CHAPTER 2

LITERATURE SURVEY

A key step of the finite element method for numerical computation is mesh generation. A mesh is a discretization of a given domain (such as a polygon or polyhedron; more realistic versions of the problem allow curved domain boundaries) into small simple “elements” (such as triangles or quadrilaterals in two dimensions and tetrahedra or hexahedra in three) meeting in well-defined ways (Bern and Plassmann 1999).

2.1 DESIRABLE PROPERTIES OF MESHES

The components of a mesh i.e. vertices, edges and faces, should satisfy the following conditions (Philip and David 2002):

- Each vertex must be shared by at least one edge. (No isolated vertices are allowed.)
- Each edge must be shared by at least one face. (No isolated edges or polylines are allowed.)
• If two faces intersect, the vertex or edge of intersection must be a component in the mesh. (No interpenetration of faces is allowed. An edge of one face may not lie in the interior of another face.)

The first goal of mesh generation is to correctly model the shape of a problem domain. A mesh must conform to the object or domain being modeled and ideally should meet constraints on both the size and shape of its elements.

Scientists and engineers often wish to model objects or domains with complex shapes, and possibly with curved surfaces. Boundaries may appear in the interior of a region as well as on its exterior surfaces. Exterior boundaries separate meshed and unmeshed portions of space and are found on the outer surface and in internal holes of a mesh. Interior boundaries appear within meshed portions of space, and enforce the constraint that elements may not pierce them. These boundaries are typically used to separate regions that have different physical properties; for example, at the contact plane between two materials of different conductivities in a heat propagation problem. An interior boundary is represented by a collection of edges (in two dimensions) or faces (in three dimensions) of the mesh. A thorough survey of the pertinent techniques is offered by Bern and Eppstein (1992).

A second goal is to offer as much control as possible over the sizes of elements in the mesh. Ideally, this control includes the ability to grade from small to large elements over a relatively short distance. The reason for this requirement being: small, densely packed elements offer more accuracy than larger, sparsely packed elements; but the computation time required to solve a
problem is proportional to the number of elements. Hence, choosing an element size entails trading off speed and accuracy.

If elements of uniform size are used throughout the mesh, one must choose a size small enough to guarantee sufficient accuracy in the most demanding portion of the problem domain, and thereby possibly incur excessively large computational demands. Given a coarse mesh with relatively few elements, it is not difficult to refine it to produce another mesh having a larger number of smaller elements. The reverse process is not so easy. Hence, mesh generation algorithms often set themselves the goal of being able, in principle, to generate a mesh with as few elements as possible. They typically offer the option to refine portions of the mesh whose elements are not small enough to yield the required accuracy.

A third goal of mesh generation is that the elements should be relatively “round” in shape, because elements with large or small angles can degrade the quality of the numerical solution. Elements with large angles can cause a large discretization error, as discussed by Jonathan Richard Shewchuk (1996). However, Babuška and Aziz (1976) show that if mesh angles approach 180°, as the element size decreases, convergence to the exact solution may fail to occur.

Small angles are also feared, because they can cause the coupled systems of algebraic equations that numerical methods yield to be ill-conditioned (Graham 1984). If a system of equations is ill-conditioned, roundoff error degrades the accuracy of the solution if the system is solved by direct methods and convergence is slow if the system is solved by iterative methods. Hence,
many mesh generation algorithms take the approach of attempting to bound the smallest angle.

There should be few elements, but some portions of the domain may need small elements so that the computation is more accurate there. The mesh can tremendously influence the accuracy and efficiency of a simulation. Although different applications have different requirements, it is generally true that a good mesh will have small elements for detail, large elements for efficiency, and "nicely shaped" (which means different things in different situations, but generally involves bounds on the angles or aspect ratio of the elements) elements for accuracy. Detailed surveys of the mesh generation literature have been supplied by Thompson and Weatherill (1993) and Bern et al (1991).

The notion of a nicely shaped element varies depending on the numerical method, the type of problem being solved and the polynomial degree of the piecewise functions used to interpolate the solution over the mesh. For physical phenomena that have anisotropic behavior, the ideal element may be long and thin, despite the claim that small angles are usually bad. The constraints of element size and element shape are difficult to reconcile because elements must meet squarely along the full extent of their shared edges or faces. Although nonconforming elements make it easier to create a mesh with seemingly nicely shaped elements, the problems of numerical error may still persist. Hence, the designer of algorithms for mesh generation is shooting at an ill-defined target.
2.2 TYPES OF MESHES

There are two major types of meshes: structured and unstructured. As discussed by Steven Owen (1998), they are distinguished by the way the elements meet; a structured mesh is one in which the elements have the topology of a regular grid, although it is deformed enough that one might fail to notice its structure. Strictly speaking, a structured mesh can be recognized by all interior nodes of a mesh having an equal number of adjacent elements.

A structured mesh is usually a warped grid of boxes, while an unstructured mesh is typically a triangulation with arbitrarily varying local neighbourhoods. A triangulation is a partition of a geometric input, typically the region defined by a point set or by simplices that meet only at shared faces. So, a triangulation consists of triangles that intersect only at shared edges and vertices. An optimal triangulation is one that is best according to some criterion that measures the size, shape or number of simplices.

FIGURE 2.1 Structured (left) and unstructured (right) meshes (Bern and Plassmann 1999)
Structured meshes exhibit a uniform topological structure that unstructured meshes lack. A functional definition is that in a structured mesh, the indices of the neighbours of any node can be calculated using simple addition, whereas an unstructured mesh necessitates the storage of a list of each node’s neighbours.

Structured meshes are characterized by regular connectivity, i.e. the points of the grid can be indexed and the neighbours of each point can be calculated rather than looked up (e.g. the neighbours of the point (i, j) are at (i+1, j), (i-1, j), etc.). Meshes on a rectangular domain are trivial to generate (though some care needs to be taken in the discretisation at convex corners). Structured mesh generation techniques concentrate on meshing domains with irregular boundaries, e.g., the flow in blood vessels or the deformation, flow and heat transfer in metal being formed in dies. The more structured the mesh, the smaller the amount of information that must be provided to the package for the mesh to be constructed.

Structured meshes must record the conditions within each cell, regardless of whether they are of interest. Because conditions are assumed to be uniform throughout each cell, a greater number of cells ensures greater accuracy in the simulation. Structured meshes also assume that, since the conditions within each cell are uniform, a cell containing two materials is recorded as having the average of the properties of each material.
The generation of both structured and unstructured meshes can be surprisingly difficult, each posing challenges of their own. The adaptability of unstructured meshes comes with new challenges, especially for 3D problems; the numerical theory becomes more difficult and the algorithmic design becomes harder.

### 2.3 WHY UNSTRUCTURED MESHES?

Some advantages that structured meshes generally hold over most applications are simplicity, convenience, availability of code, and suitability for multigrid and finite difference methods. They are typically easier to compute (saving a constant factor in runtime). The location of each cell is implicit by its coordinates, and so only one point must be logged to record the conditions of each cell. They require less computer memory, as their coordinates can be calculated, rather than explicitly stored. Finally, structured meshes offer more direct control over the sizes and shapes of elements.
Nonetheless, there are cases in which unstructured meshes are preferable or even indispensable. Many problems are defined on irregularly shaped domains, and resist structured discretization. The reason for using unstructured meshes is that the regularity of structured meshes is too limited.

Unstructured meshes based on simplices (i.e. triangles in 2D, tetrahedral in 3D) are very interesting for natural object modeling because they provide the best flexibility to fill regions defined by simplicial boundary elements (i.e. edges in 2D, triangles in 3D). There are cases in which unstructured meshes are preferable or even indispensable. Many problems are defined on irregularly shaped domains and resist structured discretization. There are several more subtle advantages of unstructured meshes.

The big disadvantage of a structured mesh is its lack of flexibility in fitting a domain with a complicated shape. They may require more elements or may result in improperly shaped elements. A number of techniques have been developed to find appropriate coordinate transformations: conformal mapping, algebraic methods and numerical methods that themselves solve differential equations. Even armed with these techniques, it may be impossible to find a transformation that fits a complicated domain acceptably well. Faced with this problem, some practitioners cut out a region of the grid, without any transformation, to give a ‘stair-case approximation’ to the domain. But then, the computed solution will be quite inaccurate near the boundary of the domain, an area that is often of special interest. Other practitioners break up the domain into simpler regions, perhaps overlapping, each of which can be more nearly matched by a deformed grid. This method and its associated numerical analysis make up ‘domain decomposition’, a large area of study in its own right. Because of the
need to fit complicated domains, such as aircraft and machine parts, the trend in simulation has been towards unstructured meshes.

Unstructured meshes are particularly beneficial where a range of feature sizes need to be accurately modelled. For example, in many EMC problems, large homogenous spaces exist between fine-featured objects which need to be correctly described for useful simulations. Unstructured meshes easily lend themselves to non-uniform meshing, employing the necessary computational resources around small features whilst minimising computational effort in the empty spaces.

Many problems are defined on irregularly shaped domains. For problems with complex geometry boundaries and with solutions that change rapidly, we need to use an unstructured mesh that has a varying local topology and spacing in order to reduce the problem size. For an example of modeling earthquakes, we need a much denser and finer discretization near the quake center, while it is desirable not to waste mesh points in regions with low activities. Anish and Walter (1997), Shang et al (2000), Chen et al (1997) have covered these in detail.

An unstructured mesh can be flexibly tailored to the physics of a problem (such as to predict the surface ground motion due to a strong earthquake), whereas the structured mesh must employ a uniform horizontal distribution of nodes, the density being dictated by the uppermost layer. As a result, it has five times as many nodes as the unstructured mesh, and the solution time and memory requirements of the simulation are correspondingly larger. The
disparity is even more pronounced in three-dimensional domains and in simulations where the scales of the physical phenomena vary more.

Unstructured meshes can be used to facilitate the connection of several different Cartesian mesh gradings; i.e. as a multigrid ‘glue’. Regularly spaced grids may be inefficient: the regular spacing between cells cannot be varied, and in areas of low or no relief, many redundant and repetitive points must still be entered, and in areas of high variation, important features may be missed if they do not fall directly on the regular grid spacing.

The big disadvantage of a structured mesh is its lack of flexibility in fitting a domain with a complicated shape. They may require more elements or may result in improperly shaped elements.

Unstructured meshes, far better than structured meshes, can provide multiscale resolution and conformity to complex geometries. Being irregular, they necessitate more computing. Despite this, unstructured meshes are better able to perform interface tracking needed to model fluid flows or cases having a great deal of unoccupied space. Because of the elasticity intrinsic in unstructured meshes, they are also excellent for concentrating cells in specific regions of interest, generating greater resolution and accuracy over those regions. It is also easy to automate the generation of unstructured meshes for random data. They can accommodate varying sample densities. Another key advantage is their superiority in mapping boundaries or material interfaces that are curved.
2.4 APPLICATIONS OF MESHES

Meshes composed of triangles or tetrahedra are used in applications such as numerical solutions of differential equations arising in physical simulation, computer graphics, interpolation, surveying, geographic information systems and terrain databases. The most challenging application of meshes is numerical methods for the solution of partial differential equations. Numerical methods such as the finite element method and the finite volume method are an irreplaceable means of simulating a wide variety of physical phenomena in scientific computing. In modeling the problems in these applications, the domains are partitioned into meshes. The mesh can tremendously influence the accuracy and efficiency of a simulation. They place particularly difficult demands on mesh generation.

Automatic unstructured mesh generation continues to be a vital technology in computational field simulations. As computing technology continues to advance and modeling requirements become more precise, automatic mesh generation techniques must rise to fulfill ever-increasing and diverse expectations.

2.5 TRIANGULATION

Triangulations are widely used as a basis for representing geometries and other information appearing in a huge variety of applications. From the history of surveying and the art of constructing maps - cartography, triangulation has been used as the basic technique for calculating the positions of landmarks in the terrain and for calculating distances between points on the Earth’s surface and a position’s elevation above sea level.
The meaning of triangulation is to generate a mesh of triangles from a given set of points. The triangles are formed by edges between the points of the set. Triangular meshes serve as a fundamental object representation not only in computer graphics but also in many other applications. The field of triangulation is part of the huge area of computational geometry.

![Example of triangulation](image)

**FIGURE 2.3 Example of triangulation**

Triangulation has been of particular interest because of their ultimate simplicity. – a triangulation is a collection of triangles, and a triangle is given by three points in space. Three points are the minimum number of points needed to represent a piece of surface in space.

Triangulation is the division of a surface or plane polygon into a set of triangles, usually with the restriction that each triangle side is entirely shared by
two adjacent triangles. It involves creating from the sample points a set of non-overlapping triangularly bounded facets; the vertices of the triangles are the input sample points. The triangle is the most popular primitive for drawing – almost all graphics libraries and hardware support triangles. It has been proved that every surface has a triangulation (Francis and Weeks 1999), but it might require an infinite number of triangles.

![Triangulation Illustration](image)

**FIGURE 2.4 Triangulation Illustration**

The formal definition: A triangulation of a set of points is a connected set of triangles in 2D or in 3D such that

1) The vertices of the triangles consist of the given points.
2) The interiors of any two triangles do not intersect.
3) If two triangles are not disjoint, then they share a vertex or have a coinciding edge.
4) All triangles are oriented consistently such that their normals point "outward".

These rules sound abstract, but some examples will shed light on them. Figures 2.3 and 2.4 are triangulations that satisfy the above rules. On the other hand, Figure 2.5 sketches three illegal triangulations, violating the above rules.
If we are given a point set, there is no unique triangulation, as can be seen from Figure 2.6. Among the many possible triangulations, there is one that is most commonly agreed to be the "best": the Delaunay triangulation.

Triangulation is the most widely used form of unstructured mesh generation, as any given arbitrary complex geometry can be more flexibly filled by triangular elements than by any other elements (Panigrahi and Sharma 2003). When polygons are specified with more than three vertices, it is possible that the vertices may not all lie in one plane.
2.6 TRIANGULATION ALGORITHMS

The other issue is how to triangulate the vertex set. Almost all programs tend to choose sample points that are most important. However, they vary in the measurement of sample points. Several triangulation schemes are available.

A number of triangulation algorithms are advocated. Different triangulation algorithms have different preferences when choosing the shapes of the triangles (i.e. how to triangulate the vertex set). Various criteria for a good triangulation have been defined. Most take four points forming the vertices of a quadrilateral and then decide which of the two possible internal diagonals is preferable. Some of the desirable criteria are:

1. Minimize the angle between the two normals of adjacent triangles in order to enforce a smooth transition (Xuanying 2003),
2. Maximize the minimum angle of a triangle (Lawson 1977),
3. Minimize the total length of the triangles (Akima 1975),
4. Minimize the maximum (or the sum) of longest edge / perimeter (Xuanying 2003),
5. Minimize the maximum (or the sum) of the aspect ratios of triangles (Gold et al 1977).

All triangulation methods try to avoid sliver triangles while favoring fat triangles (McCullogh 1983). There are several reasons behind this. One reason is that the points in a fat triangle have shorter distance to the vertices of the triangles. Sliver triangles may also cause other problems. David and Michael (1997) point out that in addition to the potential inaccuracy caused by
interpolation over a long distance, sliver triangles may also result in distorted visual effects due to aliasing and ill-conditioned linear systems which is not easy to solve.

As discussed by Raper and Kelk (1991), there are several reasons to try to build well shaped elements, triangulations whose elements have very low aspect ratios are undesirable because:

1. in graphics applications they result in shading irregularities
2. in finite elements analysis they can lead to ill-conditioned numerical problems
3. they weaken the value of the centroid as a representative location.

There are many algorithms to generate unstructured meshes. Most techniques currently in use can fit into three main categories: Octree, Delaunay and Advancing Front. By far the most popular of the unstructured meshing techniques are implementations of Delaunay triangulation.

2.6.1 Triangulation by Ear Clipping

Because triangulation of a polygon involves the diagonals of the polygon, a divide-and-conquer algorithm may be used to construct the triangulation. The idea is to find an ear, add the corresponding triangle to the list, remove the ear from the polygon, and recurse on the reduced polygon. It is assumed that the polygon has at least three vertices. This has been detailed by David (2002).
FIGURE 2.7 Illustration of Ear Clipping Algorithm

(a) Find a ear
(b) Find an ear and clip it.

(c) Keep clipping ears
(d) Keep clipping ears

(e) Keep clipping ears
(f) Keep clipping ears
2.6.2 Triangulation by Horizontal Decomposition

The idea is to construct a set of horizontal strips by sorting the $y$-values of the polygon vertices. The $y$-values can be sorted directly with an array-based sort, but by randomizing the edges and processing one at a time using a dynamic structure such as a binary search tree, the expected order is asymptotically better. Within each strip, the subpolygons are decomposed into trapezoids, each strip managing a dynamic structure such as a binary search tree to allow incremental sorting of the trapezoids. A brief description of the algorithm is presented in Paeth (1995).
The algorithm proceeds in the following three steps as illustrated in Figure 2.8.

1. Decompose the Polygon into Trapezoids.
2. Decompose the Trapezoids into Monotone Polygons.
3. Triangulate the Monotone Polygons.

![Figure 2.8 Illustration of triangulation by horizontal decomposition](image)

### 2.6.3 Delaunay Triangulation

The most popular triangulation is the Delaunay triangulation (DT), introduced by Boris Nikolaevich Delaunay in 1934 (while studying the geometric structure of crystals), along with its dual, the Voronoi Diagram, first discussed by Peter Lejeune-Dirichlet in 1850; but it was more than a half of a century later in 1908 that these diagrams were written about in a paper by Voronoi, hence the name Voronoi Diagrams. The Voronoi cells/polygons are sometimes also called Thiessen Polytopes or Dirichlet Regions or Dirichlet tessellation. Delaunay triangulation is a fundamental geometric construction, which has numerous applications in different computational problems.
The Delaunay triangulation is a geometric structure that has enjoyed great popularity in mesh generation since its infancy and it has become accepted that this is the best criterion to use for triangle definition. Shamos (1975) mentioned that only the Delaunay triangulation (equivalent to Lawson's maximized minimum angle Sibson 1977) has been shown to have a unique solution without testing every possible set of triangles.

Delaunay triangulation along with its dual, the Voronoi Diagram, is an important problem in many domains, including imaging, computer vision, terrain modeling and meshing for solving partial differential equations. In many of these domains, the triangulation is a bottleneck in the computation time, making it
important to develop fast algorithms. Its property of producing triangulations with well-shaped triangles makes it particularly appropriate for subsequent analysis. There is an excellent introduction to the Delaunay triangulation and its properties in Aurenhammer (1991) and Okabe et al (1992).

A Voronoi diagram is a geometric structure that represents proximity information about a set of points or objects. Given a set of sites or objects, the plane is partitioned by assigning to each point its nearest site. The points whose nearest site are not unique, form the Voronoi diagram. That is, the points on the Voronoi diagram are equidistant to two or more sites.

Voronoi diagram and Delaunay triangulation are an appealing solution because of their duality (they represent the same thing, just from a different point of view) and because both structures have interesting properties. The Delaunay triangulation of a vertex set simultaneously optimizes several of the quality measures: maxmin angle (i.e. maximizes the minimum angle), minmax circumcircle (i.e. minimizes the maximum circumcircle) and minmax min-containment circle over all possible triangulations of the same point set. It favors fat or equiangular triangles and tends to choose a triangulation that minimizes total edge length. The theory behind this is that the surface is more accurately represented at the vertices than at interpolated points far from a vertex. Therefore, Delaunay triangulation tries to make all points close to the vertices of the triangle where they are located. Its property of producing triangulations with well-shaped triangles makes it particularly appropriate for subsequent analysis.
2.6.3.1 Delaunay Criterion

A triangulation of a point set is called Delaunay if it satisfies the "empty circum circle" property: for each edge we can find a circle containing the edge's endpoints but not containing any other points i.e. the circumcircle of a triangle in the triangulation does not contain any input points in its interior.

The Delaunay triangulation of a given set of points is defined by the following property:

**AB is an edge of the Delaunay triangulation iff there is a circle passing through A and B so that all other points in the point set, C, where C is not equal to A or B, lie outside the circle.**

Equivalently, all triangles in the Delaunay triangulation for a set of points will have empty circumscribed circles. That is, no points lie in the interior of any triangle's circumsphere.
For any triangles $ABC$ and $ABD$, if $D$ is not inside the circumcircle of $ABC$, then the triangles $ABC$ and $ABD$ satisfy the Delaunay property. The circumcircle of a Delaunay triangle is called a Delaunay circle.

In Figure 2.12 there are three possible triangulations of this set of points. Only the first is a Delaunay triangulation, since all its circumcircles are empty.
2.6.3.2 Properties of Delaunay Triangulation

There are several useful desirable properties of the Delaunay triangulation, which make it distinct from other triangulation methods (Steven 1995):

- A convex equilateral formed by two adjacent triangles has a greater minimum internal angle than if the equilateral was formed another way i.e. the triangles are as equilateral as possible; thin wedge shaped triangles are avoided .

- The triangulation is unique (independent of the order in which the sample points are ordered) for all but trivial cases.

- Incircle: Given a triangle \( T(P_i, P_j, P_k) \) belonging to a Delaunay triangulation of a set of points \( P \), no other point of \( P \) is internal to the circle defined by \( P_i, P_j, P_k \).
• Situations with a mixture of regions of high and low density sampling are well handled, by giving a large number of triangles and hence more detail to the highly sampled regions and large triangles with less detail to the regions with a few samples.

• Discontinuities are handled quite naturally. The surface can have a discontinuity as narrow as the sampling process permits, it simply results in near vertical triangular facets.

• It should be possible to allow for variable spatial resolution; a fixed grid makes it difficult or impossible to model a significant horizontal component, such as stream meandering or fault displacement.

• MAX.-MIN. angle: A Delaunay triangulation minimizes the maximum interior angles, providing the most ‘equiangular’ triangulation of a given set of points. Given four points and the associated quadrilateral, the diagonal which splits it into two triangles is optimal in the way that maximises the lesser of the internal angles. This property guarantees that the shape of the triangles is the best possible for that set of points.

• It minimizes the maximum circumcircle over all possible triangulations of the same point set.

![Figure 2.13 Example of Delaunay Triangulation](image)

(a) A point set in the plane  
(b) Resulting Delaunay Triangulation

FIGURE 2.13 Example of Delaunay Triangulation
2.6.3.3 Degeneracies

Generally, the Delaunay triangulation is unique (independent of the order in which the sample points are ordered), but there are some cases when the triangulation is not unique as there exist different ways of connecting points and all lead to a valid triangulation. This degeneracy is quite common for regular distribution of points, for example in two dimensions when four points lie on a circle or if four points lie on the corners of a rectangle and the Voronoi vertices are coincident. But, these situations occur rarely in real data.

Common degeneracies in two or more dimensions are:

1. two points are coincident
   The Delaunay triangulation for a set of points is defined only if the points are distinct, so the uniqueness of the points in the given set is treated as a prerequisite for the applicability of this algorithm.

2. three points of a potential triangle are co-linear (or four points co-planar)
   This means that it is not possible to compute a valid centre for the circumcircle of a triangle or the centre of the inscribed sphere for a tetrahedra.

3. four or more points are cyclic.

FIGURE 2.14 Degeneracy: To the left, four dotted Voronoi edges meet at a common vertex and the dual Delaunay edges bound a quadrilateral. To the right, we have the general case, where only three Voronoi edges meet at a common vertex and the Delaunay edges bound a triangle.
It is evident from the definition of a Delaunay triangulation that problems arise in the procedure when certain degeneracies occur in the data (Lattuada and Raper 1996). Common ways to deal with such degeneracy are:

- reject the point,
- delay the point insertion,
- shift its co-ordinates i.e. arbitrarily perturb one or more of the vertices on the offending rectangle slightly.

The choice as to which is the best one to handle the degeneracy causing the non-uniqueness of the resulting triangulation often depends on the application; for example rejecting the point may be acceptable if we have a high point density.

Voronoi diagram and Delaunay triangulation can be used for modeling different kinds of data for different purposes. Many applications are possible or simplified when both structures are explicitly known. They can be used to discretise a continuous phenomenon such as the percentage of gold in the soil, the temperature of the water or the elevation of a terrain. They can also be used to represent the boundaries of real-world features, for example geological modelling of strata or cadastral models of apartment buildings.

Delaunay triangulations have been well known in the geosciences for many years, as are triangulated terrain models (Mostafavi et al 2003). Somewhat less known are the applications of Voronoi diagrams, although there are many scientists who consider them to be “a fundamental geometric data structure” (Aurenhammer 1987) – for example the work of Sambridge et al (1995).

The use of Delaunay triangulation is particularly suited when we do not want to force any constraints on the set of points to be connected i.e.
standard DT doesn’t allow edges that must appear in the triangulation to be specified in the point set. Hence, they can be directly used for smoothish natural surface. However, in surface representation, certain features of a terrain such as walls, cliffs, roads, etc. may have to be preserved, resulting in a need to accommodate non-Delaunay patches.

2.7 CONSTRAINED DELAUNAY TRIANGULATION

The typical domain for mesh generation is usually not a point set, but a polygonal region. In some cases the Delaunay triangulation of a nonconvex region's vertices contains a triangulation of the region, but in general Delaunay triangulation edges cross the region's boundary, creating an invalid mesh. A modification to the Delaunay Triangulation is the constrained Delaunay triangulation (CDT) wherein certain edges (breaklines) can be forced into the triangulation. Figure 2.15 depicts a simple example.

FIGURE 2.15 Edges are missing while their endpoints exist (Borouchaki et al 1995)
A *constrained* Delaunay triangulation of a set of line segments (which might form a polygon) is the Delaunay triangulation of the endpoints where the distance between two points is the length of the shortest path between them, which doesn't cross a line segment. In constrained Delaunay triangulation, it is possible to specify a set of constraints (edges, facets) to be included into the triangulation while maintaining all its properties. You get a triangulation in which the input segments are some of the edges. If you take the Voronoi diagram of the line segments themselves (that is, each region in the diagram is the set of points closest to some segment) you get a decomposition whose edges are lines. These are discussed in depth by Albertin and Wiggins (1994), Chew (1989), Lee and Lin (1986).

The construction of the constrained Delaunay triangulation is not straightforward. In spaces of dimension 3 or more, not every polyhedron can be discretized into constrained Delaunay triangulation without using additional points. Moreover, it was shown by Ruppert and Seidel (1992) that it is an NP-hard problem to identify such a case. Therefore the most common method is to build the Delaunay triangulation first and then modify it to enforce the constraints. This process is known as boundary recovery. In 2D, the boundary recovery can be accomplished purely by topological transformations (diagonal swapping) as shown by the shaded quadrilateral swapping in Figure 2.17. In
higher dimensional spaces, however, the recovery algorithm has usually to resort to insertion of additional points to refine the point set.

![Construction of Constrained Delaunay triangulation](image)

**FIGURE 2.17** Construction of Constrained Delaunay triangulation (Daniel and Zdenek 2005)

It is important to note that we ultimately wish to generate constrained Delaunay triangulations i.e. a modified Delaunay triangulation in which "constraints" may be specified. Since the Delaunay triangulation is unique for any point set (with the exception of sets with co-circular points), the constrained Delaunay triangulation will most likely contain some edges which are not Delaunay.

![Various triangulations](image)

**FIGURE 2.18** Various triangulations of a 2D set of points: a) general triangulation, b) constrained triangulation, c) Delaunay triangulation, d) constrained Delaunay triangulation (constraining edges are in bold) (Daniel and Zdenek 2005)
2.8 APPLICATIONS OF TRIANGULATIONS

- Surveying
- Cartography
- Rendering
- Triangulated Irregular Networks
- Visibility Determination
- Visualization
- Computer Graphics
- Computer aided design
- Representation of the geometry of geological structures
- Medical applications – representation of the anatomy of the human body
- Geographical information systems

During recent times advances in computer hardware and software have also brought triangulation technology into many new areas of application. A common challenge in all these areas is to construct the underlying geometry of the measured object. Many systems today use triangulation for this purpose.

2.9 SURFACE REPRESENTATION

Surface Representation deals with the digital representation of surfaces. The surface from which the data are sampled is known in some form such as a real model and the goal is to get a computer-based description of exactly this surface.
Surface modeling has become integral to ground engineering, hydrology, tectonics, oceanography, climatology and geo-hazard assessment. Surface data is used in a wide variety of applications such as earthworks, excavations, roads, pipelines, flood analysis, simulation, perspective views, fly-through, radio network planning, etc.

A terrain can be described as an extent of ground, region or territory (Petrie and Kennie 1991). It is part of the earth's surface. The five major features of terrain distinguished in maps are hills, saddles, valleys, ridges and depressions. Terrain also comprises of cliffs, overhangs and other constructed features such as cuts and fills, which result from the cutting-through of high areas and the filling-in of low areas to form a level bed for a road (Bourke 1987). Surface modeling is a general term to describe the process of representing a physical or artificially created surface by means of a mathematical expression. Terrain modeling is a particular category of surface modeling that deals with the specific problems of representing the surface of the Earth.

Terrain modeling is the study of ground-surface relief and pattern by numerical methods. It has become integral to hydrology, tectonics, oceanography, climatology, and geohazard assessment. It is also important to such non-geophysical applications as land-use planning, civil engineering, and microwave communications. Being the practice of ground-surface quantification, modern terrain modeling is an amalgam of earth science, mathematics, engineering, and computer science. It is commonly known as (quantitative) terrain analysis, (geo)morphometry, or quantitative geomorphology. It continues to grow through myriad applications to hydrology, geohazards mapping, tectonics, sea-floor and planetary exploration, and other fields.
Terrain models are discrete approximations of real terrains. Their specification is based on geometric data represented by a finite set of data points with associated elevation values. They play an important role for managing and visualizing geo-data in geo-information-systems as well as in virtual reality applications e.g. flight simulators.

The storage, display and analysis of data about the terrain surface is arguably one of the most widely used areas of GIS functionality. The characteristics of the terrain - its elevation, gradient and aspect - have a fundamental control on many aspects of environmental processes and also play a significant role in controlling man's activities. To fully model a surface, one would need an infinite number of points which in turn require infinite data storage.

**Where is Surface Representation used**

Surface models constructed from elevation data can be used for a variety of purposes including line-of-sight forecasting, slope and aspect determination, surface runoff pattern analysis and drainage basin delineation. They are used to relate thematic geo-data such as land use information or temperature fields with geometric data. Wise (1998) has dealt this at length.

Land surveyors have mapped much of the history of man’s use of the earth. A Babylonian boundary stone inscribed with the king’s decree and the name of the surveyor of the land still endures today after three thousand years! As civilization has expanded and matured, the surveying profession has similarly
kept pace with the need of society to define, delineate and map the land and man’s improvements thereon. Today, a land surveyor is equipped with many sophisticated surveying tools and knowledge required to provide services for land owners, developers, industry, and government. Whether the project be the layout of a new highway, bridge or dam, the precise location of offshore oil rigs or a municipality’s underground sewerage lines, a land surveyor is called upon to perform his unique role knowing that future generations may rely on the quality of his work for centuries to come.

2.10 TRIANGULATED IRREGULAR NETWORK

Surface meshing is an active area in mesh generation. Surface triangular meshing plays an important role in the areas of scientific computing, computer graphics, solid modeling, computer aided design, geographic information system, medical imaging, engineering analysis, finite element methods, numerical solutions of differential equations arising in physical simulation, etc. The goal is to approximate the surface with a mesh of simple elements such as triangles or quadrilaterals that approximates the underlying surface.

Triangulation is the most widely used form of unstructured mesh generation used in surface modeling, as any given arbitrary complex geometry can be more flexibly filled by triangular elements than by any other elements. When polygons are specified with more than three vertices, it is possible that the vertices may not all lie in one plane. Triangulation allows flexible handling of surface data and have been popular for some years. They are usually based on the Delaunay triangulation. For modeling terrain, arbitrary planar regions or
other objects given a set of sample points, the Delaunay triangulation gives a nice set of triangles to use as polygons in the model (Bentley 1979 and Cavendish 1983). The density and quality of elements in the region may be predictably controlled.

In surface representation, random elevation data is taken and this is approximated with a mesh of triangles, also known as a triangulated irregular network, or TIN. It is a system designed by Peucker et al (1978) for digital elevation modeling. TIN provides a versatile and widely used approach to represent terrain models in a way that retains the original sample points, adapts to variation in data density and incorporates linear features corresponding to natural or man-made phenomena. TIN is a vector data structure, which depicts geographic surfaces as contiguous non-overlapping irregularly shaped triangles computed from irregularly spaced 3D points. Within each triangle, the terrain surface is approximated by a plane defined by the three vertices of the triangle. The TIN data structure includes topological relationships between points and their neighboring triangles.

The edges of triangles in TINs can be adapted to be aligned with the breaklines (ridges, cliff lines) of the terrain to better represent the real features of the terrain. With the use of TINs it is also possible to accurately define irregularities such as sharp ridges and embankments. Compared to the traditional grid based techniques, terrain modeling based on TINs allow more adaptive modeling and flexible handling of terrain data and are useful for feature analysis such as silhouette extraction, visibility analysis and other possible applications. TINs eliminate the waviness around sharp surface breaks such as steep embankments, road edges, railway grades and hydrographic features.
The TIN model suits visualization purposes because of the continuous nature the triangular facets of the model add to the digital representation. In a TIN model, the sample points are simply connected by lines to form triangles, which are represented by planes that give continuous representation of the terrain surface. Creating a TIN, despite its simplicity requires decisions about how to pick the sample points from the original data set and further how to triangulate it. Rebecca et al (2004), Bern et al (1995), Herbert (2000), Christopher et al (2005) have discussed these.

One particular situation where many other techniques perform poorly is when there is a mixture of regions of high and low-density sampling. Triangulation based methods honour this situation by giving a large number of
triangles and hence more detail to the highly sampled regions and large triangles, less detail, to the regions with a few samples.

TIN allows extra information to be gathered in areas of complex relief. These information are presented basically by ridges and stream lines and can be incorporated to TIN as new triangle edges. The model considers ridges and stream lines as breaklines, because they break the smoothness of the surface. As the sample points in a TIN are irregularly distributed, more sample points can be used for jagged areas and fewer for flat areas.

The art of making good surface meshes is in keeping the number of faces to a minimum; a large number of faces cause a lot of problems when you try to use them to make animations. Also a lot of very long, acute triangles (sliver triangles) should be avoided: they are difficult to edit - changes in one part of the model will end up causing problems in other places.

2.11 CONTOURS

A piece of land on the earth’s surface can be modelled as a terrain (Gold 1979, Peucker et al 1977). In the past, survey information was collected and processed into topographic maps (Ron Singh et al 2000). Originally two separate maps were created:

- Planimetric Map - a 2 dimensional graphical representation of the positions on the earth’s surface of the natural and artificial features of a given locality.
• Topographic map – a map representing the configuration of the terrain or ground with contour lines constructed from plotted points of elevation. These methods depended on the assumption that there was a uniform slope between any two ground points located in the field. Spot elevations were usually given for critical points such as peaks, sags, streams and highway crossings.

Contouring has been the traditional way of presenting 3D terrain data on 2D sheets of paper. It has the advantage that precise height calculations can be made. Contour lines are usually drawn with line segments so they can easily be transferred to large-scale hardcopy devices such as plotters.

Currently, data is processed into digital terrain model with resultant products such as contours (Petrie and Kennie 1991, Ilfick 1979, Bourke 1987, Milne 1988). Digital representations of the terrain are central elements of the surveying process. Unlike in surface modeling, where a unique mathematical expression can often be used to define the feature of interest, it is difficult in terrain modeling to define precisely the structure of the terrain by a single global mathematical function. Nowadays, the modeling techniques are also used to create digital design models of proposed structures such as roads and buildings.
2.12 MODERN TOPOGRAPHIC MAPS

Most modern topographic maps are produced on a computer in 3 dimensions and can be viewed and plotted from any angle or position. The maps contain both planimetric and topographic features and show the horizontal distances between the features and their elevations above a given datum. Design systems can interact with these 3D maps to aid in the design of highways, etc. Topographic maps are not only essential to the planning, design and layout of projects, but are also essential to persons directing military operations. They are also of great benefit to some rescue operations and for recreational use. A topographic map without contours is a basemap.

2.13 DIGITAL TERRAIN MODEL

The commonest form of surface representation for elevation is the digital terrain model or DTM. A model is used to conceptualize certain aspects of reality. The terrain model is a mesh of interlocking triangles that provides an efficient approximation of the terrain surface.

A digital terrain model (DTM) is a topographic model of the bare earth that can be manipulated by computer programs. It is a 3D surface computed from a set of data points. Once a DTM has been created, contours, profiles, volumes between surfaces and 3D displays can be obtained easily. DTM's are used especially in civil engineering, geodesy & surveying, geophysics, geography and remote sensing.
Digital terrain models are discrete representations of the terrain surface i.e. land surface point elevations, as summarized by Peter and Gary (2003). A Digital Terrain Model is a numerical representation of the configuration of the terrain consisting of terrain features in terms of elevation and planimetric measurements obtained by sampling a topographic surface (Peucker 1977). In other words, it is a numerical representation of both planimetric details and height information that provides a continuous description of the terrain surface i.e. a very dense network of points of known X, Y, Z coordinates. DTM provides the means for representing the continuous surface of the earth in a digital form using a finite amount of storage i.e. a mathematical description of earth’s surface.

DTMs were originally used to represent the terrain surface for highway engineering purposes. But nowadays, DTM applications have been diversified in many fields, namely route engineering, landscaping, land surveying and mapping, military-purpose mapping, remote sensing, land and geographical information systems (LIS/GIS), etc.

Computers have made it possible to store the ground surface as a DTM. The rapid growth and fast emerging of DTM usage in vast fields is largely due to the advancement of recent computer technology both information computing power and graphics visualization capability. A computer processes the data into a form from which it can interpolate a three dimensional position anywhere within the model and display. Thus, a DTM is a topographic model of the bare earth that can be manipulated by computer programs. It is a 3D surface computed from a set of data points. DTMs require only that the lines and points required to describe the ground be stored. The software that analyzes this data will connect the data points with lines forming triangles. The software can then
analyze any given triangle to compute the elevation for any coordinate position within the DTM.

DTMs provide accurate, flexible, digital 3D data, which enables geospatial and analytical analysis over the terrain surface, like identification of the watershed area, physiographic and statistical amounts calculation, visualization, flood propagation model. Clematis et al (1998) and Masood et al (2005) have explained this. Digital Terrain Models may represent a common base to provide consistent information derived from combining different data.

Obtaining a DTM is a three-step process. The first step consists of acquiring three-dimensional coordinates that represent the area to be surveyed. The second step involves division of the terrain surface into simple sub-regions (e.g. triangles). The third step determines a piecewise polynomial function that describes a terrain approximation for each sub-region.

A Digital Terrain Model is a three dimensional digital model of the terrain surface which can be defined as the computational structure supporting the reconstruction of the whole terrain starting from the measured points. It has two purposes: it gives a method to approximate the real surface where data are missing and it suggests a logical structure for coding all the measures that have been collected. The surface is usually given in terms of large sets of data points, which either correspond to regular or random sampling.

Once a DTM has been created, contours, profiles, volumes between surfaces and 3D displays can be obtained easily. DTM may be used as a digital model of any single-valued surface e.g. geological horizons, rainfall or pressure,
population density. Digital terrain modeling encompasses general tasks including DTM generation, manipulation, interpretation, visualization and application. It is also a valuable component in analyses involving various surface characteristics such as profile, cross-section, line-of-sight, aspect and slope. The DTM (coupled with surface analysis tools) supports applications such as the development of topographic maps.

**DTM Products**

With a good DTM you can generate accurate contours and a host of other products. A digital terrain model can generate:

- Surface Triangles
- 3D Grid
- Cross Sections
- Flood Plains
- Water Catchment Areas
- Water Flow Models
- Pond Volumes
- Contours
- Profiles

**2.14 DATA STRUCTURES**

In order to efficiently handle geometric objects in triangulations there is the need for data structures supporting efficient operations on the objects and capable of handling queries on attributes or on the object's position within the domain.
A quick survey is done here on the existing options, to assess and compare their relative ease of implementation. (Detailed explanations of these algorithms can be found in computational geometry texts.)

There are many possibilities for a data structure to define a triangulation. The factors, which determine the best structure for storing a triangulation, include storage requirements and accessibility. In order to minimize storage, it is an accepted practice to store each point once only. Since these are floating point values, they take up the most space. Thus a basic triangulation structure would be a listing of the point set followed by the triangulation information.

### 2.14.1 Basic Triangulation Structure

A commonly used structure includes:

- A *point list* containing the coordinates of the vertices.
- A *triangle list* containing triples of pointers into the point list, where each triple indicates the vertices of a triangle.
- A *neighbor list* containing triples of pointers into the triangle list, where the $i^{th}$ triple indicates the triangles neighboring the $i^{th}$ triangle. *Also*, the $j^{th}$ entry in a triple corresponds to the neighbor opposite the $j^{th}$ vertex. If there is no neighbor, then this is marked by an entry $-1$.

This is discussed by Gerald Farin et al (1998).

The *point list* constitutes pointers into the point set, indicating which points are joined to form a triangle. The triangles are stored in a
counterclockwise orientation. The *neighbor list* gives a fast method to determine the boundary of the triangulation.

![Triangulation Diagram](image)

**FIGURE 2.21 A Triangulation Data Structure (Gerald Farin et al 1998)**

Figure 2.21 gives an example of the basic triangulation data structure. In actual implementation, the point list would consist of the coordinates of the points, i.e. each $P_i$ would be replaced by the respective coordinates. The triangle list indicates that the vertices of triangle 0 are the points P1, P6 and P4. The *neighbor list* indicates that triangle 0 has two neighbors: triangle 5 opposite point P1 and triangle 2 opposite point P4; there is no neighbor opposite point P6.

### 2.14.2 N-Tree Data Structure

This is a generic dynamic $n$-dimensional tree used to recursively divide the domain and provide efficient operations (mainly search) on geometric objects such as points, edges, triangles and tetrahedra. The N-Tree data structure is similar to the octree data structure, as they are both recursive space subdivision data structures. However in the N-Tree there is no fixed resolution for the subdivision, which means that there is no limit to the number of objects present at any moment in the tree.
This kind of data structure also provides an efficient way of checking object uniqueness; for example when an initial set of points is loaded to be triangulated, the points are all inserted in an N-Tree, duplicate points or points that are equal due to the limited machine precision or rounding error will fall into the same space partition; in this way duplicate points are eliminated and the initial requirement for the Delaunay triangulation satisfied.

The N-Tree is generic, as its main use is to store objects rather than represent them. This is true for the basic entities such as points, edges, triangles and so on. Nevertheless, as the Delaunay triangulation of a domain is itself stored in an N-Tree as a set of geometric entities (tetrahedra or triangles), we can also think of it as a way of representing an object, our domain in this case, with the full advantages of a dynamic data structure.
2.14.3 Winged Edge Data Structure

Perhaps the oldest data structure is Baumgart's winged-edge data structure (1975), which was able to efficiently represent complex topological polyhedra. Winged-edge data structures are frequently the preferred representation for three-dimensional models. This is partly because winged-edge models make it relatively easy to implement adaptive mesh refinement techniques. It quickly traverses the adjacency relations of complex geometric objects and effectively represents complex topological polyhedra.

A polyhedron is made up of four kinds of nodes: bodies, faces, edges and vertices. The body node is the head of three rings: a ring of faces, a ring of edges and a ring of vertices. In this context, a ring is a doubly linked circular list with a head node. Each face and each vertex points directly at only one of the edges on its perimeter. Each edge points to its two faces and its two vertices. Completing the topology, each edge node contains a link to each of its four immediate neighboring edges clockwise and counterclockwise about its face perimeter as seen from the exterior side of the surface of the polyhedron. These last four links are the wings of the edge, which provide the basis for efficient face perimeter and vertex perimeter accessing. Finally, the links of the edge nodes can be consistently oriented with respect to the surface of the polyhedron so that the surface always has two sides: the inside and the outside.

The winged-edge data structure uses edges to keep track of almost everything. In what follows, it is assumed that there are no holes in each face. Edges and faces are assumed to be line segments and polygons. Topologically, one can always stretch curvilinear edges and faces so that they become flat without changing the relationships among them.
Figure 2.23 shows a polyhedron with vertices, edges and faces indicated with upper cases, lower cases and digits respectively. Let us take a look at edge \( a = XY \). This edge has two incident vertices \( X \) and \( Y \) and two incident faces 1 and 2. A face is a polygon surrounded by edges. For example, face 1 has its edges \( a, c \) and \( b \), and face 2 has its edges \( a, e \) and \( d \). Note that the ordering is clockwise viewed from outside of the solid. If the direction of the edge is from \( X \) to \( Y \), faces 1 and 2 are on the right and left side of edge \( a \) respectively. To capture the ordering of edges correctly, we need four more pieces of information. Since edge \( a \) is traversed once when traversing face 1 and traversed a second time when traversing face 2, it is used twice in different directions. For example, when traversing the edges of face 1, the predecessor and successor of edge \( a \) are edge \( b \) and edge \( c \), and when traversing the edges of face 2, the predecessor and successor of edge \( a \) are edge \( d \) and edge \( e \). Note that although there are four edges incident to vertex \( X \), only three of them are used when finding faces incident to edge \( a \). Therefore, for each edge, the following information are important:

1. vertices of this edge,
2. its left and right faces,
3. the predecessor and successor of this edge when traversing its left face, and
4. the predecessor and successor of this edge when traversing its right face.

The Edge Table
Each entry in the edge table contains those information mentioned earlier: edge name, start vertex and end vertex, left face and right face, the predecessor and successor edges when traversing its left face, and the predecessor and successor edges when traversing its right face. Note that clockwise ordering (viewing from outside of the polyhedron) is used for traverse. Note also that the direction of edge \(a\) is from \(X\) to \(Y\). If the direction is changed to from \(Y\) to \(X\), all entries but the first one in the following table must be changed accordingly.

![Figure 2.24: Edge table entries for Edge a](image)

<table>
<thead>
<tr>
<th>Edge Name</th>
<th>Vertices</th>
<th>Faces</th>
<th>Left Traverse</th>
<th>Right Traverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>X Y</td>
<td>1 2</td>
<td>b (\rightarrow) d</td>
<td>e (\rightarrow) c</td>
</tr>
</tbody>
</table>

Figure 2.24 shows the information for the entry of edge \(a\). The four edges \(b, c, d\) and \(e\) are the ‘wings’ of edge \(a\) and hence edge \(a\) is ‘winged’.
With this data structure, one can easily answer the question: which vertices, edges, faces are adjacent to each face, edge or vertex. Note also that once the numbers of vertices, edges and faces are known, the size of all three tables are fixed and will not change.

Figure 2.25 is a tetrahedron with four vertices A, B, C and D, six edges a, b, c, d, e and f, and four faces 1, 2, 3 (back) and 4 (bottom). Its edge table is the following:

TABLE 2.1 Edge Table

<table>
<thead>
<tr>
<th>Edge Name</th>
<th>Vertices</th>
<th>Faces</th>
<th>Left Traverse</th>
<th>Right Traverse</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pred</td>
<td>Succ</td>
</tr>
<tr>
<td>a</td>
<td>A D</td>
<td>3 1</td>
<td>e</td>
<td>f</td>
</tr>
<tr>
<td>b</td>
<td>A B</td>
<td>1 4</td>
<td>c</td>
<td>a</td>
</tr>
<tr>
<td>c</td>
<td>B D</td>
<td>1 2</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>d</td>
<td>B C</td>
<td>2 4</td>
<td>e</td>
<td>c</td>
</tr>
<tr>
<td>e</td>
<td>C D</td>
<td>2 3</td>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>f</td>
<td>A C</td>
<td>4 3</td>
<td>d</td>
<td>b</td>
</tr>
</tbody>
</table>
Other Tables

The winged-edge data structure requires two more tables, the vertex table and the face table. These two are very simple. The vertex table has one entry for each vertex which contains an edge that is incident to this vertex. The face table has one entry for each face which contains an edge that is one of this face's boundary edges. Therefore, we have the following table. Note that since there are multiple choices of edges, you may come up with different tables:

<table>
<thead>
<tr>
<th>Vertex Name</th>
<th>Incident Edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>a</td>
</tr>
<tr>
<td>B</td>
<td>b</td>
</tr>
<tr>
<td>C</td>
<td>d</td>
</tr>
<tr>
<td>D</td>
<td>e</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Face Name</th>
<th>Incident Edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>a</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
</tr>
</tbody>
</table>

With this data structure, one can easily answer the question: which vertices, edges, faces are adjacent to each face, edge, or vertex. There are nine of these adjacency relations. For example, is vertex X adjacent to face 5? Are faces 3 and 5 adjacent to each other? The winged-edge data structure can answer these queries very efficiently and some of them may even be answered in constant time. However, it may take longer time to answer other adjacency queries. Note also that once the numbers of vertices, edges and faces are known, the size of all three tables are fixed and will not change.

The two symmetric parts in the winged-edge correspond to the two possible orientations of the edge. The inefficient case distinction in the traversal
computation results from the fact that a pointer to an edge does not encode the orientation it is currently used with.

Thus, edge is the primary data structure. Edge nodes carry most of the topology, vertex nodes carry the geometry, face nodes carry the photometry and body nodes carry the nomenclature and parts in tree structure. The primary concern is "fast extraction of topological information". This model makes it relatively easy to implement meshing algorithms. The earliest usage of this data structure is for boundary representations in solid modeling (Hoffman 1989).

2.14.4 Half-edge Data Structure

The orientation problem for the winged-edge data structure can be solved by splitting the edge into two symmetric records, called halfedges and adding mutual links to each other (Weiler 1985). There are two ways of splitting the edge: Either the edge is split along the facets such that the oriented halfedges belong to the two facets incident to this edge called the FE-Structure (Figure 2.26a) or it is split into two halfedges belonging to the two vertices incident to
this edge, called the VE-Structure (Figure 2.26b). These two variants are actually dual to each other considering the usual notion of duality for graphs, where vertex and facet are dual to each other and the dual of an edge is an edge with the two incident facets as its endpoints.

In both splitting variants a halfedge contains a pointer to an incident vertex, an incident facet, and the opposite halfedge. It is a matter of convention whether the source or target vertex is the one chosen to be stored in a halfedge or whether the facet to the left or the right is stored.

Conventions used in the halfedge data structure are depicted in Figure 2.27. The incident vertex is the target vertex of the oriented halfedge. The incident facet is to left of the halfedge which implies a counterclockwise ordering of the halfedges around the facet and a clockwise ordering around the vertex when seen from the outside. This complies with the right-hand rule for out-facing normals of plane equations for facets.
The halfedge data structure is able to model orientable 2-manifolds. It is sufficient for modeling topology even in presence of loops and multi-edges, which can occur in curved-surface environments. High-level operations maintaining integrity are again Euler operators. The Minimal Rendering Tool MRT (Kettner 1998) uses a halfedge data structure for polygonal surfaces.

2.14.5 Quad-edge Data Structure

Refinements to the winged edge idea were made by Guibas and Stolfi (1985) with the introduction of the quad-edge structure. It is a particularly elegant data structure for describing the topology and geometry of polyhedra. It can answer adjacency queries efficiently.

The quad-edge data structure can't represent all collections of polygons; it is limited to manifolds (surfaces where the neighborhood of each point is topologically equivalent to a disk; edges are always shared by two faces). It was designed for representing general subdivisions of orientable manifolds. Though similar to the winged-edge data structure, it simultaneously represents both the subdivision and its dual. Each quad-edge record groups together four directed edges corresponding to a single undirected edge in the subdivision and to its dual edge (Figure 2.28a). Each directed edge has two pointers: a next pointer to the next counterclockwise edge around its origin and a data pointer to geometrical and other nontopological information (such as the coordinates of its origin). Figures 2.28b and 2.28c illustrate how three edges incident on the same vertex are represented using the quad-edge data structure: the vertex itself corresponds to the inner cycle of pointers in Figure 2.28c. The remaining three cycles correspond to the three faces meeting at the vertex.
The Quad-Edge data structure captures all the topological information of the subdivision of a surface. It provides a fully symmetric view on the primal and the dual graph as can be seen from Figure 2.29.

Aside from a primitive to create an edge (MakeEdge), a single topological operator Splice is defined that can be used to link disjoint edges together as well as to break two linked edges apart. This operator is its own inverse and together with MakeEdge it can be used to construct any subdivision.
* Edge $e$ - directed edge
* Edge $e$\rightarrow Sym() - edge pointing opposite to $e$
* Edge $e$\rightarrow Rot() - dual edge pointing to the left of $e$
* Edge $e$\rightarrow InvRot() - dual edge pointing to the right of $e$

FIGURE 2.30 Edge Operators

FIGURE 2.31 Initial Configuration of Quad-Edge Data Structure
Figure 2.31 shows only the subdivision and corresponding edge records shaped as crosses. Figure 2.32 illustrates the vertices added to the data structure. Vertices are actually loops that are connected to the edges that the points are connected in the original subdivision.

FIGURE 2.32 Vertices added to the Quad-Edge Data Structure
The quad-edge data structure, thus, provides all necessary information for a given edge. The quad-edge data structure is popular because it is elegant and simultaneously represents a graph and its geometric dual (such as a Delaunay triangulation and the corresponding Voronoi diagram). The quad-edge data structure gets its name because the duality is built in at a low level by storing quadruples of directed edges together.

Salamanca (1990) studied the quad-tree data structure specifically for DTM. Details of these structures have been referred by Mark (1976), Peucker et al (1979), Samet (1984), Cebrian et al (1985) and Chen and Tobler (1986).
2.14.6 Triangular Data Structure

Each triangle record contains six pointers: three pointers to neighboring triangles and three pointers to vertices. A triangulation contains roughly three edges for every two triangles.

The triangular data structure is superior in time as well as space. When a program makes structural changes to a triangulation, the amount of time used depends in part on the number of pointers that have to be read and written. This amount is smaller for the triangular data structure; more of the connectivity information is implicit in each triangle.

2.14.7 Vertex-Edge-Face Table

A simple data structure is a vertex-edge-face table (Philip and David 2002). The $N$ unique vertices are stored in an array, Vertex[0] through Vertex[N-I]; so vertices can be referred to by their indices into the array.
Edges are represented by pairs of vertex indices and faces are represented by ordered lists of vertex indices. The table is defined by the grammar:

```
VertexIndex = 0 through N - 1;
VertexIndexList = EMPTY or { VertexIndex V, VertexIndexList VList; }
EdgeList = EMPTY or { Edge E, EdgeList EList; }
FaceList = EMPTY or { Face F, FaceList FList; }
Vertex = { VertexIndex V, EdgeList EList, FaceList FList; }
Edge = { VertexIndex V[2], FaceList FList; }
Face = { VertexIndexList VList; }
```

The edge list EList in the Vertex object is a list of all edges that have an end point corresponding to the vertex indexed by V. The face list FList in the Vertex object is a list of all faces that have a vertex corresponding to the vertex indexed by V. The face list FList in the Edge object is a list of all faces that share the specified edge. An Edge object does not directly know about edges sharing either of its vertices. A Face object does not know about vertices or edges that share the face's vertices. This information can be indirectly obtained by various queries applied to the subobjects of either Edge or Face.

By the definition of a polymesh, the face list in Edge cannot be empty since any edge in the collection must be part of at least one face in the collection. Similarly, the edge and face lists in Vertex must both be nonempty. If both were empty, the vertex would be isolated. If the edge list were not empty and the face list were empty, the vertex would be part of an isolated polyline and the immediately adjacent edges have no faces containing them.

The edges can be classified according to the number of faces sharing them. An edge is a **boundary edge**, if it has exactly one face using it. Otherwise, the edge is an **interior edge**. If an interior edge has exactly two faces sharing it, it
is called a manifold edge. All edges of a polyhedron are required to be of this type. If an interior edge has three or more faces sharing it, it is called a junction edge.

One important question is whether one should choose a data structure that uses a record to represent each edge or one that uses a record to represent each triangle. Kettner (1998) gives an excellent overview and comparison of different mesh data structures.