PERGAMON

Computers & Graphics 23 (1999) 59–72

COMPUTERS & GRAPHICS

Computer Graphics in India

Zeus: surface modeling, surface grid generation, tetrahedral volume discretization

Dinesh Shikhare[a], S. Gopalsamy[a], T. Sathi Reddy[a], Ashwini Patgawkar[a], Satyashree Mahapatra[a], S.P. Mudur[a,*], K.P. Singh[b], Indira Narayanswamy[b], Laxmi Ravishankar[b]

[a]*Graphics and CAD Division, National Centre for Software Technology, Juhu, Mumbai 400049, India*
[b]*Aeronautical Development Agency, Bangalore, India*

Abstract

This paper describes an interactive aircraft surface engineering and grid generation system Zeus, that has been developed and one that has evolved with actual use over the last decade. Zeus is an interactive system for the design, specification and visualization of aircraft geometries. The package supports surfaces defined in a number of representations including planes, spheres, cylinders, bilinear surfaces, NURBS and triangulated surfaces. It includes fairly sophisticated geometric editing and rendering operations. It derives its main strength from the fact that it includes a comprehensive suite of tools for surface and volume grid generation and their visualization. During these ten years of development, there have been a number of innovative computational techniques and algorithms that have been devised, implemented and tested. This paper is primarily about the Zeus system and these new innovative techniques.

Keywords: Geometric modeling; Grid generation; Triangulation; 3D Delaunay; Data-structures

1. Background

Zeus is an interactive system for the design, specification and visualization of aircraft geometries. The system incorporates a comprehensive suite of tools that gives facilities for surface modeling, surface grid generation and volume grid generation. These are basic requirements for CFD analysis and iterative design of complex systems like aircrafts. Zeus has multiple methods for creation of curves and surfaces. The package supports surfaces defined in a number of representations including planes, spheres, cylinders, bilinear surfaces, NURBS and triangulated surfaces. It includes fairly sophisticated geometric editing and rendering operations. It derives its main strength from the fact that it includes a comprehensive suite of tools for surface and volume gridding and their visualization. It provides robust geometric algorithms such as surface–surface intersection, projection of points/curves onto surfaces, surface tessellation and tetrahedralization. Zeus also has an effective user interface for easy access to the functionality of such a tool set. The system is a culmination of about a decade's sustained development work toward geometric modeling and grid generation of both trimmed surfaces and volumetric spaces. Early developments were reported in [1, 2]. During these ten years of development, there have been a number of innovative computational techniques and algorithms that have been devised, implemented and tested.

Over the last two decades there have been many commercial systems dealing quite effectively with doubly curved surface geometries [3–6]. The aircraft industry needs have been one of the major driving forces. More

*Corresponding author. Tel.: 0091 22 6201 606; fax: 0091 22 6210 139; e-mail: mudur@saathi.ncst.ernet.in.

0097-8493/99/$ – see front matter
PII: S0097-8493(98)01116-2

and more mathematical techniques continue to be evolved for dealing efficiently and with increasingly complex surface shapes and configurations. Most commercial systems are general purpose and try and cater to the requirements of a large number of different engineering user groups. The systems are large, complex and expensive. However, not many of them include specific methods for grid generation and analysis. Such systems expect that the surface geometries would be suitably exported to separate grid generation programs. The need for Zeus arose out of the requirement that a single system should be able to deal with surface specification and also with grid generation facilities.

From its inception it was clear that the development of Zeus would be a continuing effort. Hence a number of design goals were kept in mind over all these years. These are listed below and over the years we have found that they have helped ensure the life of Zeus for such a long time.

1. *Application specific*: It caters specifically to the aircraft surface geometry, i.e., the mathematical representations, parametric limits, specification and editing operations, user-interface terminology, visual feedback mechanisms and storage structures are determined with aircraft surfaces in mind.
2. *Robustness*: Efficient and proven mathematical techniques have to be incorporated. Correctness and robustness of the algorithms are of definite importance.
3. *Effective user-interface*: Designed to be able to quickly edit, visualize, discretize the surface geometry.
4. *Scalability*: Implemented with appropriate data structures that can deal with increase in geometric components. Volumetric grids can typically include hundreds of thousands of tetrahedra.
5. *Extensibility*: Desired with a well-defined geometric kernel that can be easily extended to support newer geometries.
6. *Import/export of data to other complementary packages*: Supports interoperability with other commercial CAD/CAE packages in use for similar user domains.
7. *Use of good software engineering principles*: Should not be very dependent on the designers or implementors at any later stage.

2. Model of usage

The primary goal of Zeus being surface and volume grid generation for CFD analysis of aircraft geometry, the intended model of usage would typically have the form described below (see Fig. 1).

- *Surface Modeling*: Surface geometry is either imported from packages like CATIA or is constructed from section curves by lofting a surface or is defined by control points. Zeus uses B-spline curves and surfaces for modeling surface geometry. Once a surface is constructed in the system by any of the above techniques, it can be edited by editing its control mesh.

  To generate surface grids on the surface patches of interest, the surfaces must be intersected and trimmed against each other or against planes to clip away the unwanted parts of the modeled surfaces. The trimmed regions so formed are used for surface grid generation.
- *Surface grid generation*: Once surface patches of interest for grid generation are identified, two distinct approaches are available for surface grid generation. They are "Paneling" and "Topology-based" grid generation. We briefly describe them here:

  1. *Paneling*: Grid generation by paneling proceeds by taking planar sections of the surface configuration. Usually parallel set of planes is used for obtaining intersection curves. These curves are trimmed to obtain composite curves describing the "outer profile" of the aircraft geometry. These section curves are

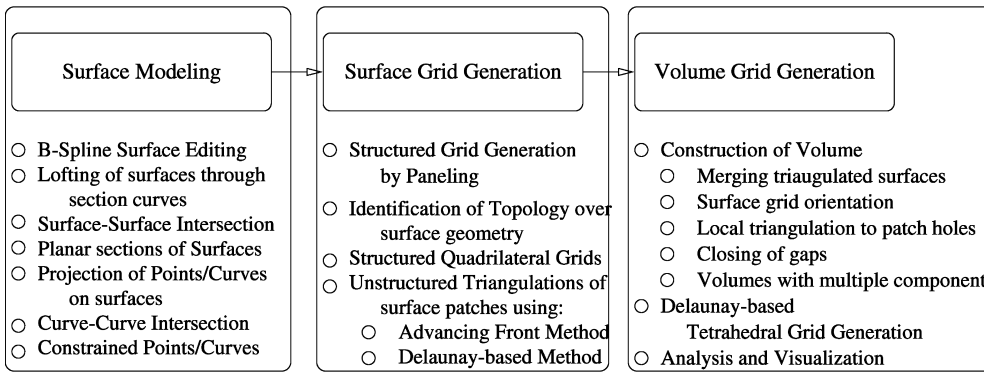

Fig. 1. Model of usage: surface modeling, surface grid generation and volume grid generation.

discretized by distributing equal number of points along each curve. A mesh is constructed by joining corresponding points across adjacent curves.

2. *Topology-based methods*: These methods obtain a topological identification of "faces" over the surfaces in the given geometry. These faces are bounded by "edges" which may be surface boundaries or intersection curves or isoparametric curves. These edges are discretized by distributing points over them using some clustering function. Given a face with each of its bounding edges discretized, Zeus has two types of grid generation methods:

- *Structured quadrilateral grid generation*: If the given face is four-sides and the number of points on the opposite edges is equal, then such a face forms a suitable candidate for structured quadrilateral grid. The interior points are distributed to obtain as much orthogonality at grid-line intersections and as much smoothness of arclength and area gradients as possible.
- *Unstructured surface triangulation*: This class of grid generation methods can handle faces bounded by more than four sides and arbitrary number of points distributed on the bounding edges of the faces. The face identified over the 3D surface geometry is triangulated with constraints such as aspect-ratios and density. Zeus has Delauanay-based as well as advancing front algorithms implemented for direct surface triangulation.
- *Volume grid generation*: Triangulated surface geometry is used to obtain a tetrahedral decomposition of space around it. For example, triangulated aircraft geometry may be placed in a huge triangulated sphere. Surface grids generated over individual patches are composited to form a closed mesh by patching holes and gaps, if needed. The space bounded by the outer sphere surface and the inner aircraft geometry forms the region of interest for tetrahedral decomposition for CFD analysis. Zeus implements Delaunay-based volume grid generation method. This grid must contain the vertices, edges and faces of the bounding surface grids.

3. Zeus entities and operations

The entities and algorithms implemented in Zeus can be broadly classified in the following categories: (a) geometry, (b) topology, (c) surface grids and (d) volume grids. In the following subsection, we give a detailed description of the entities in each category along with the operations supported on each of the entities.

3.1. Geometry

As mentioned in the previous section, 3D geometry component of Zeus consists of points curves, clustergraphs, surfaces, planes and algorithms associated with them. For each kind of geometry entity, standard operations like creation, editing, archival and drawing functionality is built into the classes that encapsulate these entities.

1. *Point*: A 3D point in Zeus is either (a) a control point (also called a free point) used to define a higher-level entity such as a curve, a surface or a plane; or (b) a constrained point that lies on a curve or a surface. A free point has three degrees of freedom in space, whereas, a constrained point's freedom is constrained by its underlying entity (see Fig. 2). A constrained point on a curve has only one degree of freedom and that on a surface has two degrees of freedom. However, these degrees of freedom may be further reduced by additional constraints.

Constrained points can have multiple representations across underlying entities. For example, a constrained point generated by the intersection of a curve and a surface has two representations: one on the curve and the other on the surface (each representation has a pointer to

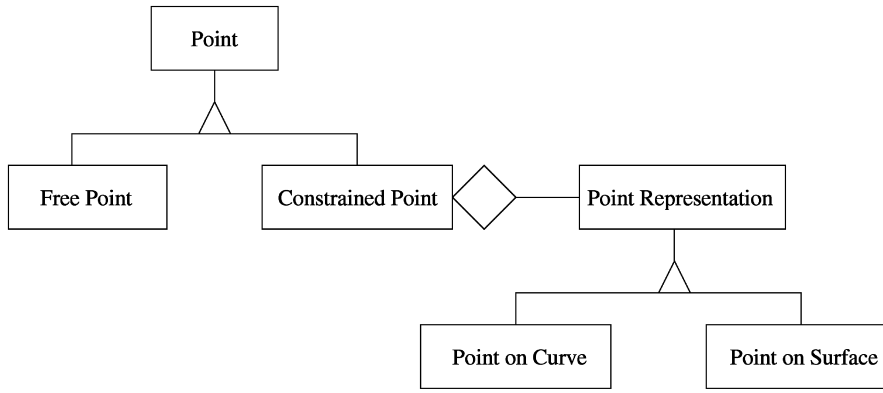

Fig. 2. Free point and types of constrained points.

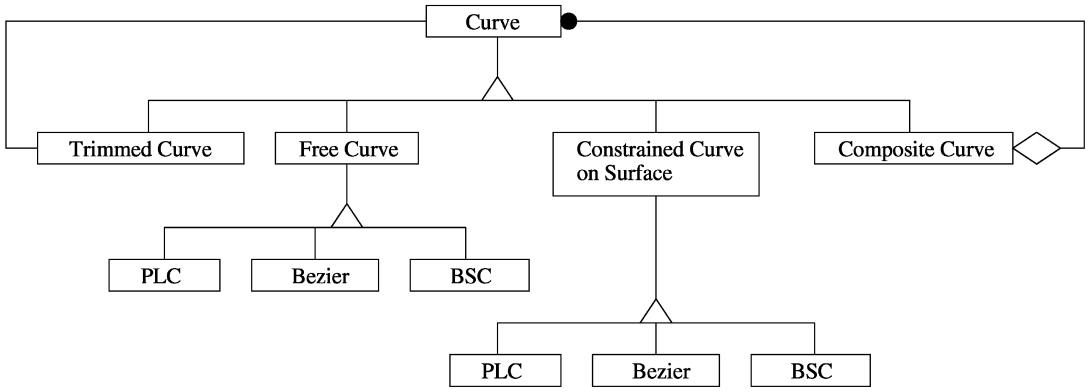

Fig. 3. Free curves, trimmed curves, composite curves and constrained curves.

the underlying entity and position of the point with respect to the entity, i.e. the parameter values).

Constrained points are generated by intersection of curves, surfaces, or by projection of points on these higher-level entities.

2. *Curves*: Zeus supports free curves with B-spline [7] representation, constrained curves (curves constrained to lie on surfaces) represented using composite 2D Béziers [8] (2D since they are defined in terms of (u, v) parameters on surface). In addition to these, trim curves and composite curves are supported (see Fig. 3).

B-spline curves implemented are not limited in terms of their order or number of control points. The sequence of control points is formed using free points described above. B-splines of order 2 are used to create and represent piecewise linear curves (PLCs). B-splines are constructed either by using control points or by interpolation through a given sequence of points. B-splines are represented using control points, knot vector and a flag that indicates whether it is open or closed. These curves are also called "free curves" since they are not constrained to lie on any underlying entity that constrains the degrees of freedom.

Constrained curves are obtained via the following different operations:

(a) surface–surface intersection (SSI),
(b) plane–surface intersection,
(c) projection of a curve on a surface, and
(d) extraction of iso-parametric curves.

The first three operations in the above list generate constrained curve representations in the form of composite Bézier curves. Whereas iso-parameter curves can be represented simply by a pointer to the underlying surface, direction of parameter and parameter value. Constrained curves can have multiple representations on different underlying entities.

Trim curves are required to represent parts of curves that are modeled or obtained by intersection operation. They are represented by a pointer to the underlying curve, and parametric bounds t_1 and t_2. Composite curves are composed by connecting multiple curves with some user-defined continuity requirements.

3. *Surfaces*: The types of surfaces supported in Zeus are: (a) B-spline surface, (b) triangulated surface, (c) trim surface and (d) composite surface (see Fig. 4).

B-spline surfaces are represented in terms of control-point mesh, knot vectors for u and v directions and two flags indicating whether the surface is open or closed in either direction. B-spline surfaces are created using a specification of a control mesh, orders of polynomials in either directions and the open/closed flags. They can also be constructed by lofting a surface through a sequence of B-spline curves.

Zeus also incorporates triangulated surface representation for handling surfaces that are captured and imported as triangulations.

Trim surface is constructed by identifying a sequence of trimming curves in a specific orientation. The region bounded by trimming curves is the trim surface. Composite surfaces are constructed by a set of surfaces connected at their boundaries (trimmed or otherwise) to satisfy some user-defined continuity criterion.

Numerous operations are implemented using surfaces:

(a) surface–surface intersection,
(b) projection of a point on surface,
(c) projection of a curve on surface, and
(d) planar sections of a surface [9].

4. *Plane*: A plane is used in Zeus for taking planar sections of surfaces. It is represented by a position vector and a normal direction. User specifies a plane either via mouse interactions or by specifying the coefficients of the plane equation. Various other creation operations are supported such as construction of a sequence of planes parallel to a given plane and a sequence of planes between two planes.

5. *Cluster Graph*: Cluster graph is basically a function of the form $y = f(s)$ where y denotes the cluster

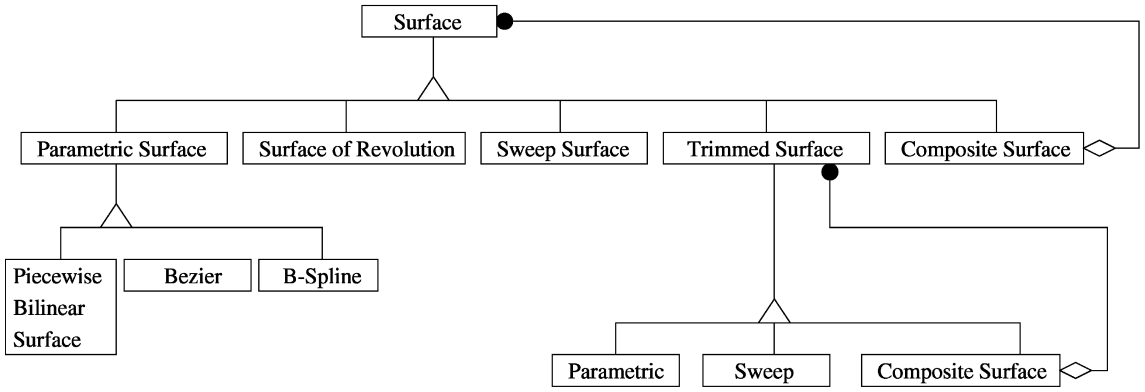

Fig. 4. Types of surfaces in Zeus.

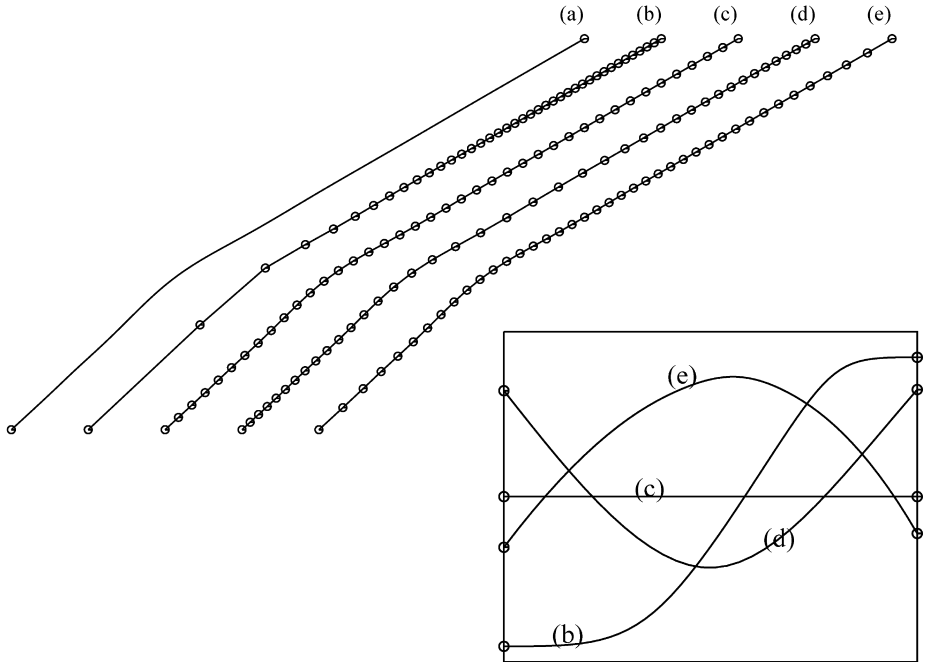

Fig. 5. Different point distributions along a boundary curve of the wing geometry and the corresponding cluster graphs. (a) is the original curve and the clustering are shown in (b)–(e).

factor and s the arc length parameter of an edge or a curve segment. The cluster graph is interpreted as follows:

If N is the total number of points to be distributed along an edge and the arc length parameter s of the edge varies from, say, 0 to L, then the number of points along the edge from s_1 to s_2 is given by

$$N^* \frac{\int_{s_1}^{s_2} f(s)\,\mathrm{d}s}{\int_0^L f(s)\,\mathrm{d}s}.$$

Fig. 5 shows different point distributions on a curve specified by different cluster graphs.

3.2. Topology

The entire surface of a complex body is defined as the union of surface patches, each surface patch being a trimmed portion of a B-spline surface or any parametric surface. The topological break up has to be such that there are no overlapping patches – that is, any two

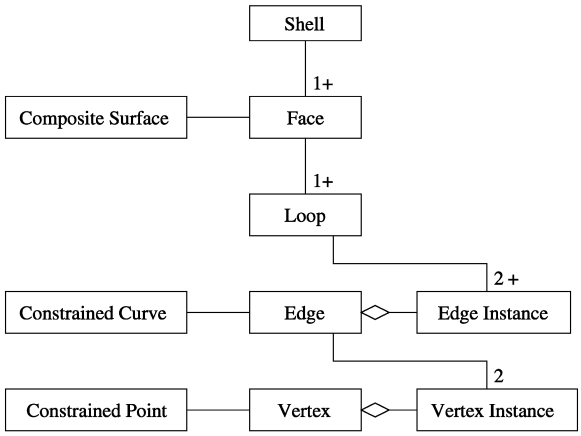

Fig. 6. Hierarchical structure of NMG data-structure.

distinct patches are either disjoint or share a vertex or share one or more edges.

In order to deal with complex surfaces like an aircraft surface, an efficient data structure is necessary to exactly define and validate a composite surface and for subsequent analysis and maintenance. For this, we have implemented the nmg data structure of [10] which is a generalization of the so-called radial edge data structure [11]. This is referred to as the topological model of the composite surface.

In the topological model (see Fig. 6) a composite surface is called a shell and a surface patch is called a face. A face is bounded by one or more closed loops of edges. A shell is at the top of the following hierarchy of topological entities [10]:

A shell is composed of faces; a face is defined by one or more loops (bounding curves) over an underlying surface; a loop is formed by a connected sequence of edges; an edge is defined by two vertices and an underlying curve; a vertex is defined by a point.

3.3. Surface tessellation by paneling

A grid generated by paneling is called a mesh. From now on we follow this terminology in the rest of this paper. The procedure of mesh generation by paneling is outlined below stepwise.

1. Take sections of a surface by intersecting it with a sequence of planes or with other surfaces (see Fig. 7).
2. Discretize the section curves or parts of section curves by computing sequences of points on them. On each curve (or part of it), the computed points can be uniformly distributed or clustered according to a cluster graph. A discretized section curve is called a *Polygonal Mesh Curve* (PMC). A PMC is stored as a separate entity in ZEUS. So, a PMC is a piecewise linear curve whose vertices are constrained to lie on a section curve.
3. Take an ordered sequence of PMCs, all having the same number of points. A *panel* is a four-sided region (quadrilateral) defined by its vertices. A panel formed by four vertices A, B, C, D is denoted by (A, B, C, D). From the given PMCs, form panels by joining adjacent points as follows (see Fig. 7):

   Let there be m PMCs having n points each. Let $\{P_{ij}\}, 1 \leqslant j \leqslant n$ be the n points on the ith PMC. Join P_{ij} and $P_{i+1,j}$ for $1 \leqslant i \leqslant m-1$ and $1 \leqslant j \leqslant n$. Then $(P_{ij}, P_{i+1,j}, P_{i+1,j+1}, P_{i,j+1})$ for $1 \leqslant i \leqslant m-1$ and $1 \leqslant j \leqslant n-1$ are the panels which are obtained from the given sequence of PMCs.

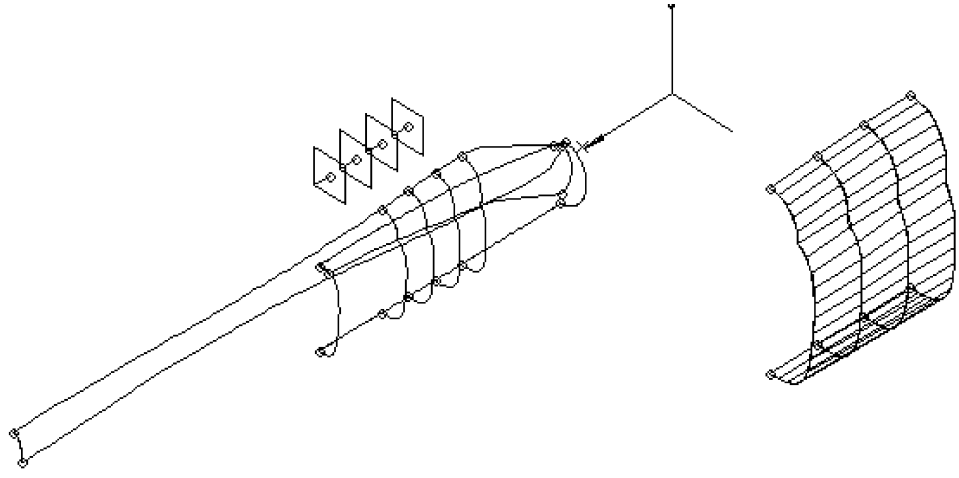

Fig. 7. Trim curves across surfaces for creating panels.

4. A mesh is a collection of panels. So, a mesh is nothing but a *quadrilateral grid*. In Zeus a mesh is stored as a separate entity. By the above steps 1–3, we obtain a mesh over the given surface by collecting all the panels created in step 3.
5. A mesh covering a complex geometry defined by a composite surface is obtained by merging different meshes defined over different parts of the geometry (see Fig. 8).

While orthogonality, uniformity and smoothness constraints are not taken into account in technique and many operations need to be done manually, this is a powerful tool especially in places where other algebraic methods fail to give acceptable structured quadrilateral grids.

3.4. Surface grid generation

Once the topological model of the composite surface is created, grid generation is done in three steps:

1. Compute grid points over the edges. Clustering of grid points over an edge can be specified by a cluster graph (see Fig. 9).
2. Over each face (surface patch), create structured or unstructured grid corresponding to the already computed grid points on the edges of that patch. This ensures the requirement that adjacent patches have the same grid points along their common edges.
3. The grid over the whole shell (composite surface) is obtained as the union of the grids over all the component faces.

3.4.1. Edge grid

Edge grid points can be uniformly or non-uniformly distributed over the edge. This entity is represented by a pointer to a topological edge, pointer to a cluster graph and number of points on the edge grid.

3.4.2. Face grid

A face grid is a collection of quadrilateral or triangular regions called elements such that:

1. The union of all the elements is the whole face
2. Intersection of any two elements is any of the following:
   - empty;
   - a node (corner point) of an element;
   - an element-edge common to both the elements.
3. All nodes of elements should lie on the underlying surface of the face.

3.4.3. Structured face grid

A face grid is said to be structured if the following conditions are satisfied:

1. The face is defined by a single loop having four edges.
2. Number of grid points on the two pairs of opposite edges are same, say m and n.
3. The grid is defined by an $m \times n$ array of points $\{P_{i,j}\}$, $1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n$ such that, the first row of points lie on the first edge, last row on third edge, first column on fourth edge and the last column on second edge.
4. The elements of the grid are the quadrilaterals with nodes $\{P_{i,j}, P_{i+1,j}, P_{i+1,j+1}, P_{i,j+1}\}$ where $1 \leqslant i \leqslant m - 1$, $1 \leqslant j \leqslant n - 1$, so that the total number of elements $= (m-1)*(n-1)$.

To be useful for CFD analysis, structured quadrilateral grids must satisfy qualitative requirements: (a) smoothness: the gradient of distances between successive grid points on the grid-lines should be smooth,

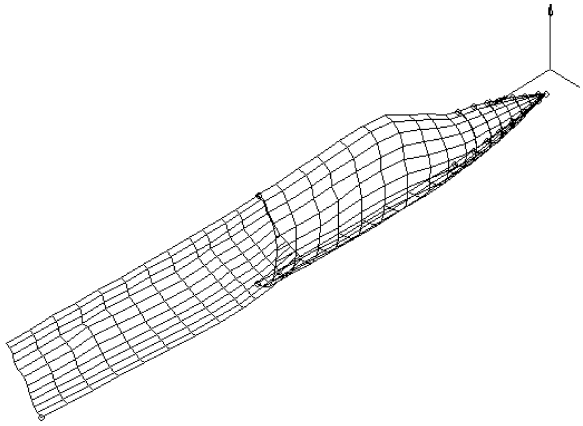

Fig. 8. Structured grid by paneling over a complex surface geometry.

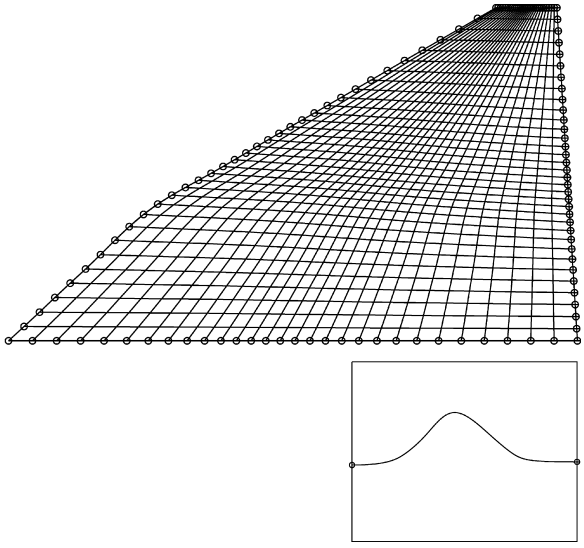

Fig. 9. Structured grid on wing surface and clustering function for distribution of boundary points.

(b) orthogonality: grid-lines meeting at vertices should be orthogonal as far as possible. Often it is hard to achieve these qualities across the entire surface grid generated over trimmed surface patches. This is because the trimmed patch may not be aligned with the isoparametric curves of the surface on which the patch is identified, and also the shape of the boundary curves on the trimmed patch as seen on the parametric domain may lead to a concave shape (Fig. 10).

3.4.4. Unstructured face grid

A face grid which is not structured is called an unstructured face grid. Unstructured face grids are defined over faces of non-rectangular topology, that is, faces having 3 edges or more than four edges and also over faces of rectangular topology when one does not want equal number of points on opposite edges.

Zeus has implementations of extended Delaunay triangulation method [12] and advancing front algorithm [13] for direct surface triangulation. These implementations are described in the next section.

3.4.5. Shell grid

Union of face grids on all faces that form a shell gives us a shell grid. A shell grid usually refers to a grid on complete surface geometry of a system like an aircraft or a car body.

3.5. Tetrahedral volume grid generation

Having obtained surface grid for the entire surface geometry, the next logical step toward CFD analysis is to obtain a volume grid. Zeus incorporates Delaunay-based tetrahedral volume grid generation tool.

The problem of grid generation can be stated as:

> Given a bounded volume defined in terms of oriented triangulated surfaces, the volume grid generation problem is to decompose the volume into tetrahedral elements such that the union elements spans the given 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 mesh is also required to contain the triangles of the meshes defining the boundary of the volume.

The following facilities are built into this part of the package:

1. *Composition of volume*: Surface grids generated over topological faces of geometric model can be merged along boundaries to compose a closed triangulated surface. If the triangulated model has holes or gaps, then they are patched by local triangulation or by collapsing the gaps. Such a completely closed shape may then be placed in a triangulated enclosure such as a huge sphere, a cylinder with closed ends or a box. Such a volume with multiple connected components must be consistently oriented to identify the bounded volume for tetrahedral discretization (see Fig. 11).
2. *Tetrahedral decomposition*: A Delaunay-based algorithm for tetrahedral grid generation of the volume observes the following constraints:
   - Initial triangulated boundary is always a part of the final tetrahedral grid. That is, all vertices, edges and triangles are a part of the set of tetrahedra generated.
   - Quality of the desired grid is specified in terms of aspect ratio of tetrahedral elements and smoothness of volume gradients along any line considered within the bounded region.
3. *Smoothing*: A modified Laplacian smoothing algorithm is implemented for smoothing tetrahedral grids. This smoother consists of a local operator that modifies the position of an interior vertex. This operator tries to move the vertex to the centroid of its neighborhood. If the dihedral angles of tetrahedra connected to this vertex are distributed more uniformly

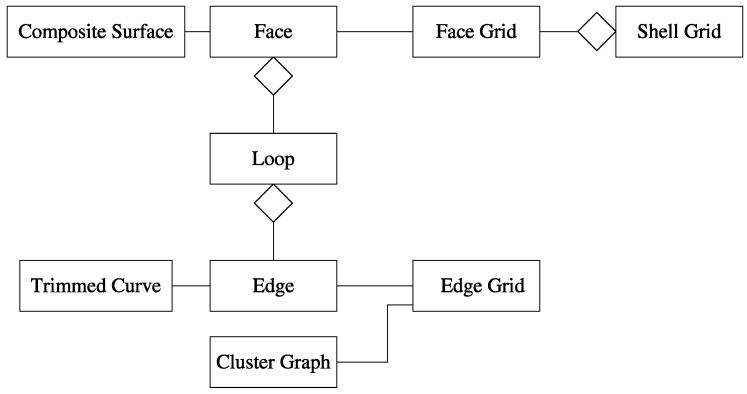

Fig. 10. Grids over topological identification of faces over surface geometry.

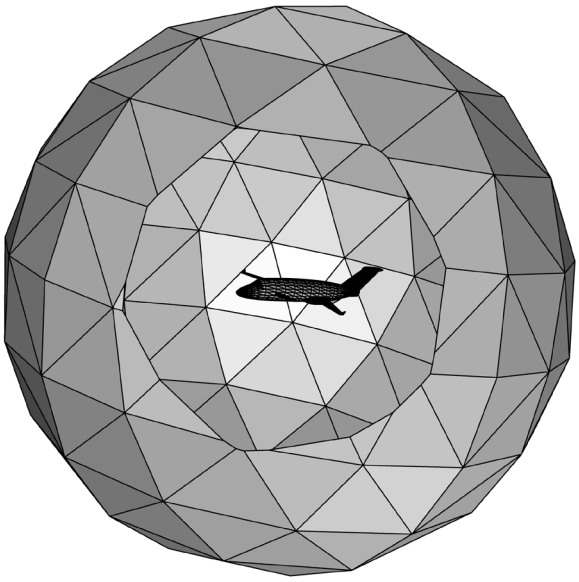

Fig. 11. Composition of volume.

than earlier and this new local configuration is valid, then it is retained; otherwise the change is undone for that vertex. This operator is iteratively applied to all interior vertices (Fig. 12).

3.5.1. Grid representation

To represent the tetrahedral volume grid, we use radial-edge data structure. This is different from that used to identify higher-level surface topology of geometry. The aim of this data structure is to facilitate fast updation and querying of variable and fixed cardinality relationships in the grid. Examples of fixed cardinality relationships in a tetrahedral grid are: an edge owns two vertices; a triangle owns three edges; a face is shared by at most two tetrahedra; etc. Examples of variable cardinality relationships are: a vertex may be shared by many edges, triangle and tetrahedra; an edge may be radially shared by multiple triangles; and so on. A hierarchical relationship between grid vertices, edges, triangles and tetrahedra is illustrated in Fig. 13.

4. Innovative algorithms and techniques

Many new algorithmic contributions have been in the course the development of this package. We outline them in this section.

4.1. Compact geometric bounds

We have given a new method to estimate the maximum deviation of a curve segment from the line joining its end points [14]. If $f:[0,1]\to R^3$ is an nth degree polynomial curve with $2\leqslant n\leqslant 9$, and if $F(u)$ is the perpendicular distance of $f(u)$ from the line joining $f(0)$ and $f(1)$, then

$$\max_{0\leqslant u\leqslant 1}|F(u)|\leqslant\left[\sum_{i=1}^{n-1}|F(u_i)|^2\right]^{1/2},$$

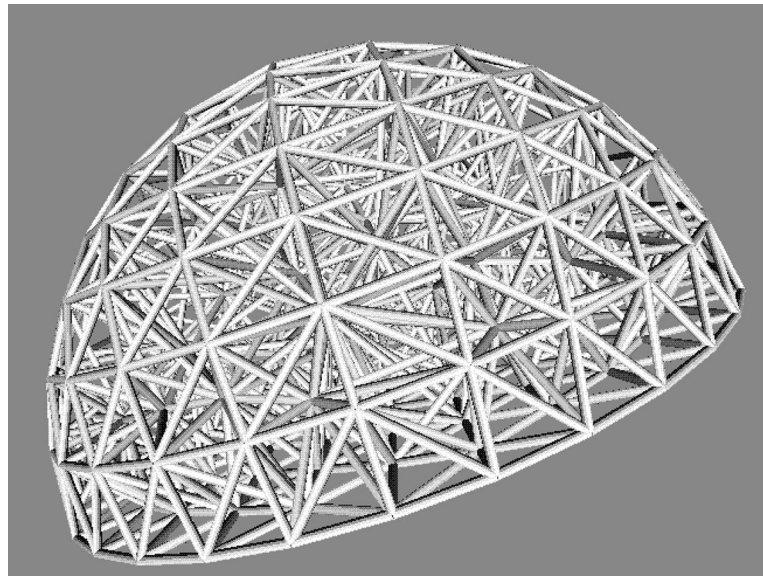

Fig. 12. Tetrahedral volume grid.

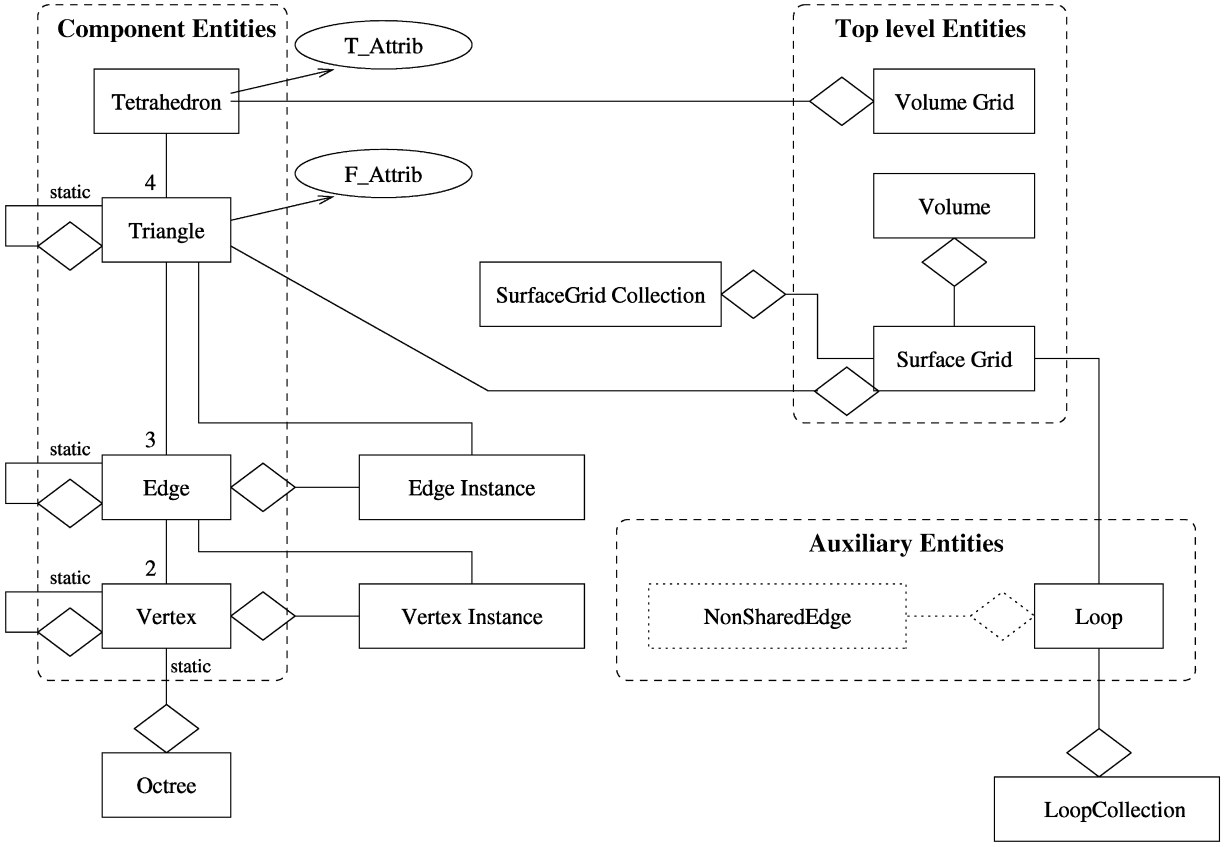

Fig. 13. Radial-edge data-structure for tetrahedral volume grids.

where u_i, $1 \leqslant i \leqslant n-1$, are pre-computed fixed parameter values (golden points) independent of the polynomial curve f. This determines the linearity of a curve segment and is used in subdivision algorithms for problems such as point inversion and curve–curve intersection. The above method has also been extended to rational polynomial curves and polynomial surface patches.

4.2. Optimal choice of parameter values for interpolation

Given a sequence of points $Q_i, 1 \leqslant i \leqslant n$, in R^3 and, optionally, tangent directions T_i and curvature values κ_i, the problem is to find a parametric curve $f:[a,b] \to R^3$ such that

$$f(u_i) = Q_i, f'(u_i) = \alpha_i T_i,\ f''(u_i) = \beta_i T_i + \alpha_i^2 \kappa_i N_i,$$

for some parameter values u_i in $[a,b]$ and α_i, β_i in R, $1 \leqslant i \leqslant n$; N_i is the principal normal vector at Q_i. α_i and β_i are known as shape parameters [15]. The choice of values for the parameters u_i, α_i, and β_i will affect the shape of the resulting interpolation curve [16, 17]. In order to obtain *fair* curves, we have added the following energy minimization constraint to the above interpolation problem:

$$\text{Minimize } E(f) = \int_a^b \| f^{(r)}(u) \|^2 \, \mathrm{d}u,$$

by varying u_i, α_i, and β_i in the interpolation problem. $f^{(r)}$ is the rth order derivative of f. For $r = 1, 2$ and 3, $E(f)$ can be considered as a measure of arc length, curvature and rate of change of curvature, respectively. It is found that with $r = 1$, one gets very taut curve while with $r = 2$ or 3, one gets smooth and rounded curves.

4.3. Planar sections

We represent a planar section of a B-spline surface by its pre-image, a *trim curve*, in the 2D parametric domain of the surface. The trim curve is computed as a 2D

composite Bézier curve with G^1 or G^2 continuity. This is done in two steps [9]:

1. Compute a sequence of exact intersection points in 3D lying on a net of iso-parametric curves and get the corresponding points in the 2D parametric domain.
2. Between any two adjacent points in the parametric domain, fit a Bézier curve which minimizes the integral of the square deviation of the corresponding curve on the surface from the intersecting plane.

4.4. Two-phase technique for structured grid generation

Structured quadrilateral grid generation over arbitrary trimmed patch is a difficult problem due to the quality and validity constraints. We have developed a two-phase algorithm [18] which has the following phases: (a) Reparameterization of trimmed four-sided surface patch for obtaining a convenient computational domain. This new parameterization is used for obtaining an initial grid as well as for subsequent phase. (b) Grid optimization phase: in this phase, an objective function is obtained in terms of quality metrics from the surface, which when minimized gives most desirable grids.

4.5. Advancing front method directly on the surface

The standard advancing front procedure [19, 20] can be extended to generate grids directly over curved surfaces [13]. We have implemented this method to generate unstructured grids over trimmed parametric surface patches. The procedure involves the following steps:

1. Form the initial front by taking the grid points of the boundary defining the surface patch. These are 3D points lying on the surface.
2. Select one of the edges of the front as the current edge. One can select the shortest edge as the current edge.
3. Compute a trial point lying on the surface such that it forms an equilateral triangle with the current edge. This is done as follows:
   - Compute the tangent plane of the surface at the mid point of the current edge and select the point on the tangent plane which forms an equilateral triangle with the current edge and lies on the proper side of the edge.
   - Project this point on to the surface. This projected point can be perturbed locally to obtain a trial point.
4. Check if the trial point forms a valid triangle with the current edge and also does not lie near any existing vertex or edge. If the trial point is suitable then it is chosen as the forming vertex; otherwise, one of the existing nearby vertices which forms the most valid triangle with the current edge is chosen as the forming vertex.
5. Form the new triangle consisting of the current edge and the forming vertex.
6. Update the front – the current edge and any other edge of the new triangle which is in the front are removed from the front and other edge(s) of the new triangle are added to the front.
7. Goto step 2 if the front is not empty.

The computationally difficult steps in the above procedure are getting the trial point, which involves projection, and the validity check. For validity check, it is useful to keep the 2D parameter values of the vertices, because one can do validity check in the parametric domain and quality check in the physical domain. Fig. 14 is an example of a grid generated by this method.

4.6. Extended Delaunay algorithm for surface grid generation directly over the physical domain

We have implemented an extended Delaunay procedure which is applied directly on the physical domain of the surface. The algorithm we have extended is the *edge swapping algorithm* for 2D Delaunay triangulation [21]. The outline of our procedure, the details of which are given in [22], is as follows:

1. Compute an initial triangulation in the parametric domain and map it onto the surface.
2. For each edge in the parametric domain, compute the arc-length of its image on the surface. Assign this arc-length as the length of the edge.
3. Perform edge swapping procedure [21] using these arc-lengths as the lengths of the edges.
4. Make a list of edges whose arc-lengths are more than a precomputed estimate of edge arc-length on the surface around that location.
5. Take the edge of maximum arc-length in the list and call it the current edge.
6. Compute a suitable point on the surface lying on the image arc of the current edge and compute its

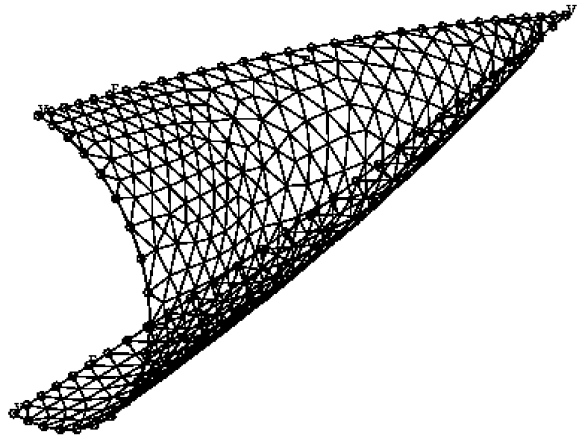

Fig. 14. Unstructured grid generated by advancing front method directly on the *s* surface.

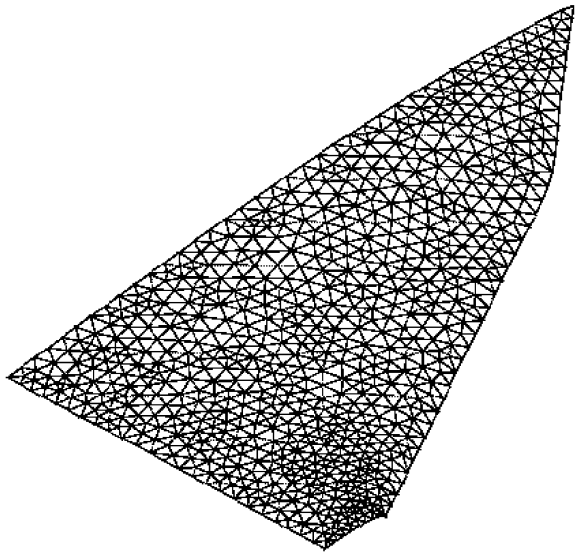

Fig. 15. Delaunay triangulation directly on the surface.

pre-image in the parametric domain. This is the new point to be inserted.

7. Insert this new point and replace the two triangles sharing the current edge by four triangles having the new point as one of their vertices.
8. Recompute the triangulation by edge swapping – again using the arc-lengths on the surface instead of edge lengths in the parametric domain.
9. Update the list of edges and goto Step 5 if the list is nonempty.

Note that here also we have followed the principle – valid triangulation structure in the parametric domain and quality evaluation in the physical domain. The grid in Fig. 15 is generated by this algorithm. Fig. 16 is an example of grid generated on a composite surface.

4.7. Generic grid smoothing on the surface

The well-known *Laplacian smoothing* procedure [23] of moving the interior grid points to the centroids of the surrounding polygons can be extended to surface grid. Since the grid points are constrained to lie on the surface, the interior points are to be moved along the surface only.

We can implement this using local projections. For this we assume that an interior grid point and the grid points connected to it are nearly coplanar. The procedure is as follows:

1. For each interior grid point compute and preserve the indices of the grid points connected to it.
2. For each interior point:
   (a) Get the connected grid points and compute the best plane for these connected points and the interior point.
   (b) Project all of these points on to this plane.
   (c) Move the projection of the interior point to the centroid of other projected points in the plane. Care has to be taken when the surrounding polygon of these projected points is not convex. In that case, the centroid of the kernel of the polygon can be taken.
   (d) Project back the new interior point onto the surface
3. Repeat step 2 till a stable state is reached. In practice, we have found that 5–10 iterations are sufficient to reach a near-stable state.

4.8. Acceleration of tetrahedral volume grid generation

Numerous acceleration techniques have been implemented in the tetrahedral volume grid generation algorithm. The Delaunay-based algorithm relies on fast spatial and connectivity queries for its speed. These include querying for a point or set of points lying in a specified spatial neighborhood, queries about topological

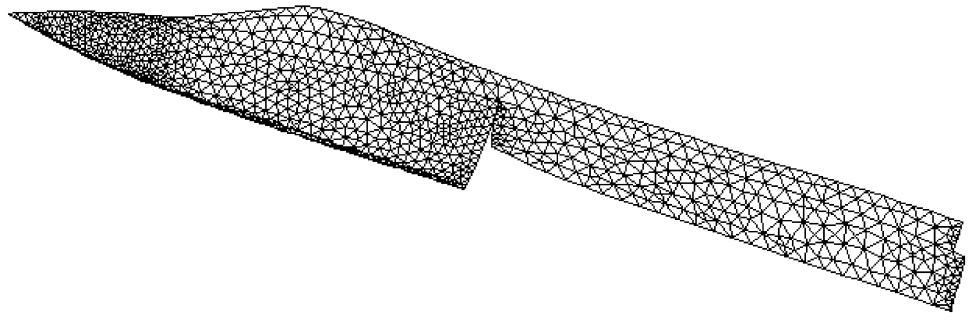

Fig. 16. Triangulation over a composite surface.

neighborhood and sharing information between vertices, edges, triangles, tetrahedra and meshes.

4.8.1. Octree

This spatial data-structure [24] is adopted to speed up queries about spatial neighborhood. Insertion of every new point in Delaunay-based algorithm requires determination of a list of tetrahedra in the neighborhood 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 up to 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.

4.8.2. Radial-edge data-structure

All entities in this data-structure 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 grid entities is as per Fig. 13. The data structure implemented has been adapted from the work reported by Muuss et al. [10] and Bruzzone [25]. 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 entities are surface-grid, volume and volume-grid are called top-level entities since the user is exposed only to these entities. The auxiliary entities are data structures used only by internal algorithms.

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.

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 capture the following relationships:

1. *Vertex-edge*: An edge is connected to exactly two vertices (constant cardinality relationship), however, a vertex may be connected variable number of edges, depending on particular configurations of facets in a surface-grid or tetrahedra in a volume-grid.
2. *Edge-facet*: A facet refers to exactly three edges, but an edge may be referenced in many facets radially connected to it (in case of volume-grid). In surface-grid, however, at most two facets can be connected to an edge.
3. *Facet-tetrahedron*: A tetrahedron refers to exactly four facets, and in a tetrahedral volume-grid in 3D, a facet may be associated with at most two tetrahedra. Note that this association has a constant cardinality both ways.
4. *Vertex-facet*, *vertex-tetrahedron*, *edge-tetrahedron*: These relationships are "derived" relationships. The relationships from tetrahedron to lower-level entities in the hierarchy can be trivially derived through traversals using pointers. The relationships in the opposite direction are captured using some auxiliary entities since the cardinality is not constant.

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.

5. Conclusion

Zeus is a system that was developed with a single user group in mind and has been used extensively by the user group. Close collaboration with the user group and their inputs have enabled us to hone the system and its functionality to suit to the user. Many of the techniques that are built into Zeus are complex and difficult to implement in a usable fashion. Grid generation is hard and particularly when complex geometric configurations like aircraft surfaces are involved. Work on Zeus will continue. There are plenty of improvements required by the user – better response times, grid adaptability, automatic CAD data repair, automatic topology determination, volumetric grid visualization techniques, etc.

References

[1] Shevare GR, Mudur SP. Constrained-interior interpolating surfaces. Proc of the Conference on Curves & Surfaces in Computer Vision and Graphics in SPIE/SPSE Symposium on Electronic Imaging Science and Technology, Santa Clara, CA, 1990.

[2] Mudur SP, Khandekar D. An interactive system for quick modeling of aircraft surfaces. Proc of the Conference on Curves & Surfaces in Computer Vision and Graphics in SPIE/SPSE Symposium on Electronic Imaging Science and Technology, Santa Clara, CA, 1990.
[3] Dassault Systems, France. CATIA user's manual (base system) November 1986.
[4] Computer Vision Corporation, USA, NURBS user's guide and reference, October 1988.
[5] Spatial Technology, Inc., ACIS 3D Toolkit, November 1994.
[6] Spatial Technology, Inc., ACIS Geometric Modeler, November 1994.
[7] Bartels R, Beatty J, Barsky B. An introduction to splines for use in computer graphics. Los Altos, CA, Morgan Kaufmann, 1987.
[8] Farin G. Curves and surfaces for computer aided geometric design. New York: Academic Press, 1988.
[9] Gopalsamy S, Reddy TS. Compace representation for planar sections of parametric surfaces. International Conference on Computer Graphics, February 1993.
[10] Muuss MJ, Butler LA, Combinatorial solid geometry, boundary representations and *n*-manifold geometry. In: Rogers, Earnshaw, editors. State of the art in computer graphics – visualization and modeling. New York: Springer, 1991.
[11] Weiler KJ. The radial edge structure: a topological representation for non-manifold geometric modeling. In: Wozney M, McLaughlin H, Encarnacao J, editors, Geometric modeling for CAD applications. Amsterdam: North-Holland, 1987: 37–66.
[12] Holmes DG, Snyder DD. The generation of unstructured triangulation meshes using delaunay triangulation. In: Sengupta S, et al., editors, Numerical grid generation in computational fluid mechanics. Swansea: Pineridge, 1988.
[13] Nakahashi K, Sharov D, Direct surface triangulation using the advancing front method, AIAA-95-1686-CP, 1995.
[14] Gopalsamy S, Khandekar D, Mudur SP. A new method of evaluating compact geometric bounds for use in subdivision algorithms. Computers Aided Geometric Design. 1991; 8:337–56.
[15] Barksy, BA. Computer graphics and geometric modeling using beta-splines. Berlin: Springer, 1988.
[16] Khandekar DR, Gopalsamy S, Mudur SP. Mathematical techniques for the interactive design of b-spline curves and surfaces using from constraints. Eurographics Workshop on Computer Graphics and Mathematics, October 1991.
[17] Gopalsamy S, Reddy TS. Energy and error function minimisation for computation of optimal shape parameters. Computers and Graphics 1993; 17(4):403–5.
[18] Dinesh S, Gopalsamy S, Mudur SP. Two phase technique for tessallation of complex geometric models, 8th International Conference on Engineering Computer Graphics and Descriptive Geometry, July, 1998.
[19] George PL. Automatic mesh generation – application to finite element methods. New York, Paris: Wiley, Masson 1991.
[20] Lohner R, Parikh P. Generation three-dimensional unstructured grids by the advancing-front method, AIAA-88-0515, 1988.
[21] Lawson CL. Software for C^1 surface interpolation. In: Rice JR, editor. Mathematical software III. New York: Academic Press, 1977; 161–94.
[22] Gopalsamy S, Reddy TS, Dinesh S, Mudur SP. Extended delaunay triangulation directly on surface patches. Technical Report. Graphics and CAD Division, National Centre for Software Technology, July 1997.
[23] Freitag L, Ollivier-Gooch C. A Comparison of tetrahedral mesh improvement techniques. In: Proceedings of the 5th International Meshing Roundtable, October 1996.
[24] Hanan S. Applications of spatial data structures: computer graphics, image processing and GIS, Reading, MA: Addison-Wesley, 1990.
[25] Bruzzone E, De Floriani L. Two data structures for building tetrahedralizations, Visual Computer, 1990;6(5):266–83.