle and tetrahedral meshing are by far the most common forms of unstructured mesh generation. Most techniques currently in use can fit into one of three main categories:

1. Octree
2. Delaunay
3. Advancing Front

Although there is certainly a difference in complexity when moving from 2D to 3D, the algorithms discussed are for the most part applicable for both triangle and tetrahedral mesh generation.

2.1 Octree

The Octree technique was primarily developed in the 1980s by Mark Shephard's[5,6] group at Rensselaer. With this method, cubes containing the geometric model are recursively subdivided until the desired resolution is reached. Figure 1 shows the equivalent two-dimensional quadtree decomposition of a model. Irregular cells are then created where cubes intersect the surface, often requiring a significant number of surface intersection calculations. Tetrahedra are generated from both the irregular cells on the boundary and the internal regular cells. The Octree technique does not match a pre-defined surface mesh, as an advancing front or Delaunay mesh might, rather surface facets are formed wherever the internal octree structure intersects the boundary. The resulting mesh also will change as the orientation of the cubes in the octree structure is changed and can also require. To ensure element sizes do not change too dramatically, a maximum difference in octree subdivision level between adjacent cubes can be limited to one. Smoothing and cleanup operations can also be employed to improve element shapes.

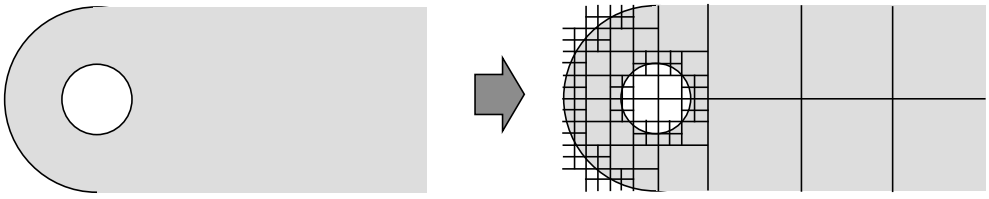

Figure 1. Quadtree decomposition of a simple 2D object

From the survey, only four of the 38 codes generating tetrahedral meshes reported using some form of octree technique. SCOREC[7] at Rensselaer develops a set of mesh generation tools called MEGA that

utilizes the Octree technique that is available through their partners program. A public domain octree mesh generator called QMG[8] is available from Steve Vivasis at Cornell.

2.2 Delaunay

By far the most popular of the triangle and tetrahedral meshing techniques are those utilizing the Delaunay[9] criterion. The Delaunay criterion, sometimes called the "empty sphere" property simply stated, says that any node must not be contained within the circumsphere of any tetrahedra within the mesh. A circumsphere can be defined as the sphere passing through all four vertices of a tetrahedron. Figure 2 is a simple two-dimensional illustration of the criterion. Since the circumcircles of the triangles in (a) do not contain the other triangle's nodes, the empty *circle* property is maintained. Although the Delaunay criterion has been known for many years, it was not until the work of Charles Lawson[10] and Dave Watson[11] that the criterion was utilized for developing algorithms to triangulate a set of vertices. A simple public domain 3D Delaunay triangulation program called Qhull is available from the University of Minneapolis. The criterion was later used in developing meshing algorithms by Timothy Baker[12] at Princeton, Nigel Weatherill[13] at Swansea, Paul-Louis George[14] at INRIA among others.

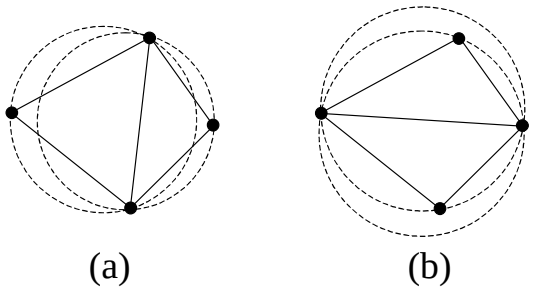

Figure 2. Example of Delaunay criterion. (a) maintains the criterion while (b) does not.

The Delaunay criterion in itself, is not an algorithm for generating a mesh. It merely provides the criteria for which to connect a set of existing points in space. As such it is necessary to provide a method for generating node locations within the geometry. A typical approach is to first mesh the boundary of the geometry to provide an initial set of nodes. The boundary nodes are then triangulated according to the Delaunay criterion. Nodes are then inserted incrementally into the existing mesh, redefining the triangles or tetrahedra locally as each new node is inserted to maintain the Delaunay criterion. It is the method that is chosen for defining where to locate the interior nodes that distinguishes one Delaunay algorithm from another.

2.2.1 Point insertion

The simplest point insertion approach is to define nodes from a regular grid of points covering the domain at a specified nodal density. In order to provide for varying element sizes, a user specified sizing function can also be defined and nodes inserted until the underlying sizing function is satisfied. Another approach is for nodes to be recursively inserted at triangle or tetrahedral centroids. Weatherill and Hassan[13] propose a tetrahedral mesh generation scheme where nodes are inserted at a tetrahedron's centroid provided the underlying sizing function is not violated.

An alternate approach is to define new nodes at element circumcircle/sphere centers as proposed by Chew[15] and Ruppert[16]. When a specific order of insertion is followed, this technique is often referred to as "Guaranteed Quality" as triangles can be generated with a minimum bound on any angle in the mesh. Jonathon Shewchuk[17] at CMU has developed a 2D version of this algorithm and makes it available free of charge for research purposes.

Similar to the circumcircle point insertion method, another technique introduced by Rebay[18] is the so-called, Voronoi-segment point insertion method. A Voronoi segment can be defined as the line segment between the circumcircle centers of two adjacent triangles or tetrahedra. The new node is introduced at a

point along the Voronoi segment in order to satisfy the best local size criteria. This method tends to generate very structured looking meshes with six triangles at every internal node.

Another method, introduced by Marcum[19] is an advancing front approach to node insertion. Nodes are inserted incrementally, but added from the boundary towards the interior. Each facet is examined to determine the ideal location for a new fourth node on the interior of the existing Delaunay mesh. The node is then inserted and local reconnection is performed. This method tends to generate elements well aligned with the boundary with a very structured appearance to the mesh. Dave Marcum provides both a 2D and 3D version of his mesh generators through the ERC[20] at Mississippi State.

One straightforward method used by INRIA[21] in their mesh generator GSH3D[22], is point insertion along edges. A set of candidate vertices is generated by marching along the existing internal edges of the triangulation at a given spacing ratio. Nodes are then inserted incrementally, discarding nodes that would be too close to an existing neighbor. This process is continued recursively until a background sizing function is satisfied.

A variety of other methods for point insertion have also been proposed, but most have a similar flavor to those discussed above

2.2.2 Boundary Constrained Triangulation

In many finite element applications, there is a requirement that an existing surface triangulation be maintained. In most Delaunay approaches, before internal nodes are generated, a three dimensional tessellation of the nodes on the geometry surface is produced. In this process, there is no guarantee that the surface triangulation will be satisfied. In many implementations, the approach is to tessellate the boundary nodes using a standard Delaunay algorithm without regard for the surface facets. A second step is then employed to force or recover the surface triangulation. Of course by doing so, the triangulation may no longer be strictly "Delaunay", hence the term "Boundary Constrained Delaunay Triangulation".

In two dimensions the edge recovery is relatively straightforward. George[23] describes how the edges of a triangulation may be recovered by iteratively swapping triangle edges. The process is considerably more complex in three dimensions, since after recovering all edges in the surface triangulation, there is no guarantee that the surface facets themselves will be recovered. Additional facet recovery operations can be required to maintain the surface triangulation. While the two dimensional recovery process is guaranteed to produce a boundary conforming triangulation, there are cases[24] in three dimensions where a valid triangulation can not be defined without first inserting additional vertices. This fact increases the complexity of any three dimensional boundary recovery procedure. Two different methods presented in the literature for recovery of the boundary include George[14] and Weatherill[13].

In the first approach defined by George[14] and implemented in INRIA's GSH3D[22] software, edges are recovered by performing a series of tetrahedral transformations by swapping two adjacent tetrahedra for three, as shown in Figure 3. Where a swap cannot resolve the edge, nodes must sometimes be inserted. After edges have been recovered, in order to recover the face, additional transformations are performed, mostly characterized by swapping three adjacent tetrahedra at an edge for two. More complex transformations or additional nodes can be inserted during the face recovery phase if the transformations do not resolve the surface facet.

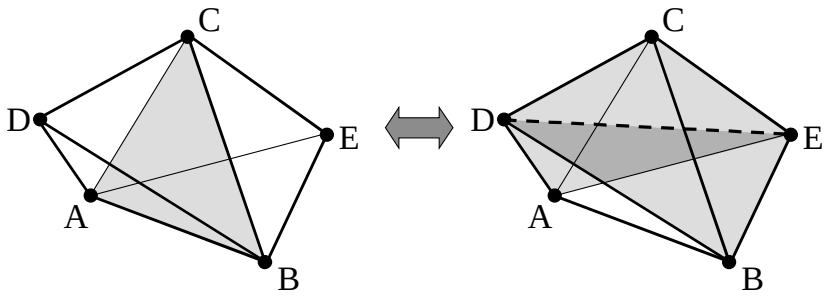

Figure 3. Tetrahedral transformation where two tets are swapped for three.

The second approach defined by Weatherill also involves an edge recovery phase and a face recovery phase. The main difference with this approach is that rather than attempting to transform the tetrahedra to recover edges and faces, nodes are inserted directly into the triangulation wherever the surface edge or facet cuts non-conforming tetrahedra. This process temporarily adds additional nodes to the surface. Once the surface facets have been recovered, additional nodes that were inserted to facilitate the boundary recovery are deleted and the resulting local void retriangulated.

Another approach presented by Barry Joe[25], is able to avoid the boundary recovery problem altogether. Provided the geometry is convex, Joe is able to define a boundary conforming tetrahedral mesh. The emphasis in this method, rather than attempting to repair the boundary of an arbitrary non-convex surface triangulation, is to decompose the geometry into convex regions that can be separately processed. An older unsupported public domain version of Barry Joe's code, Geompack, is available from the University of Alberta[26] via anonymous ftp.

2.3 Advancing Front

Another very popular family of triangle and tetrahedral mesh generation algorithms is the advancing front, or moving front method. Two of the main contributors to this method are Rainald Lohner[27,28] at George Mason University and S. H. Lo[29,30] at the University of Hong Kong. In this method, the tetrahedra are built progressively inward from the triangulated surface. An active front is maintained where new tetrahedra are formed. Figure 4 is a simple two-dimensional example of the advancing front, where triangles have been formed at the boundary. As the algorithm progresses, the front will advance to fill the remainder of the area with triangles. In three-dimensions, for each triangular facet on the front, an ideal location for a new fourth node is computed. Also determined are any existing nodes on the front that may form a well-shaped tetrahedron with the facet. The algorithm selects either the new fourth node or an existing node to form the new tetrahedron based on which will form the best tetrahedron. Also required are intersection checks to ensure that tetrahedron do not overlap as opposing fronts advance towards each other. A sizing function can also be defined in this method to control element sizes. Lohner[28] proposed using a course Delaunay mesh of selected boundary nodes over which the sizing function could be quickly interpolated. A version of S. H. Lo's advancing front mesh generator is available with the ANSYS[31] suite of mesh generation tools.

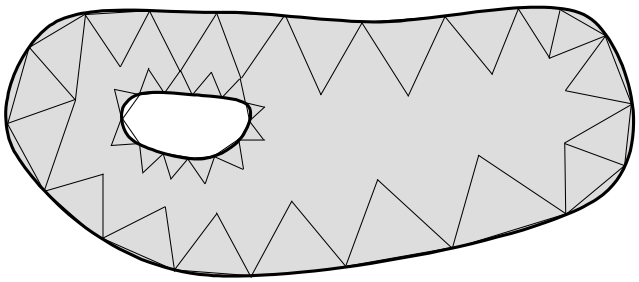

Figure 4. Example of advancing front where one layer of triangles has been placed

A form of the advancing front method, sometimes called "advancing layers", is also used for generating boundary layers for CFD, Navier-Stokes applications. This method lends itself well to control of element sizes near the boundary. Pirzadeh[32] presents a method where the elements are stretched in the direction of the boundary, the expected direction of fluid flow. A public domain version of Pirzadeh's code, VGRID[33] is available from NASA, Langley.

3. Quad/Hexahedral Meshing

Automatic unstructured mesh generation algorithms have lent themselves more readily to triangle and tetrahedral meshing. As a result, most of the literature and software are triangle and tetrahedral. In spite of this, there is a significant group of literature that focuses on unstructured quad and hexahedral methods. Unstructured quad[34] and hex[35] meshing software have also become widely available in recent years. Unlike

triangle and tetrahedral methods, extension from a 2D quadrilateral algorithm to a 3D hexahedral method is not generally straightforward.

3.1 Mapped Meshing

When the geometry of the domain is applicable, quad or hex mapped meshing[36] will generally produce the most desirable result. Although mapped meshing is considered a structured method, it is quite common for unstructured codes to provide a mapped meshing option. For mapped meshing to be applicable, opposite edges of the area to be meshed must have equal numbers of divisions. In 3D, each opposing face of a topological cube must have the same surface mesh. This can often be impossible for an arbitrary geometric configuration or can involve considerable user interaction to decompose geometry into mapped meshable regions and assign boundary intervals. In order to reduce human interaction, research has be done in recent years through the CUBIT[37] project at Sandia National Labs to automatically recognize features[38]and decompose geometry[39] into separate mapped meshable areas and volumes. Work has also been done to automate interval assignments[40].

Another category of mapped meshing, also developed as part of the CUBIT[37] project is referred to as sub-mapping[41]. This method, rather than decomposing the geometry directly, determines an appropriate *virtual* decomposition based on corner angles and edge directions. The separate map-meshable regions are then meshed separately. This method is suitable for blocky shapes and volumes that have well defined corners and cube-like regions.

Sweeping, sometimes referred to as 2 ½-D meshing, is another class of mapped hexahedral meshing. A quadrilateral mesh can be *swept* through space along a curve. Regular layers of hexahedra are formed at specified intervals using the same topology as the quadrilateral mesh. This technique can be generalized to mesh certain classes of volumes by defining so-called *source* and *target* surfaces. Provided the source and target surface have similar topology and the surfaces are connected by a set of map-meshable surfaces, the quad elements of the source area can be swept through the volume to generate hexahedra as shown in Figure 5. Care must be taken in locating internal nodes during the sweeping process and several papers[42,43] have been presented addressing this issue.

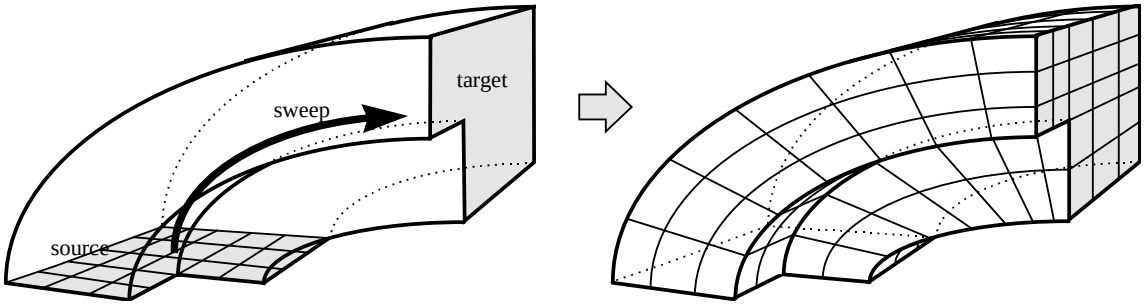

Figure 5. Hex elements generated by sweeping

Blacker[44] generalizes and extends the applicability of sweeping when he introduces the *Cooper Tool*. The Cooper tool allows for multiple source and target surfaces while still requiring a single sweep direction. With this tool, the topology is allowed to branch or split along the sweep direction. In addition, the topology of source and target surfaces are not required to be similar. With these requirements relaxed, a greater subset of geometry may be meshed with generally very high quality elements. The cooper tool is provided as part of the Fluent pre-processor, Gambit[45].

3.2 Unstructured Quad Meshing

Unstructured quadrilateral meshing algorithms can, in general, be grouped into two main categories: direct and indirect approaches. With an indirect approach, the domain is first meshed with triangles. Various algorithms are then employed to convert the triangles into quadrilaterals. With a direct approach,

quadrilaterals are placed on the surface directly, without first going through the process of triangle meshing.

3.2.1 Indirect Methods

One of the simplest methods for indirect quadrilateral mesh generation includes dividing all triangles into three quadrilaterals, as shown in Figure 6. This method guarantees an all-quadrilateral mesh, but a high number of irregular nodes are introduced into the mesh resulting in poor element quality. An alternate algorithm is to combine adjacent pairs of triangles to form a single quadrilateral as shown in Figure 7. While the element quality increases using this method, a large number of triangles may be left.

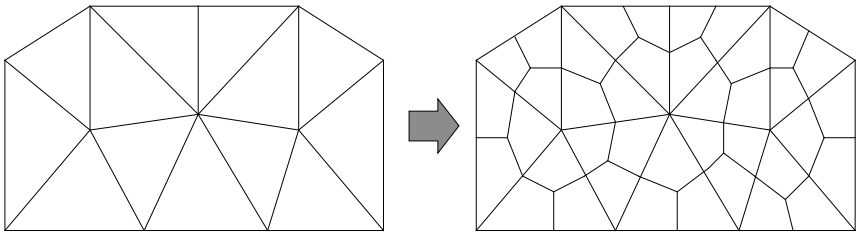

Figure 6. Quad mesh generated by splitting each triangle into three quads

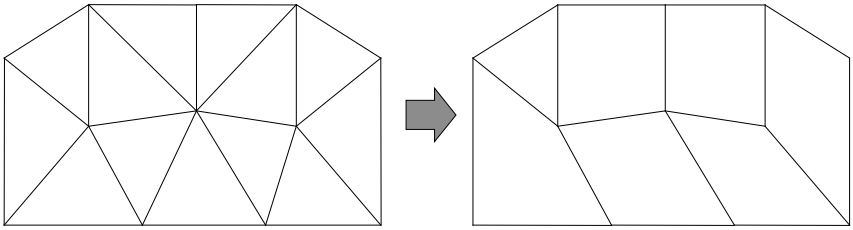

Figure 7. Quad-dominant mesh generated by combining triangles.

The triangle combining method can be improved, if some care is taken in the order in which triangles are combined. In an effort to maximize the number of quadrilaterals, Lo[46] defined an algorithm that suggested several heuristic procedures for the order in which triangles could be combined. The result is a quad-dominant mesh containing a minimal number of triangles. Johnston[47] proposes additional local element splitting and swapping strategies to increase the number and quality of quads.

Lee[48] later enhances Lo's [46] strategy by including local triangle splitting. In addition, an advancing front approach is used over the initial triangles. An initial set of fronts is defined consisting of the edges of triangles at the boundary of the domain. Triangles are systematically combined at the front, advancing towards the interior of the area. Each time a set of triangles is combined the front advances. The front always defines the division between quadrilaterals already formed and triangles yet to be combined. With this technique, Lee is able to guarantee an all-quadrilateral mesh, provided the initial number of edges on the boundary is even.

Since all operations are local, indirect methods have the advantage of being very fast. Global intersection checks are not necessary as is required with some forms of direct methods. The drawback to indirect methods is that there are typically many irregular nodes left in the mesh. Even if few irregular nodes exist, there is no guarantee that the elements will align with the boundary, a desirable property for some applications. Some of the irregular nodes can be reduced, and hence element quality increased by performing topological clean-up operations (discussed later).

Another method recently introduced by the author, known as Quad Morphing[49] also utilizes an advancing front approach to convert triangles to quads, but is able to significantly reduce the number of irregular nodes in the mesh. With this approach, local edge swaps are performed and additional nodes introduced in order to ensure boundary alignment and orthogonality. Any number of triangles may be deleted to create a single quad.

3.2.2 Direct Methods

Many methods for direct generation of quad meshes have been proposed. Among these methods, there appears to be two main categories. The first are methods that rely on some form of decomposition of the domain into simpler regions than can be resolved by one of a series of templates. The second category are those that utilize a moving front method for direct placement of nodes and elements.

3.2.2.1 Quad Meshing by Decomposition

The quad-tree decomposition technique proposed by Baehmann[50] is among the first methods utilizing decomposition of the area for quadrilateral meshing. After an initial decomposition of the 2D space into a quad-tree based on local feature sizes, quadrilateral elements are fitted into the quad-tree *leaves*, adjusting nodes in order to conform to the boundary.

Talbert[51] later introduces another decomposition technique. With this approach, the domain is recursively subdivided into simple polygonal shapes. The resulting polygons satisfy a limited number of templates into which quadrilateral elements are inserted. Chae[52] has recently proposed enhancements to Talbert's algorithm with similar work presented by Nowottny[53].

Quadrilateral meshing utilizing a *medial axis* decomposition of the domain was first introduced by Tam[54]. The medial axis can be thought of as a series of lines and curves generated from the midpoint of a maximal circle as it is rolled through the area (Figure 8). Having decomposed the area into simpler regions, sets of templates are then employed to insert quadrilaterals into the domain. Linear programming techniques are used in order to maintain compatibility of element divisions between adjoining regions of the domain.

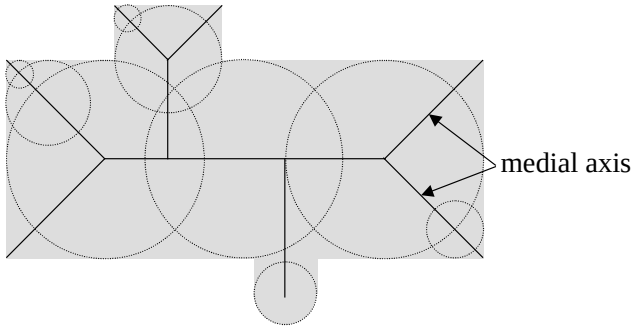

Figure 8. Decomposition of an area using the medial axis

Joe[55] also utilizes decomposition algorithms to decompose the area into convex polygons. Using techniques previously developed for triangle mesh generation[56], Joe constructs a boundary constrained quadrilateral mesh within each convex sub-domain of the area.

3.2.2.2 Advancing Front Quad Meshing

Zhu[57] is among the first to propose a quadrilateral meshing algorithm using an advancing front approach. Starting with an initial placement of nodes on the boundary, individual elements are formed by projecting edges towards the interior. Two triangles are formed using traditional triangle advancing front methods and then combined to form a single quadrilateral.

The *paving* algorithm introduced by Blacker and Stephenson[58], presents a method for forming complete rows of elements starting from the boundary and working in. Methods for projection of nodes, handling of special geometric situations and intersection of opposing fronts are discussed. Cass[59] further developed paving, by generalizing the method for three-dimensional surfaces. White[60] recently proposed enhancements to the paving algorithm suggesting individual placement of elements rather than complete rows. The paving algorithm is currently implemented as part of the CUBIT[37] software as well as several commercial packages including MSC Patran[61] and Fluent's Gambit[45] software.

3.3 Unstructured Hex Meshing

Similar to quadrilateral meshing, there are both direct and indirect methods for unstructured hex meshing.

3.3.1 Indirect Methods

Indirect methods, although not in wide use have been proposed for some applications[62]. Provided a solid can be tet meshed, each tetrahedron can be subdivided into four hexahedra as shown in Figure 9. Most finite element analysts, because of the poor element quality that will in general result, have rejected this solution.

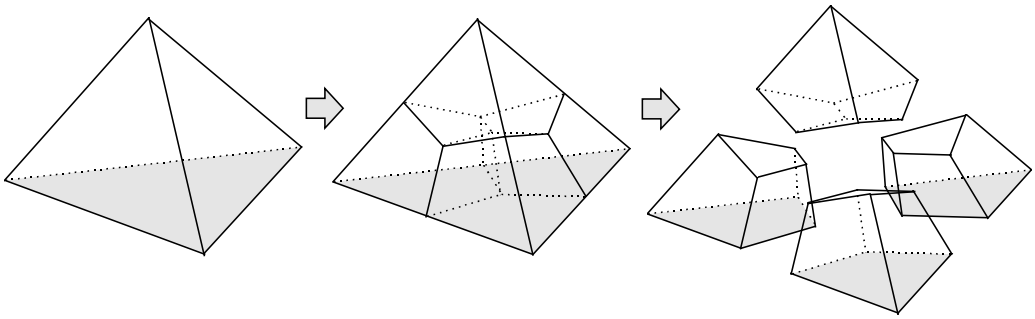

Figure 9. Decomposition of a tetrahedron into four hexahedra

An equivalent indirect hexahedral mesh generation scheme that will combine tetrahedra, similar to combining triangles to form quadrilaterals has not been presented in the literature. The simplest tetrahedralization of a cube will contain five tetrahedra. An indirect method that combines tets to form hexes would therefore need to look for combinations of five or more tetrahedra to form a single hexahedra. This problem to date has not proved a reasonable nor tractable method for mesh generation.

3.3.2 Direct Methods

There are currently four distinct strategies proposed for unstructured all-hex mesh generation that are predominant in the literature:

1. grid-based
2. medial surface
3. plastering
4. whisker weaving

3.3.2.1 Grid-Based

The grid-based approach, proposed by Schneiders[63] involves generating a *fitted* three dimensional grid of hex elements on the interior of the volume. Hex elements are added at the boundaries to fill gaps where the regular grid of hexes does not meet flush with the surface. This method, while robust, tends to generate poor quality elements at the boundary of the volume. Hex elements will in general not be aligned with the boundary. The resulting mesh generated from the grid-based approach is also highly dependent upon the orientation of the interior grid of hex elements. In addition, element sizes must be approximately all the same. In recent work, Weiler[64] and Schneiders[65] have introduced modifications that allow for significant transition in element sizes utilizing an octree decomposition of the domain. Mesh generators based on the grid-based approach are available in the Hexar[66] software from Cray Research and in MARC's Mentat[67] software.

3.3.2.2 Medial Surface

Medial surface methods[68,69,70] involve an initial decomposition of the volume. As a direct extension of the medial axis method for quad meshing, the domain is subdivided by a set of medial surfaces, which can be thought of as the surfaces generated from the midpoint of a maximal sphere as it is rolled through the volume. The decomposition of the volume by medial surfaces is said to generate map meshable regions. A series of templates for the expected topology of the regions formed by the medial surfaces are utilized to fill the volume with hexahedra. Linear programming is used to ensure element divisions match from one region to another. This method, while proving useful for some geometry, has been less than reliable for general geometry. Robustness issues in generating the medial surfaces as well as providing for all cases of regions defined by the medial surfaces has proved to be a difficult problem. Medial surface methods are incorporated into the FEGS' CADFix[71] hexahedral mesh generator and within Solidpoint's Turbomesh[72] software.

3.3.2.3 Plastering

Plastering[73,74] is an attempt to extend the paving algorithm to three dimensions. With this method, elements are first placed starting with the boundaries and advancing towards the center of the volume as shown in Figure 10. A heuristic set of procedures for determining the order of element formation is defined. Similar to other advancing front algorithms, a current front is defined consisting of all quadrilaterals. Individual quads are projected towards the interior of the volume to form hexahedra. In addition, plastering must detect intersecting faces and determine when and how to connect to pre-existing nodes or to *seam* faces. As the algorithm advances, complex interior voids may result, which in some cases are impossible to fill with all-hex elements. Existing elements, already placed by the plastering algorithm must sometimes be modified in order to facilitate placement of hexes towards the interior.

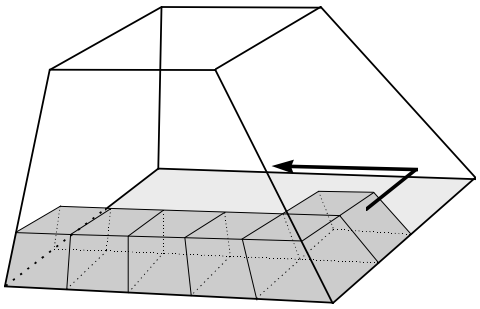

Figure 10. Plastering process forming elements at the boundary.

Currently, the plastering algorithm has not been proven to be reliable on a large class of problems. Although in many cases, several layers of hex elements may be successfully placed on the boundary of the volume, intersection and closure procedures are less than reliable. Sandia's CUBIT[37] project is continuing research on plastering and makes it available in their software.

3.3.2.4 Whisker Weaving

Whisker weaving, first introduced by Tautges and Blacker[75], is based on the concept of *the spatial twist continuum* (STC)[76]. Tautges describes the STC as the dual of the hexahedral mesh, represented by an arrangement of intersecting surfaces which bisect hexahedral elements in each direction. Figure 11 shows a simple representation of the *twist planes* of the STC defined for a volume composed of only two hexahedra.

The principal behind whisker weaving is to first construct the STC or dual of the hex mesh. With a complete STC, the hex elements can then be fitted into the volume using the STC as a guide. This is done by beginning with a topological representation of the loops formed by the intersection of the twist planes with the surface. The loops can be easily determined from an initial quad mesh of the surface. The objective of the whisker weaving algorithm is to determine where the intersections of the twist planes will

occur within the volume. Since this is done topologically, there are no actual intersection calculations performed. Once a valid topological representation of the twist planes has been achieved, hexes are then formed inside the volume. One hex is formed wherever three twist planes converge.

The whisker weaving algorithm has achieved some success, but has yet to prove itself as robust and reliable for a wide variety of problems.

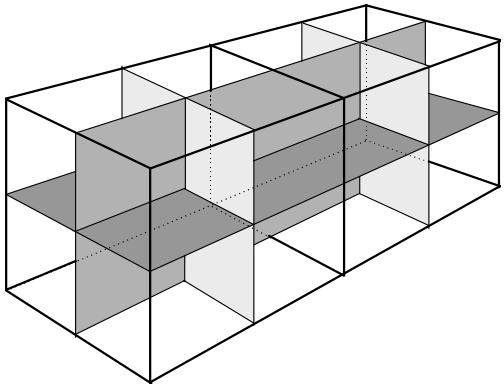

Figure 11. The STC composed of four twist planes, for a solid composed of two hexahedra

3.4 Hex-Dominant Methods

Since most methods for all-hex meshing appear to be less than robust, some researchers have proposed using a mixed hexahedra/tetrahedra mesh. A hex-dominant approach appears to be satisfactory in many cases. One simple approach introduced by the author[77] is to manually subdivide the geometry into regions that will readily accept a mapped mesh and those that are more geometrically complex. Within the complex regions a tet mesh is defined. Wherever the tet elements interface directly with hex elements, a pyramid shaped element may be formed. This option is provided with the ANSYS[31] mesh generation software.

Tuchinsky[78] recently proposed an algorithm for combining both plastering and tetrahedral meshing technologies. Using the plastering algorithm, hex elements are advanced as far as possible into the volume. The remaining voids within the volume are then filled with tetrahedra. The user also has the option of forming pyramid shaped elements at the interface between hex and tet elements. The CUBIT[37] software now provides an option to allow a hex-dominant mesh.

Min[79] also presents a similar method for hex-dominant meshing, utilizing offset geometry from the boundaries in order to form layers of hexes. After a series of shrunken shells have been advanced towards the interior of the volume, the remainder of the volume is filled with tetrahedra. In addition to tets and hexes, Min introduces pyramid and wedge shaped elements where applicable.

4. Surface Meshing

Many of today's mesh generation problems involve the formation of elements on arbitrary three-dimensional surfaces. These surfaces are typically represented by NURBS, which have been generated within a commercial CAD package. The resulting surface elements can either be used directly as structural shell elements, or used as input to a volumetric mesh generator. In either case, the algorithms used for two-dimensional mesh generation require some modification in order to generalize them for use on three-dimensional surfaces. Surface mesh generation algorithms can be classified as either parametric space or direct 3D.

4.1 Parametric Space

Parametric space algorithms will form elements in the two-dimensional parametric space of the surface. Since all NURBS surfaces have an underlying u-v representation, it can often be efficient to mesh in two dimensions and as a final step, map the u-v coordinates back to world space, x-y-z coordinates. The drawback to this method is that the elements formed in parametric space may not always form well-shaped elements in three dimensions once mapped back to the surface. To resolve this, parametric surface meshers can do one of two things: 1) modify or reparamaterize the underlying parametric representation so there is a reasonable mapping from parametric space to world space; or 2) modify the mesh generation algorithm so that stretched or anisotropic elements meshed in 2D will map back to well-shaped, isotropic elements in 3D.

The first method requires that in order to have a good paramaterization, the surface derivatives, ($\Delta\mathbf{u}$, $\Delta\mathbf{v}$), should not vary widely over the domain. Some exact arc-length reparamaterizations have been defined in the literature[80], but can be excessively costly. An approximate arc-length paramaterization or "warped parametric space" can be defined by selectively evaluating surface derivatives over the domain and adjusting local u-v values to hold the magnitude of $\Delta\mathbf{u}$, $\Delta\mathbf{v}$ roughly constant. For many cases, a warped parametric space can generate reasonable surface meshes, but there are many problems that the reparamaterization cannot adequately resolve. For this reason, much of the literature on surface meshing focuses on the second option of forming anisotropic elements in 2D that will map back to isotropic elements in 3D.

A common method used in practice is to take advantage of surface derivatives, $\Delta\mathbf{u}$, $\Delta\mathbf{v}$, easily computed from a NURBS surface. George and Borouchaki[81] propose the use of a metric derived from the first fundamental form of the surface. The metric is in the form of a 2X2 matrix and is used to transform vectors and distances in parametric space. With their Delaunay approach, the "empty circle" property, effectively becomes an "empty ellipse" property. Also included with the metric is the option to incorporate element sizing and stretching properties. A similar approach to parametric Delaunay surface meshing is presented by Chen and Bishop[82] and available in MARC's Mentat[67] software. Equivalent advancing front surface mesh generation algorithms, which utilize a metric derived from the first fundamental form of the surface are presented independently by Cuilliere[83] and Tristano[84]. Tristano's implementation is available in a recent release of the ANSYS[31] mesh generation tools.

4.2 Direct 3D

Direct 3D surface mesh generators form elements directly on the geometry without regard to the parametric representation of the underlying geometry. In some cases where a parametric representation is not available or where the surface paramaterization is very poor, direct 3D surface mesh generators can be useful. Lau and Lo[85,86] present an advancing front approach for arbitrary 3D surfaces. In this method surface normals and tangents must be computed in order to compute the direction of the advancing front. In addition, a significant number of surface projections are required to ensure that new nodes remain on the surface. Also of significance is the increased complexity of the intersection calculations required to ensure that triangles on the surface do not overlap.

A direct 3D implementation[59] of the paving[44] algorithm is also available in the CUBIT[37] software. Similar issues regarding additional projection and evaluations are also of significance to 3D paving. Cass[59] defines a heuristic "sticky space" in order to detect intersecting or overlapping quadrilaterals.

5. Mesh Post-processing

It is rare that any mesh generation algorithm will be able to define a mesh that is optimal without some form of post-processing to improve the overall quality of the elements. The two main categories of mesh improvement include smoothing and clean-up. Smoothing includes any method that adjusts node locations while maintaining the element connectivity. Clean-up generally refers to any process that changes the element connectivity.

5.1 Smoothing

Most smoothing procedures involve some form of iterative process that repositions individual nodes to improve the local quality of the elements. A wide variety of smoothing techniques have been proposed. These methods can generally be classified as follows:

1. Averaging methods
2. Optimization-based methods
3. Physically-based methods
4. Mid-node placement

5.1.1 Averaging Methods

Of the wide variety of smoothing algorithms, the simplest and most straight forward is Laplacian smoothing[87]. With this method, an internal node in the mesh is placed at the average location of any node connected to it by an edge. With little modification, this technique can be applicable for any element shape. Most smoothing procedures will iterate through all the internal nodes in the mesh several times until any individual node has not moved more than a specified tolerance. Although it has its problems, it is simple to implement and is in wide use. Similar to Laplacian, there are a variety of other smoothing techniques, which iteratively reposition nodes based on a weighted average of the geometric properties of the surrounding nodes and elements. Canann[88] provides an overview of some of the common methods in use.

Averaging methods quite often also employ some form of additional constraint on the movement of a node. For example, because Laplacian smoothing alone sometimes has the tendency to invert or degrade the local element quality, a comparison of local element quality is made before and after the proposed move and the node moved only if element quality is improved. This is often referred to as *constrained Laplacian smoothing*. Canann[88] presents criteria for the movement of the node with this method.

5.1.2 Optimization-Based Methods

Rather than relying on heuristic averaging methods, some codes use optimization techniques to improve element quality. Optimization-based smoothing techniques measure the quality of the surrounding elements to a node and attempt to optimize by computing the local gradient of the element quality with respect to the node location. The node is moved in the direction of the increasing gradient until an optimum is reached. Canann[88] and Freitag[89] both present optimization-based smoothing algorithms.

While maintaining that optimization-based smoothing techniques provide superior mesh quality, the computational time involved is generally too excessive to use in standard practice. Canann[88] and Freitag[90] both recommend a combined Laplacian/optimzation-based approach. What is generally advocated is that Laplacian smoothing is done for the majority of the time, reverting to optimization based smoothing only when local element shape metrics drop below a certain threshold.

5.1.3 Physically-Based Methods

Another important area of mesh improvement includes methods that reposition nodes based on a simulated physically based attraction or repulsion force. Lohner[91] simulates the force between neighboring nodes as a system of springs interacting with each other. Shimada[92] and Bossen[93] view the nodes as the center of bubbles that are repositioned to attain equilibrium. With changes in the magnitude and direction of inter-particle forces, different anisotropic characteristics and element sizes can be achieved.

5.1.4 Mid-node Placement

While most smoothing methods focus on repositioning corner nodes, Salem[94] recently introduced a method providing criteria for repositioning mid-nodes on quadratic elements to improve element quality. This

method computes a region surrounding the mid-node known as the mid-node admissible space (MAS), shown in Figure 12, where the mid-node can safely be moved and maintain or improve element quality.

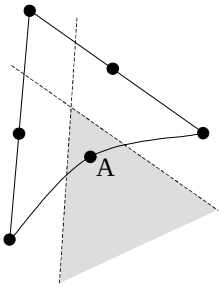

Figure 12. Mid-node admissible space for node at A

5.2 Cleanup

Like smoothing, there are a wide variety of methods currently employed to improve the quality of the mesh by making local changes to the element connectivities. Cleanup methods generally apply some criteria that must be met in order to perform a local operation. The criteria in general can be defined as: 1) shape improvement or 2) topological improvement.

In addition, cleanup operations are generally not done alone, but are used in conjunction with smoothing. Freitag[95] describes how smoothing and cleanup may be combined to efficiently improve overall element quality.

5.2.1 Shape improvement

For triangle meshes, simple diagonal swaps are often performed. For each interior edge in the triangulation a check can be made to determine at what position the edge would effectively improve the overall or minimum shape metric of its two adjacent triangles. The Delaunay criteria can also be used to determine the position of an edge. For Tetrahedral meshes, Barry Joe[96] presents a series of local transformations that are designed to improve the element quality. These include swapping two adjacent interior tets sharing the same face for three tets (see Figure 3). Likewise, three tets can be replaced with two. Other more complex transformations are also defined.

In some applications where mixed element meshes are supported, the element quality of two adjacent triangles may be preferable to a single poor quality quadrilateral. When this is the case, selected quadrilaterals may be split.

In some cases, particularly with curved surfaces, the elements resulting from the mesh generator may deviate significantly from the underlying geometry. For a triangle mesh, edge swaps can be performed based on which local position of the edge will deviate least from the surface. Although not strictly a cleanup operation, local refinement of the mesh may also be considered to capture surface features.

5.2.2 Topological Improvement

A common method for improving meshes is to attempt to optimize the number of edges sharing a single node. This is sometimes referred to as node *valence* or *degree*. In doing so, it is assumed that the local element shapes will improve. For a triangle mesh there should optimally be 6 edges at a node and four edges at a node surrounded by quads. Whenever there is a node that does not have an ideal valence, the quality of the elements surrounding it will also be less than optimal. Performing local transformations to the elements can improve topology and hence element quality. Several methods have been proposed for improving node valence for both triangle[97] and quadrilateral[98,99] meshes.

For volumetric meshes, valence optimization becomes more complex. In addition to optimizing the number of edges at a node, the number of faces at an edge can also be considered. For tetrahedral meshes this can involve a complex series of local transformations. For hexahedral elements, valence optimization is generally not considered tractable. The reason for this is that local modifications to a hex mesh will typically propagate themselves to more than the immediate vicinity. One special case of cleanup in hex meshes used in conjunction with the whisker weaving algorithm is presented by Mitchell[100].

5.3 Refinement

Element refinement procedures are numerous. For our purposes, refinement is defined as any operation performed on the mesh that effectively reduces the local element size. The reduction in size may be required in order to capture a local physical phenomenon, or it may be done simply to improve the local element quality. Some refinement methods in themselves can be considered mesh generation algorithms. Starting with a coarse mesh, a refinement procedure can be applied until the desired nodal density has been achieved. Quite frequently, refinement algorithms are used as part of an adaptive solution process, where the results from a previous solution provide criteria for mesh refinement. Methods have been proposed for triangle and tet refinement as well as quad and hex.

5.3.1 Triangle/Tetrahedral Refinement

Although there are certainly more methods defined, three of the principal methods for triangle and tetrahedral refinement include:

1. Edge bisection
2. Point insertion
3. Templates

5.3.1.1 Edge Bisection.

Edge bisection involves splitting individual edges in the triangulation. As a result, the two triangles adjacent the edge are split into two. Extended to volumetric meshing, any tetrahedron sharing the edge to be split must also be split as illustrated in Figure 13. Rivara[101] proposes criteria for the splitting of edges based on the longest edge of a triangle or tetrahedron.

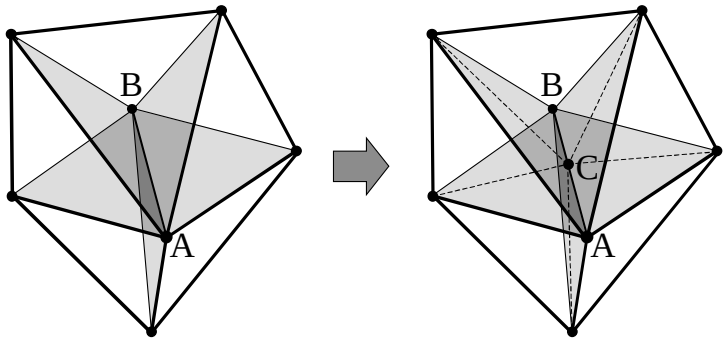

Figure 13. Edge bisection in a tetrahedral mesh. Edge A-B is split at point C, also splitting its surrounding tetrahedra.

5.3.1.2 Point Insertion

A simple approach to refinement is to insert a single node at the centroid of an existing element, dividing the triangle into three or tetrahedron into four. This method does not generally provide good quality elements, particularly after several iterations of the scheme. To improve upon the scheme, a Delaunay approach can be used that will delete the local triangles or tetrahedra and connect the node to the

triangulation maintaining the Delaunay criterion. Any of the Delaunay point insertion methods discussed previously could effectively be used for refinement.

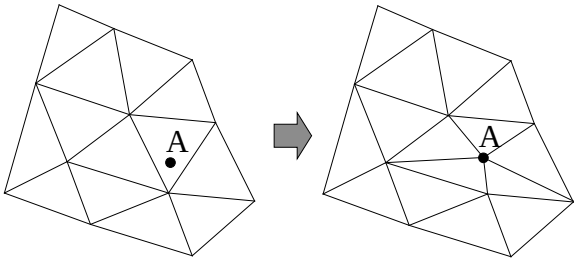

Figure 14. Example of Delaunay refinement, where point A is inserted.

5.3.1.3 Templates

A template refers to a specific decomposition of the triangle. One example is to decompose a single triangle into four similar triangles by inserting a new node at each of its edges as show in Figure 15. The equivalent tetrahedron template would decompose it into eight tetrahedra where each face of the tet has been decomposed into 4 similar triangles. To maintain a conforming mesh, additional templates can also be defined based on the number of edges that have been split. Staten[102] outlines the various templates needed to locally refine tetrahedra while maintaining a conforming mesh.

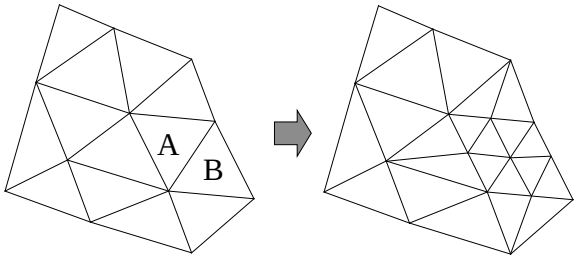

Figure 15. Example of local triangle refinement using a template where elements at A and B are refined

5.3.2 Quad/Hex Refinement

Because of the structured nature of quad and hex meshes, the point insertion and edge bisection methods are generally not applicable. The main methods used for quad and hex refinement involve decomposing the elements based on a set of predefined templates. Both Schneiders[103] and Staten[98] propose algorithms and a series of templates for element decomposition. An example of local quad refinement is shown in Figure 16. In order to maintain a conforming mesh, some quad and hex refinement schemes will often necessarily introduce triangle or alternate shaped elements including tetrahedra and pentahedra.

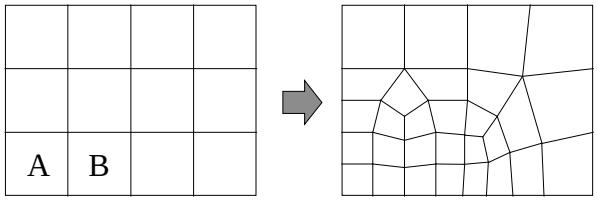

Figure 16. Example of local quad refinement where elements at A and B are refined by one half.

6. Conclusion

This survey has touched only briefly on some of the main issues and algorithms used in unstructured mesh generation. There are many more important aspects of unstructured mesh generation that were not addressed. Due to time and space constraints, it was not intended to be a comprehensive overview of the subject. Instead, it was the intent to focus on some of the more fundamental algorithms and procedures. Often times in the research and development of software, we tend to forget what has gone before us, or fail to look at what is already readily available. The most efficient way to provide new and innovative technology is to build on the accomplishments of others. We should recognize the innovations and creativity of others in the field and try to improve upon what has gone before.

References

1 Steven J. Owen, (1998) Meshing Software Survey, *web page:* http://www.andrew.cmu.edu/user/sowen/softsurv.html

2 Joe F. Thompson, (1985) "Numerical Grid Generation, Foundation and Applications", Elsevier. Posted on www at http://www.erc.msstate.edu/education/gridbook/index.html

3 Joe F. Thompson, (1996) "A Reflection on Grid generation in the 90s: Trends Needs and Influences", *5th International Conference on Numerical Grid Generation in Computational Field Simulations, Mississippi State University, April 1996*. pp.1029-1110

4 Steven J. Owen, Meshing Software Survey, Structured Grid Generation Software, *web page:* http://www.andrew.cmu.edu/user/sowen/software/structured.html

5 Mark A.Yerry and Mark S, Shephard, (1984) "Three-Dimensional Mesh Generation by Modified Octree Technique", International Journal for Numerical Methods in Engineering, vol 20, pp.1965-1990

6 Mark S. Shephard and Marcel K. Georges, (1991) "Three-Dimensional Mesh Generation by Finite Octree Technique", *International Journal for Numerical Methods in Engineering*, vol 32, pp. 709-749

7 Scientific Computation Research Center (SCOREC), Rensselaer Polytechnic Institute, *web site:* http://www.scorec.rpi.edu/

8 Stephen A. Vavasis, QMG *web site:* http://simon.cs.cornell.edu/Info/People/vavasis/qmg-home.html

9 Boris, N. Delaunay, (1934) "Sur la Sphere" *Vide. Izvestia Akademia Nauk SSSR*, VII Seria, Otdelenie Matematicheskii i Estestvennyka Nauk Vol 7 pp.793-800

10 C. L. Lawson, (1977) "Software for C^1 Surface Interpolation", *Mathematical Software III*, pp.161-194

11 David F. Watson, (1981) "Computing the Delaunay Tesselation with Application to Voronoi Polytopes", *The Computer Journal*, Vol 24(2) pp.167-172

12 Timothy J. Baker, (1989) "Automatic Mesh Generation for Complex Three-Dimensional Regions Using a Constrained Delaunay Triangulation", *Engineering with Computers*, vol 5, pp.161-175

13 N. P. Weatherill and O. Hassan (1994) "Efficient Three-dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints", *International Journal for Numerical Methods in Engineering*, vol 37, pp.2005-2039

14 P.L. George, F. Hecht and E. Saltel (1991) "Automatic Mesh Generator with Specified Boundary", *Computer Methods in Applied Mechanics and Engineering*, North-Holland, vol 92, pp.269-288

15 Paul L. Chew, (1989) "Guaranteed-Quality Triangular Meshes", *TR 89-983*, Department of Computer Science, Cornell University, Ithaca, NY, April 1989

16 Jim Ruppert, (1992) "A New and Simple Algorithm for Quality 2-Dimensional Mesh Generation". Technical Report UCB/CSD 92/694, University of California at Berkely, Berkely California

17 Jonathan Richard Shewchuk, (1996) "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator", http://www.cs.cmu.edu/~quake/triangle.html , 1996

18 S. Rebay, (1993) "Efficient Unstructured Mesh Generation by Means of Delaunay Triangulation and Bowyer-Watson Algorithm", *Journal Of Computational Physics*, vol. 106, pp.125-138

19 David L. Marcum and Nigel P. Weatherill, "Unstructured Grid Generation Using Iterative Point Insertion and Local Reconnection", *AIAA Journal*, vol 33, no. 9, pp.1619-1625, September 1995

20 David L. Marcum Solidmesh *web site:* http://www.erc.msstate.edu/thrusts/grid/solid_mesh/

21 H. Borouchaki, F. Hecht, E. Saltel and P. L. George, "Reasonably Efficient Delaunay Based Mesh Generator in 3 Dimensions", *Proceedings 4th International Meshing Roundtable*, pp.3-14, October 1995

22 TetMesh, GSH3D *web site:* http://www.simulog.fr/tetmesh/

23 P.L. George, , F. Hecht and E. Saltel, (1991) "Automatic Mesh Generator with Specified Boundary", *Computer Methods in Applied Mechanics and Engineering*, vol 92, pp.269-288

24 B. Joe, (1992) "Three-dimensional boundary-constrained triangulations", Artificial Intelligence, Expert Systems, and Symbolic Computing -- Proceedings of the 13th IMACS World Congress, ed. E. N. Houstis and J. R. Rice, Elsevier Science Publishers, pp. 215-222.

25 B. Joe, (1991) "GEOMPACK - A Software Package for the Generation of Meshes Using Geometric Algorithms", *Advances in Engineering Software*, vol 56, no. 13, pp.325-331

26 B. Joe, GEOMPACK, *anonymous ftp:* ftp://ftp.cs.ualberta.ca/pub/geompack

27 Rainald Lohner, Paresh Parikh and Clyde Gumbert, (1988) "Interactive Generation of Unstructured Grid for Three Dimensional Problems", *Numerical Grid Generation in Computational Fluid Mechanics '88*, Pineridge Press, pp.687-697

28 R. Lohner, (1996) "Progress in Grid Generation via the Advancing Front Technique", *Engineering with Computers*, vol 12, pp.186-210

29 S. H. Lo, (1991) "Volume Discretization into Tetrahedra-I. Verification and Orientation of Boundary Surfaces", *Computers and Structures*, vol 39, no. 5, pp.493-500

30 S. H. Lo, (1991) "Volume Discretization into Tetrahedra - II. 3D Triangulation by Advancing Front Approach", *Computers and Structures*, vol 39, no 5, pp.501-511

31 ANSYS *web site:* http://www.ansys.com

32 Shahyar Pirzadeh, (1993) "Unstructured Viscous Grid Generation by Advancing-Layers Method", AIAA-93-3453-CP, AIAA, pp.420-434

33 TetrUSS, Tetrahedral Unstructured Software System (includes VGRID mesh generator), *web site:* http://ad-www.larc.nasa.gov/tsab/tetruss/

34 Steven J. Owen, Meshing Software Survey, Quadrilateral Mesh Generation Software: *web page:* http://www.andrew.cmu.edu/user/sowen/software/quadrilateral.html

35 Steven J. Owen, Meshing Software Survey, Hexahedra Mesh Generation Software: *web page:* http://www.andrew.cmu.edu/user/sowen/software/quadrilateral.html hexahedra.html

36 W.A. Cook, and W.R. Oakes (1982). "Mapping Methods for Generating Three-Dimensional Meshes", *Computers in Mechanical Engineering*, August 1982, pp. 67-72

37 CUBIT Mesh Generation Toolkit, *web site:* http://endo.sandia.gov/SEACAS/CUBIT/Cubit.html

38 Timothy J. Tautges, , Shang-sheng Liu, Yong Lu, Jason Kraftcheck, Rajit Gadh, (1997) "Feature Recognition Applications in Mesh Generation", *AMD-Vol. 220 Trends in Unstructured Mesh Generation*, ASME, pp.117-121

39 Shang-Sheng Liu, and Rajit Gadh, (1996) "Basic LOgical Bulk Shapes (BLOBS) for Finite Element Hexahedral Mesh Generation", *5th International Meshing Roundtable*, pp.291-306

40 Scott A. Mitchell, (1997) "High Fidelity Interval Assignment", *Proceedings, 6th International Meshing Roundtable*, pp.33-44

41 David R. White, (1995). "Automated Hexahedral Mesh Generation by Virtual Decomposition", *Proceedings, 4th International Meshing Roundtable*, Sandia National Laboratories, pp.165-176

42 Matthew L. Staten, Scott A. Canann, and Steve J. Owen (1998) "BMSWEEP: Locating Interior Nodes During Sweeping", *7th International Meshing Roundtable*

43 Mingwu, Lai, Steven E. Benzley, Greg Sjaardema and Tim Tautges (1998) "A Multiple Source and Target Sweeping Method for Generating All-Hexahedral Finite Element Meshes", *5th International Meshing Roundtable*, pp.217-228

44 Ted D. Blacker, (1996). "The Cooper Tool", *Proceedings, 5th International Meshing Roundtable*, pp.13-29

45 Fluent, Gambit *web site:* http://www.fluent.com/software/gambit/gambit.htm

46 S.H. Lo, (1989). "Generating Quadrilateral Elements on Plane and Over Curved Surfaces", *Computers and Structures*, Vol.31(3), pp.421-426

47 Bruce P Johnston, John M. Sullivan Jr. and Andrew Kwasnik (1991). "Automatic Conversion of Triangular Finite Element Meshes to Quadrilateral Elements", *International Journal for Numerical Methods in Engineering*, Vol.31, pp.67-84

48 C.K Lee, and S.H. Lo (1994). “A New Scheme for the Generation of a Graded Quadrilateral Mesh,” *Computers and Structures*, Vol.52 pp.847-857

49 Steven J. Owen, Matthew L. Staten, Scott A. Canann and Sunil Saigal, (1998) “Advancing Front Quad Meshing Using Local Triangle Transformations”, *Proceedings, 7th International Meshing Roundtable*

50 Peggy L. Baehmann, Scott L. Wittchen, Mark S. Shephard, Kurt R. Grice and Mark A. Yerry, (1987). “Robust Geometrically-based, Automatic Two-Dimensional Mesh Generation,” *International Journal for Numerical Methods in Engineering*, Vol.24, pp.1043-1078

51 J.A. Talbert, and A.R. Parkinson, (1991). “Development of an Automatic, Two Dimensional Finite Element Mesh Generator using Quadrilateral Elements and Bezier Curve Boundary Definitions”, *International Journal for Numerical Methods in Engineering*, Vol.29 pp.1551-1567

52 Soo-Won Chae, and Jung-Hwan Jeong, (1997). “Unstructured Surface Meshing Using Operators”, *Proceedings, 6th International Meshing Roundtable*, pp.281-291

53 Dietrich, Nowottny, (1997). “Quadrilateral Mesh Generation via Geometrically Optimized Domain Decomposition”, *Proceedings, 6th International Meshing Roundtable*, pp.309-320

54 T. K. H. Tam and C. G. Armstrong (1991). “2D Finite Element Mesh Generation by Medial Axis Subdivision”, *Advances in Engineering Software*, Vol.13, pp.313-324

55 Barry Joe, (1995). “Quadrilateral Mesh Generation in Polygonal Regions”, *Computer Aided Design*, Vol.27, pp.209-222

56 Barry Joe, (1986). “Delaunay Triangular Meshes in Convex polygons”, *SIAM J. Sci. Stat. Comput*., Vol.7, pp.514-539

57 J.Z. Zhu, O.C. Zienkiewicz, E. Hinton and J. Wu (1991). “A New Approach to the Development of Automatic Quadrilateral Mesh Generation,” *”, International Journal for Numerical Methods in Engineering*, Vol.32 pp.849-866

58 Ted D. Blacker, and Michael B. Stephenson (1991). “Paving: A New Approach to Automated Quadrilateral Mesh Generation”, *International Journal for Numerical Methods in Engineering*, Vol 32 pp.811-847

59 Roger J. Cass, , Steven E. Benzley, Ray J. Meyers and Ted D. Blacker (1996). “Generalized 3-D Paving: An Automated Quadrilateral Surface Mesh Generation Algorithm”, *International Journal for Numerical Methods in Engineering*, Vol. 39 pp.1475-1489

60 David R. White and Paul Kinney (1997). “Redesign of the Paving Algorithm: Robustness Enhancements through Element by Element Meshing,” *Proceedings, 6th International Meshing Roundtable*, Sandia National Laboratories, pp. 323-335

61 MacNeal-Schwendler Home Page, *web site:* http://www.macsch.com/

62 Takeo Taniguchi, Tomoaki Goda, Harald Kasper and Werner Zielke, (1996) "Hexahedral Mesh Generation of Complex Composite Domain", *5th International Conference on Grid Generation in Computational Field Simmulations*, Mississippi State University. pp 699-707

63 Robert Schneiders, (1996) “A Grid-Based Algorithm for the Generation of Hexahedral Element Meshes”, *Engineering With Computers.* Vol.12 pp.168-177

64 F. Weiler, R. Schindler and R. Schneiders, (1996) “Automatic Geometry-Adaptive Generation of Quadrilateral and Hexahedral Element Meshes for the FEM”, *Proceedings, 5th International Conference on Numerical Grid Generation in Computational Field Simmulations*, Mississippi State University, pp.689-697

65 Robert Schneiders, (1997) “An Algorithm for the Generation of Hexahedral Element Meshes Based On An Octree Technique”, *Proceedings, 6th International Meshing Roundtable*, Abstract only pp.195-196

66 Monika Wierse, Jean Cabello and Yoshihiko Mochizuki, (1998) “Automatic Grid Generation with HEXAR”, *Proceedings 6th International Conference on Numerical Grid Generation in Computational Field Simulations, ed. M. Cross et. al*., University of Greenwich, UK., pp. 843-852

67 MARC *web site*: http://toto.marc.com/

68 T.S. Li, R.M. McKeag and C.G. Armstrong, (1995) “Hexahedral Meshing Using Midpoint Subdivision and Integer Programming”, *Computer Methods in Applied Mechanics and Engineering*, Vol.124, pp.171-193

69 M.A. Price and C.G. Armstrong, (1995) “Hexahedral Mesh Generation by Medial Surface Subdivision: Part I”, *International Journal for Numerical Methods in Engineering*. Vol 38(19), pp.3335-3359

70 M.A. Price and C.G. Armstrong, (1997) “Hexahedral Mesh Generation by Medial Surface Subdivision: Part II,” *International Journal for Numerical Methods in Engineering*. Vol 40, pp.111-136

71 FEGS *web site:* http://fegs.co.uk

72 Solidpoint *web site:* http://www.99main.com/~diholm/

73 Scott A. Canann, (1991) “Plastering and Optismoothing: New Approaches to Automated, 3D Hexahedral Mesh Generation and Mesh Smoothing,” Ph.D. Dissertation, Brigham Young University, Provo, UT.

74 Ted D. Blacker and R. J. Myers, (1993). “Seams and Wedges in Plastering: A 3D Hexahedral Mesh Generation Algorithm,” Engineering With Computers, Vol.2, pp.83-93

75 Timothy J. Tautges, Ted Blacker and Scott Mitchell, (1996) “The Whisker-Weaving Algorithm: A Connectivity Based Method for Constructing All-Hexahedral Finite Element Meshes,” *International Journal for Numerical Methods in Engineering*, Vol.39, pp.3327-3349

76 Peter Murdoch, and Steven E. Benzley, (1995) “The Spatial Twist Continuum”, *Proceedings, 4th International Meshing Roundtable*, Sandia National Laboratories, pp.243-251

77 Steven J. Owen, Scott A. Canann and Sunil Saigal, (1997) "Pyramid Elements for Maintaining Tetrahedra to Hexahedra Conformability", *AMD-Vol. 220 Trends in Unstructured Mesh Generation*, ASME, pp.123-129

78 Phillip Tuchinsky, M., Brett W. Clark, (1997) “The Hex-Tet, Hex-Dominant Automesher: An Interim Progress Report”, *Proceedings, 6th International Meshing Roundtable,* pp.183-193

79 Weidong Min, (1997) “Generating Hexahedron-Dominant Mesh Based on Shrinking-Mapping Method”, *Proceedings, 6th International Meshing Roundtable*, pp.171-182

80 R. T. Farouki, (1997) “Optimal paramaterizations,” *Comuter Aided Geometric Design*, vol. 14 153-168

81 Paul-Louis George, and Houman Borouchaki (1998) *Delaunay Triangulation and Meshing: Application to Finite Elements*, Hermes, France, 413 p.

82 Hao Chen and Jonathan Bishop (1997) “Delaunay Triangulation for Curved Surfaces”, *Proceedings, 6th International Meshing Roundtable*, pp.115-127

83 J. C. Cuilliere, (1998) "An adaptive method for the automatic triangulation of 3D parametric surfaces", *Computer-Aided Design*, vol 30, no. 2, pp.139-149

84 Joseph R. Tristano, Steven J. Owen and Scott A. Canann, (1998) “Advancing Front Surface Mesh Generation in Parametric Space Using a Riemannian Surface Definition”, *7th International Meshing Roundtable*

85 Lau, T.S. and S.H. Lo, (1996) "Finite Element Mesh Generation Over Analytical Surfaces", *Computers and Structures*, vol 59, no. 2, pp.301-309

86 Lau, T.S., S. H. Lo and C. K. Lee, (1997) "Generation of Quadrilateral Mesh over Analytical Curved Surfaces", *Finite Elements in Analysis and Design*, vol 27, pp.251-272

87 Field, D. A.(1988), “Laplacian smoothing and Delaunay triangulations”, *Commuications in Applied Numerical Methods.*, vol. 4, pp. 709-712.

88 Scott A. Canann, Joseph R. Tristano and Matthew L. Staten, (1998) “An Approach to Combined Laplacian and Optimization-Based Smoothing for Triangular, Quadrilateral, and Quad-Dominant Meshes,” *Proceedings, 7th International Meshing Roundtable*

89 Lori Freitag, Mark Jones, and Paul Plassmann, (1995) "An Efficient Parallel Algorithm for Mesh Smoothing", *Proceedings, 4th International Meshing Roundtable*, pp.47-58

90 Lori A. Freitag, (1997) "On Combining Laplacian and Optimization-Based Mesh Smoothing Techniques", *AMD-Vol. 220 Trends in Unstructured Mesh Generation*, pp.37-43

91 R. Lohner, K. Morgan and O. C. Zienkiewicz, (1986) “Adaptive Grid Refinement for Compressible Euler Equations”, *Accuracy Estimates and Adaptive refinements in Finite Element Computations, I. Babuska et. al. eds.*, Wiley, pp. 281-297

92 Kenji Shimada, Atsushi Yamada and Takayuki Itoh, (1997) “Anisotropic Triangular Meshing of Parametric Surfaces via Close Packing of Ellipsoidal Bubbles”, *Proceedings, 6th International Meshing Roundtable*, pp.375-390

93 Frank, J Bossen and Paul S. Heckbert (1996) "A Pliant Method for Anisotropic Mesh Generation", *Proceedings, 5th International Meshing Roundtable*, pp.63-76

94 Ahmed Z.I.Salem, Scott A. Canann, and Sunil Saigal, (1997) “Robust Distortion Metric for Quadratic Triangular 2D Finite Elements”, *AMD-Vol. 220 Trends in Unstructured Mesh Generation*, pp.73-80

95 Freitag, Lori A. and Carl Ollivier-Gooch, (1997) "Tetrahedral Mesh Improvement Using Swapping and Smoothing", *International Journal for Numerical Methods in Engineering*, vol. 40, pp.3979-4002

96 Barry Joe, (1995) "Construction of Three-Dimensional Improved-Quality Triangulations Using Local Transformations", *Siam J. Sci. Comput.*, vol 16, pp.1292-1307

97 S. A. Canann, S. N. Muthukrishnan and R. K. Phillips (1996) "Topological Refinement Procedures for Triangular Finite Element Meshes", *Engineering with Computers*, vol 12, pp.243-255

98 Staten, Matthew L. and Scott A. Canann, (1997) "Post Refinement Element Shape Improvement for Quadrilateral Meshes", *AMD-Vol. 220 Trends in Unstructured Mesh Generation*, pp.9-16

99 Paul Kinney, (1997) "CleanUp: Improving Quadrilateral Finite Element Meshes", *Proceedings, 6th International Meshing Roundtable*, pp.437-447

100 Scott A Mitchell and Timmothy J. Tautges, (1995) "Pillowing Doublets: Refining A Mesh to Ensure That Faces Share At Most One Edge", *Proceedings, 4th Internatinal Meshing Roundtable*, pp.231-240, October 1995

101 Rivara, Maria-Cecilia, (1997) "New Longest-Edge Algorithms For the Refinement and/or Improvement of Unstructured Triangulations", *International Journal for Numerical Methods in Engineering*, vol. 40, pp.3313-3324

102 Staten, M.L. and N.L. Jones (1997) "Local Refinement of Three-Dimensional Finite Element Meshes", *Engineering with Computers*, vol 13, pp.165-174

103 R. Schneiders, (1996) "Refining Quadrilateral and Hexahedral Element Meshes", *5th International Conference on Numerical Grid Generation in Computational Field Simulations*, Mississippi State University, pp.679-688

Appendix

Meshing Software Survey

A survey was conducted during September 1998 of current mesh and grid generation software. Over 100 surveys were mailed to software vendors, research labs and educational institutions. This list is definitely not all-inclusive, but I believe is fairly representative of what is currently available, as well as what is state-of-the-art. Only those responding to the survey are included here. While the emphasis of the survey is unstructured codes, there are also a considerable number of structured codes included. The codes range from simple research codes that are used only by a few people, to commercial products incorporated into sophisticated analysis packages. An online and up-to-date copy of this survey is available on the web as part of the *Meshing Research Corner* web site maintained at Carnegie Mellon University by the author at the following URL: http://www.andrew.cmu.edu/user/sowen/softsurv.html

Survey Statistics

Total number of software products in survey	81
Element Shapes	
Number of products that generate triangles	52
Number of products that generate quadrilaterals (non-structured codes)	25
Number of products that generate tetrahedra	39
Number of products that generate hexahedra (non-structured codes)	20
Number of products that generate structured quads or hexes	21
Availability	
Number of Public Domain Codes	34
Number of Research Codes	24
Number of Commercial Products	33
Number of Products Available as Stand-Alone Meshing Generator	40

Number of Products providing Source Code	21
Engineering Discipline	
Number of Products used for Structural Applications	23
Number of Products used for CFD (Fluids) Applications	47
Number of Products used for EMAG (Electro-magnetic) Applications	23
Number of Products used for Thermal Applications	9
Number of Products used for Environmental Applications	12
Tri/Tet Meshing Algorithm	
Number of tri/tet codes using some form of Delaunay Algorithm	37
Number of tri/tet codes using some form of Advancing Front Algorithm	23
Number of tri/tet codes using an Octree Algorithm	4
Quad/Hex Meshing Algorithm	
Number of quad/hex codes using an Advancing_Front Algorithm	9
Number of quad/hex codes using a Medial Axis/Surface Algorithm	2
Number of quad/hex codes using an indirect Algorithm (combine triangles)	5
Number of quad/hex codes using a Sweeping or Extrusion Algorithm	8
Number of quad/hex codes using a Mapped Meshing Algorithm	11
Other Features	
Number of Products providing Boundary Layer definition	17
Number of Products providing Adaptivity	18
Number of Products providing Anisotropy (stretched elements)	16
Number of Products providing Refinement	27
Number of Products providing Mesh Improvement	8

Software Products

The following is the complete list of software included in the survey ordered alphabetically by product name. Also included is a contact indvidual and web site. A separate web page for each product listing basic features and comments provided by the contact is also provided on-line.

2-D GWADAPT, University of Nevada Las Vegas/Nevada Center for Advanced Computational Methods (NCACM), Dr. Yitung Chen or Dr. Laxmi Gewali, nccm_www@aurora.nscee.edu, http://www.unlv.edu/Research_Centers/NCACM/

3DMAGGS (Three-Dimensional Multi-block Advanced Grid Generation System), NASA Langley Research Center/Lockheed Martin Engineering & Sciences, Stephen J. Alter, Charles G. Miller, s.j.alter@larc.nasa.gov, http://ab00.larc.nasa.gov/~salter/3DMAGGS.html

ADMesh, Varlog, Anthony D. Martin, amartin@varlog.com, http://www.varlog.com/products/admesh

AFLR2, Engineering Research Center for Computational Field Simulation, Mississippi State University, David L. Marcum, marcum@erc.msstate.edu,

AFLR3, Engineering Research Center for Computational Field Simulation, Mississippi State University, David L. Marcum, marcum@erc.msstate.edu, http://www.erc.msstate.edu/thrusts/grid/solid_mesh

Algor Finite Element and Event Simulation Software, Algor, Inc., Julie Halapchuk, Marketing Communications, info@algor.com, http://www.algor.com

Altair Hypermesh, Altair Computing, Inc., George Christ, gjc@altair.com, http://www.altair.com/Products/HyperMesh.html

AMESH - Multi-Region Finite Element Meshing for Casting Processes, EKK, Inc, shawnekk@mail.ic.net, ekk@mail.ic.net, http://ic.net/~ekk/amesh.htm

ANSYS, ANSYS, Inc., Local ANSYS Support Distributor, , http://www.ansys.com

Argus ONE (Argus Open Numerical Environments), Argus Interware, Inc., Joshua Margolin, margolinj@argusint.com, http://www.argusint.com http://www.argusint.com/MeshGeneration.html

AVL FAME, AVL LIST GmbH, Anton Plimon, Robert Schmitz (North America), ap@avl.com schmitz@avlna.com, http://www.avl.com/html/69.htm

BAMG, INRIA, Frédéric Hecht, Frederic.Hecht@inria.fr, http://www-rocq.inria.fr/gamma/cdrom/www/bamg/eng.htm

BL2D, INRIA Rocquencourt, B.P. 105, 78153 Le Chesnay Cedex (France), Patrick Laug, Houman Borouchaki, Patrick.Laug@inria.fr Houman.Borouchaki@univ-troyes.fr, http://www-rocq.inria.fr/gamma/cdrom/www/bl2d/eng.htm

CADfix, FEGS Ltd., John Rawlinson, john.rawlinson@fegs.co.uk, http://www.fegs.co.uk/index.html http://fegs.co.uk

CAF2D / GENMESH, Yeungnam Univ., Dept. of Mechanical Engineering, CAF Lab or OnDemand Soft (venture company), Professor Jong-Youb Sah, jysah@ynucc.yeungnam.ac.kr, http://caflab.yeungnam.ac.kr/genmesh.html

CAGI, ERC, Mississippi State University, Bharat Soni, bsoni@erc.msstate.edu, http://www.erc.msstate.edu/thrusts/grid/cagi/index.html

Cart3D, NASA Ames Research Center, Michael J. Aftosmis, Cathy Pochel (licensing), aftosmis@nas.nasa.gov cpochel@mail.arc.nasa.gov, http://george.arc.nasa.gov/~aftosmis/cart3d/

CFD-GEOM, CFD Research Corporation, John Whitmire, jbw@cfdrc.com, http://www.cfdrc.com http://www.cfdrc.com/datab/Software/geom/cfdgeom.html

Chalmesh, Chalmers University of Technology, Dept. of Naval Arch. & Ocean Eng., Anders Petersson, andersp@na.chalmers.se, http://www.na.chalmers.se/~andersp/chalmesh.html

COG, WIAS Berlin, Ilja Schmelzer, schmelzer@wias-berlin.de, ftp://ftp.wias-berlin.de/pub/cog/index.html

CSCMDO, Computer Sciences Corporation/NASA LaRC GEOLAB, William T. Jones, w.t.jones@LaRC.nasa.gov, http://geolab.larc.nasa.gov/CSCMDO

CUBIT Mesh Generation Toolkit, Sandia National Laboratories, David R. White, drwhite@sandia.gov, http://endo.sandia.gov/SEACAS/CUBIT/Cubit.html

delaundo, ipol, Von Karman Institute, Brussels, Belgium, Jens-Dominik Müller, muller@comlab.ox.ac.uk, http://www.cerfacs.fr/~muller/grids.html http://www.comlab.ox.ac.uk/oucl/people/jens-dominik.muller.html

DesignSpace, ANSYS Inc., Local ANSYS Support Distributor, , http://www.designspace.com/

EasyMesh, University of Trieste, D.I.N.M.A, Bojan Niceno, niceno@wt.tn.tudelft.nl, http://www-dinma.univ.trieste.it/~nirftc/research/easymesh/

EMC2, INRIA, Frédéric Hecht and Eric Saltel, Frederic.Hecht@inria.fr, http://www-rocq.inria.fr/gamma/cdrom/www/emc2/eng.htm

FELISA, NASA and MIT, Karen Bibb, NASA Langley Research Center, k.l.bibb@larc.nasa.gov, http://ab00.larc.nasa.gov/~kbibb/felisa.html

FEMGV (Version 5.1-01), Femsys Ltd., Steve Attwood, Derek Styles, info@femsys.co.uk s.attwood@femsys.co.uk d.styles@femsys.co.uk, http://www.femsys.co.uk/

GENIE++, ERC, Mississippi State University, Professor Bharat K. Soni, bsoni@erc.msstate.edu, http://www.erc.msstate.edu/thrusts/grid/genie/index.html

GeoCad, Industrial Research Ltd. (NZ) / Geothermal Energy Research and Development (Japan), Dr Stephen P White, s.white@irl.cri.nz, http://tui.grace.cri.nz/~steve/

geomagic Wrap, Raindrop Geomagic, Inc., Ping Fu, Chantelle Hougland, inquiry@geomagic.com, http://www.geomagic.com/

Geompack, University of Alberta, Computer Science, Barry Joe, bjoe@netcom.ca, ftp://ftp.cs.ualberta.ca:/pub/geompack/

GMS (Groundwater Modeling System), Environmental Modeling Research Laboratory (formerly the ECGL), Norm Jones, njones@et.byu.edu, http://www.emrl.byu.edu/

GMSH, Ecole Polytechnique de Montreal & University of Liege, Jean-François Remacle, remacle@meca.polymtl.ca, http://www.meca.polymtl.ca/~remacle/Mesh.html

Gridgen, Pointwise, Inc, Rick Matus, gridgen@pointwise.com, http://www.pointwise.com/

Gridomatic, VKI, UCDavis, Cislunar Aerospace, Dave Banks, dbanks@cislunar.com, http://mae.engr.ucdavis.edu/CFD/dbanks/Hybrid/gridomatic.html

gridpak, Rutgers University, Kate Hedstrom, kate@ahab.rutgers.edu, http://marine.rutgers.edu/po/gridpak.html

GridPro/AZ-Manager 3000, CLE GmbH and PDC, New York, Dr. Jochem Hauser (CLE), Dr. Peter Eiseman (PDC), jh@cle.de, http://www.cle.de/cfd/products/GridPro/index.html

GridTool, GEOLAB at NASA Langley Reseach center, Pat Kerr, P.A.KERR@LaRC.NASA.GOV, http://geolab.larc.nasa.gov/GridTool/

GRUMMP, University of British Columbia, Carl Ollivier-Gooch, cfog@mech.ubc.ca, http://tetra.mech.ubc.ca/GRUMMP

GUM-B, Engineering Research Center, Miss. State, Mike Remotigue, remo@erc.msstate.edu, http://www.erc.msstate.edu/thrusts/grid/index.html

ICEM CFD, ICEM CFD Engineering, Kristian Debus, Support and Releases, debus@icemcfd.com, http://www.icemcfd.com/

Javamesh, University of Pittsburgh, Steven Lin, steven@leetide.net, http://www.steven.pop.net.tw/javamesh/

LaGriT (Los Alamos Gridding Toolbox), Los Alamos National Laboratory, Carl Gable or Denise George, gable@lanl.gov, dgeorge@lanl.gov, http://www.t12.lanl.gov/~lagrit/

MAFIA-M, CST, Marko Walter, info@cst.de, http://www.cst.de/

MEGA (Meshing Environment for Geometry-based Analysis), Scientific Computation Research Center, Rensselaer Polytechnic Institute, Mark Shephard, Shephard@scorec.rpi.edu, http://www.scorec.rpi.edu/

MegaCads (Multiblock-Elliptic-Grid-generation-And-CAD-System), DLR, Institute of Design Aerodynamics and MEGAFLOW project, Olaf Brodersen and Prof. Dr. Horst Körner, megacads@dlr.de Olaf.Brodersen@dlr.de, http://www.bs.dlr.de/sm/ea/Proj_MEGAFLOW/MegaCadsOverview.html

Mentat, MARC Analysis Research Corporation, Jon Bishop (Mentat Manager), jon@marc.com, http://toto.marc.com/

MESH, ISE Integrated Systems Engineering AG, Zurich, ISE support, support@ise.ch, http://www.ise.ch/mesh.htm

Mesh++, Center for Advanced Studies, Research and Development in Sardinia (CRS4), Gianluigi Zanetti, zag@crs4.it, http://www.crs4.it/Areas/cfd/GRID_GENERATION/link1.html

Mesh-Maker, Environment Centre, University of Leeds, Jason Lander, jason@lec.leeds.ac.uk, http://www.lec.leeds.ac.uk/%7Ejason/Mesh-Maker/

mesh2d, Scientific Compuattional Research Center, SCOREC, B. Kaan Karamete, kaan@scorec.rpi.edu, http://scorec.rpi.edu/~kaan/

MG (Mesh Generator), TeCGraf - The Computer Graphics Tecnology Group of PUC-Rio, Luiz Cristovão Gomes Coelho, lula@tecgraf.puc-rio.br, http://www.tecgraf.puc-rio.br/~lula/mg/mg.html

MTC, SCC/ CEMEF, Philippe DAVID, phdavid@scconsultants.com, http://www.scconsultants.com/

NETGEN, Institut of Analysis and Numerical Mathematics, Johannes Kepler University, Linz, Austria, Joachim Schöberl, joachim@numa.uni-linz.ac.at, http://nathan.numa.uni-linz.ac.at/netgen/usenetgen.html

OVERGRID, MCAT, Inc. at NASA Ames Research Center, William M. Chan, wchan@nas.nasa.gov, http://halfdome.arc.nasa.gov/cfd/CFD4/og_man.html

Overture, Centre for Applied Scientific Computing, Lawrence Livermore National Laboratory., Bill Henshaw, overture-support@c3serve.c3.lanl.gov, http://www.c3.lanl.gov/Overture

Preproc, Numerical Methods Laboratory, "POLITEHNICA" University of Bucharest, Tiberiu Chelcea, tibi@lmn.pub.ro, http://www.lmn.pub.ro/~tibi/mesh_gen/mesh_gen.html

PRISM, NASA Ames Research Center, Shishir Pandya, pandya@nas.nasa.gov, http://www.engr.ucdavis.edu/~spandya/prism.html

Qhull, The Geometry Center, University of Minneapolis, Brad Barber, bradb@geom.umn.edu, http://www.geom.umn.edu/locate/qhull

QMG, Cornell University, Stephen A. Vavasis, vavasis@cs.cornell.edu, http://simon.cs.cornell.edu/Info/People/vavasis/qmg-home.html

QUAD - GEN, Computational Mechanics Australia Pty. Ltd, Dr. Alexander Tsvelikh, Director, comecau@bigpond.com sashat@ozemail.com.au, http://www.ozemail.com.au/~sashat/ http://www.ozemail.com.au/~sashat/quadgen.htm

QuikGrid, Perspective Edge Software, John Coulthard, w.j.coulthard@ubc.ca, http://www.interchg.ubc.ca/coulthrd/pes.html

samm, adapco, Wayne R. Oaks, wayne@adapco.com, http://www.adapco.com/samm.html

SCP Grid Library, Scalable Concurrent Programming Laboratory, Syracuse University, Marc Rieffel, marc@scp.syr.edu, http://www.scp.syr.edu/~marc/grid

SD (Super Delaunay) librarySDI (Super Delaunay Indexed) library, David Kornmann, David Kornmann, david@iki.fi, http://www.iki.fi/~david

SKY/Mesh2, Skyblue Systems, James Joseph, sales@skybluesystems.com, http://skybluesystems.com/mesh2.htm

SolidMesh, Engineering Research Center for Computational Field Simulation, Mississippi State University, David L. Marcum, marcum@erc.msstate.edu, http://www.erc.msstate.edu/thrusts/grid/solid_mesh

TetMesh GHS3D, SIMULOG, Mark Loriot, loriot@simulog.fr, http://www.simulog.fr/tetmesh/

TIGER-II Turbomachinery Grid Generation System, Version 2.01, Catalpa Research, Inc., Dr. Alan M. Shih, shih@catalpa.net, http://www.catalpa.net/

TMG (triangular mesh generator), Istituto di Analisi Numerica (CNR) of Pavia, Dipartimento di Matematica, University of Milano, Maurizio Paolini, m.paolini@dmf.bs.unicatt.it, http://www.dmf.bs.unicatt.it/~paolini/tmg/

TOAST, University College London, Dr Martin Schweiger, Dr. Simon Arridge, martins@medphys.ucl.ac.uk S.Arridge@cs.ucl.ac.uk, http://www.medphys.ucl.ac.uk/toast/index.htm

Triangle: A Two-Dimensional Quality Mesh Generator and Delaunay Triangulator., Carnegie Mellon University, Jonathan Richard Shewchuk, jrs@cs.cmu.edu, http://www.cs.cmu.edu/~quake/triangle.html

TriGrid, Channel Consulting Ltd, Adrian Dolling, adolling@channel.bc.ca, http://www.channel.bc.ca

TrueGrid, XYZ Scientific Applications, Inc., Matthew Koebbe, Ph.D., xyz@netcom.com, http://www.truegrid.com/

TRUMPET, NASA Lewis Research Center, Philip C. E. Jorgenson, jorgenson@lerc.nasa.gov, http://www.lerc.nasa.gov/WWW/microbus/cese/jorgenson/jorgenson.html

TurboMesh, SolidPoint, David Holmes, diholm@99main.com, http://www.99main.com/~diholm/

VGM, NASA Langley Research Center/Lockheed Martin Engineering & Sciences, Stephen J Alter, Charles Miller, s.j.alter@larc.nasa.gov, http://ab00.larc.nasa.gov/~salter/VGM-web.html

VGRID, VGRIDns (Navier-Stokes version), NASA Langley Research Center, Shahyar Z. Pirzadeh, s.pirzadeh@larc.nasa.gov, http://ad-www.larc.nasa.gov/tsab/tetruss/

Xcog, Chalmers University of Technology, Dept. of Naval Arch. and Ocean Eng., Anders Petersson, andersp@na.chalmers.se, http://www.na.chalmers.se/~andersp/xcog/xcog.html

XGEN, Charles University, Prague, Pavel Solin, solin@karlin.mff.cuni.cz, http://www.karlin.mff.cuni.cz/katedry/knm/xgen/