

GEOMETRIC APPROACHES TO MESH GENERATION

CHRISTOPH M. HOFFMANN*

Abstract. We review three approaches to mesh generation that are based on analyzing and accounting for the geometric structure of the domain. In the first approach, due to Armstrong, the domain is partitioned into subdomains based on the medial-axis transform, a tool for analyzing spatial structures. In the second approach, due to Cox, the design history defines a geometric structure of the domain. The design primitives of that structure are meshed separately, and mesh overlap is accounted for by coupling equations. The third approach argues that mesh generation ought to be integrated into the shape design process, by meshing design features separately and resolving overlapping meshes by standard geometric computations.

1. Introduction. The problem of meshing a geometric domain has two aspects, a physical aspect that accounts for the behavior of the solution of the physical problem, and a geometric aspect that accounts for the shape of the domain. Applications, such as in manufacturing, not only involve analyzing specific domains in two or three dimensions, but also involve design computations that produce the shape in the first place. Despite the fact that applications require both, the more geometric activity of designing a shape and representing its geometry has developed separately from the analysis side that is developing techniques to solve physical problems by numerical or semi-numerical techniques. It is unfortunately rare to find workers versed in both the intricacies of the geometric side as well as the physical side of the problem.

In this paper we pay attention to the geometric side of the problem, primarily because of the perceived need to create a greater awareness of the geometric side of things in the community of numerical analysts and applied mathematicians. We consider three different approaches.

In the first approach, the geometric structure of the domain is analyzed using the medial axis transform, a concept made popular in computer vision, but found elsewhere in a variety of equivalent or closely-related formulations. Here, we discuss the work of Cecil Armstrong and his colleagues, although other researchers and groups have pursued a similar tack and employed the medial axis transform as well.

In the second approach, a specific design paradigm is coupled with the process of mesh generation. A domain is thought of as a Boolean combination of primitive shapes, each easily meshed. In combination, the domain is then covered with a number of overlapping meshes, and the physical problem formulation resolves the overlap by certain coupling equations that

* Supported in part by ONR Contract N00014-90-J-1599, by NSF Grant CCR 86-19817, and by NSF Grant ECD 88-03017.

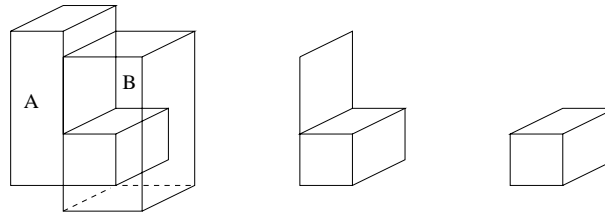
force compatibility of the solution in the overlapped region. We discuss here work by Jordan Cox, but also refer the reader to work by others, in particular the work of William Henshaw in this volume.

In the third approach, we discuss some of the modern feature-based design paradigms that are evolving especially in mechanical design. Having argued elsewhere that this design paradigm ought to be supported by a separate representation, [15], we advocate here that the high-level feature representation would be directly translated into finite-element meshes, so that the mesh is built up in step with the creation of the domain itself. This requires only a few additional operations and produces, in contrast to the second approach, nonoverlapped, compatible meshes.

We begin our exposition with a brief summary of geometry representations. While in two dimensional situations it is quite straightforward to devise simple and intuitive shape representations, in three-space the representation of geometric shapes becomes cumbersome and, at first glance, may appear overly complex to the nonspecialist. Nevertheless, to-date simpler three-dimensional representations have not been found, at least not without accepting severe restrictions on the geometric coverage; i.e., on the range of possible shapes.

2. Solid Modeling Representations. Solid modeling has produced three major families of shape representations, constructive solid geometry, boundary representation, and spatial subdivision. There are other representations that completely define a solid, three-dimensional domain; for instance, the medial-axis transform. As these are not in wide use, we will not discuss them, except to the extent that they are relevant to specific aspects of mesh generation. Also omitted is a description of representations such as wire frames, that do not define a solid shape unambiguously.

2.1. Constructive Solid Geometry. In *constructive solid geometry* (CSG), [24], a complex shape is built from primitive shapes by operations of union, difference and intersection. The primitives are specified by a few shape parameters. Customarily, the primitives are a block, parameterized by three side lengths; a sphere, parameterized by radius; a cylinder or a cone, each parameterized by radius and height; and a torus, parameterized by major and minor radius. One assumes that each of these primitives are at a default location in a local coordinate frame. The local frames are related to a global frame by rotation and/or translation, and the primitives so positioned are combined with the operations of regularized union, regularized difference, and regularized intersection. A regularized operation differs from a set-theoretic operation in that lower-dimensional structures of the result are “removed.” For instance, in Figure 2.1 the set-theoretic intersection of the block B and the L-shaped object A is shown in the middle. It consists of a block and an attached “dangling” face. The regularized intersection is shown on the right, and has no isolated lower-dimensional structures. To obtain a regularization one computes conceptually the clo-

FIG. 2.1. *Set-theoretic and Regularized Intersection of two Shapes*

sure of the interior of the set-theoretic result. In practice, regularization is incorporated into the algorithms that implement the Boolean operation. The details depend on the representation.

In pure CSG, a complex shape is simply an algebraic expression formed from operands that are the name and parameter value(s) of the primitives used, and from operators that represent the regularized Boolean operations and rigid-body motions. The expression can be represented internally by a tree, and a number of geometric operations, such as testing whether a given point is inside, outside, or on the surface of a three-dimensional object, can be implemented as a suitable tree-traversal. For details see, e.g., [10,19]

2.2. Boundary Representation. In *boundary representation* (Brep) one describes the surface of a solid domain as a collection of faces, edges, and vertices, along with the adjacencies between them. There are many variants differing in detail conventions, but all describe the surface by specifying faces, edges and vertices and their adjacencies; see, e.g., [10,19].

The description of a face has two parts. A surface is specified of which the face is a subset. The surface can be an implicit, a parametric, or a procedurally defined surface. In addition, the boundaries of the face are described, by edges and vertices. In some versions of a Brep, the bounding edges and vertices are organized into closed loops, along with information on the nesting of loops.

The description of an edge consists of the definition of a space curve of which the edge is a segment, and of the vertices bounding the segment. The curve might be the intersection of two surfaces, or a parametric space curve, or a procedurally defined curve. A vertex is typically described by point coordinates. Some versions restrict the topological structure of a face to be homeomorphic to a unit disk with zero or more internal holes.

An elaborate convention of orientations designates on which side of a face to find the interior of the solid, and on which side the exterior. Furthermore, orientation conventions tell on which side of an edge, embedded in the surface of the face, to find the face interior. Likewise, one can determine on which side of a vertex, on a space curve, to find the interior of an edge. These conventions are in part explicit, and in part implicit. For example, at a vertex it is not uncommon that a computation on the

incident faces and edges is needed to determine a direction into the interior of the solid.

In early boundary representation schemes the surface was required to be a closed, orientable manifold in 3-space. This restriction is too narrow in that regularized Boolean operations with such “manifold solids” can result in nonmanifold solids which would then be invalid objects. Therefore, nonmanifold boundary representations are becoming the norm.

Most commercial solid modelers, and many research solid modelers use a Brep as internal solid representation despite the greater demands on storage. One of the factors influencing this decision is the option, in Brep, to use faces that are part of a spline surface, so that the large variety of shapes studied in computer-aided geometric design can be used in solid modeling. This increases the scope of solid modelers, and is required in applications such as aerospace and shipbuilding, as well as in the design of automobile bodies.

2.3. Spatial Subdivision. Some solid modeling systems use a spatial subdivision scheme. In such a scheme, the volume of the solid is represented as the union of adjacent, nonoverlapping cells. If the cells have a fixed shape such as cubes oriented along the principal Cartesian directions, then the representation is typically approximate. When the cubes are regular in size, we obtain voxel representations; e.g., [17]. If the cubes are obtained by an adaptive subdivision of a large cube, we obtain octree representations; e.g., [4,25].

Irregular subdivisions can be boundary-conforming. Here, a given shape is (nearly) exactly the union of cells of irregular size and orientation. For example, the binary space partition tree [21] is such a representation and can represent any polyhedral shape exactly. Subdivisions such as Delaunay triangulation of domains, discussed by others in this volume, also represent polyhedra exactly. By mapping techniques, e.g. [30], curved domains can also be represented exactly.

Subdivision representations have not been used widely in solid modeling. They are the representation of choice in analysis problems solved by numerical integration.

2.4. Dual-Purpose Representations. Nonmanifold boundary representations, e.g., [31], have been advocated as representations that serve both the needs of analysis and of solid modeling. Since in such representations faces internal to the solid are permitted, nonmanifold boundary representations can represent spatial subdivisions. However, since they conform to the requirements of boundary representations, with complex data structures to designate face areas, edges and adjacencies, it seems that using nonmanifold Breps for complete meshes would unnecessarily add to the storage requirements. However, a subdivision of the domain into a small number of subdomains, each containing a part of the mesh in a more traditional representation, might advantageously be represented

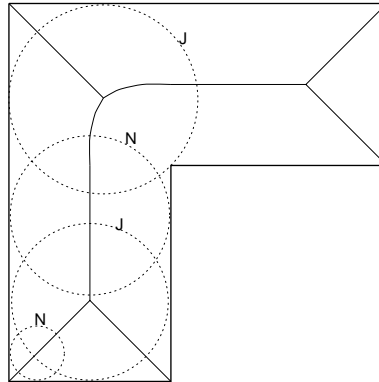


FIG. 3.1. *L-shaped Domain, Its Medial Axis, and Several Maximal Circles*

using nonmanifold Breps.

3. MAT-Based Mesh Generation. The *medial axis transform* (MAT) is a shape abstraction introduced by Blum [3] in computer vision. The concept plays a role in some approaches to mesh generation because it provides an algorithmic way to partition domains into subdomains that are relatively easy to mesh, and compatibly so. We explain the basic concepts, and then discuss in some detail Armstrong’s method for meshing two- and three-dimensional domains [2,1]. For other approaches to mesh generation using the MAT see [22,28,32].

3.1. Medial-Axis Transform. Let S be a compact two-dimensional domain with continuous boundary of finite length. The *medial axis* (MA) of S is the closure of the locus of all maximal inscribed disks. An inscribed disk D is maximal if there is no other inscribed disk D' that properly contains D . See also Figure 3.1. Medial axis points can be classified by type, [3]. A *normal* point is the center of a disk that touches the boundary in two distinct points. In the figure, two such disks are shown labeled N. A *junction* or branch point is the center of a disk that touches the boundary in three or more points. Two such disks are shown in the figure labeled J. An *end* point is one whose disk has zero radius or whose disk touches the boundary in exactly one point. The MA points at the five convex corners of the domain are an example. A point has *finite contact* if it is the center of a disk that touches the domain boundary in an arc of nonzero length.

The *medial axis transform* (MAT) of S is the medial axis of S plus a function that assigns to each point of the medial axis the radius of the disk centered there. Both concepts generalize to three-dimensional domains when considering inscribed spheres in place of inscribed disks.

Blum considered the MAT as shape abstraction, and conjectured that the native representation of the human vision system was related to the MAT. The medial axis transform of a 2D domain can be thought of as a

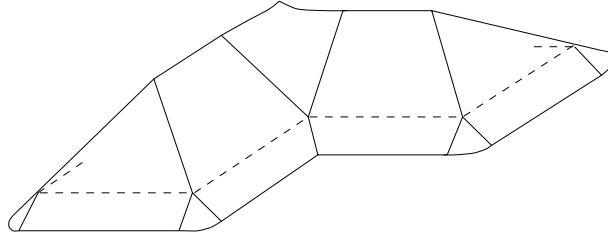


FIG. 3.2. *MAT of the L-shaped Domain, as Singularities of the Euclidean Distance Function*

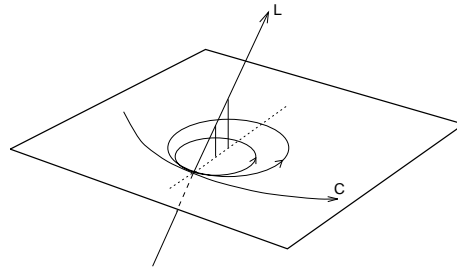


FIG. 3.3. *The Cyclographic Map at a Curve Point*

3D graph, by considering the radius a third coordinate. To obtain a closely related concept, we assign to each point of the domain its minimum distance to the domain boundary as value. We so obtain the Euclidean distance function of the boundary. With the convention that interior points have positive distance and exterior points negative distance, the graph of the distance function is a certain surface, as illustrated in Figure 3.2. Offsets of the boundary are then the intersection of the distance surface with a parallel plane whose elevation above (or below, for exterior offsets) is equal to the offset distance. The medial axis transform is simply the locus of first-order discontinuities of the distance function. Thus, we can also think of the medial axis as shock waves of a wave front that initially is on the domain boundary and propagates inward.

Around the turn of the century, Müller [20] formulated the concept of cyclographic map, developing ideas dating back to Laguerre. Given an oriented curve C in the plane, Müller considers oriented circles tangent to the curve. At a curve point, the centers of these circles are on the curve normal. He associated with each circle a point in 3-space above the center at a distance equal to the radius, for positively oriented circles. For negatively oriented circles, the point is below the center at a distance equal to the radius. All such associated points therefore lie on a line L that has an angle of 45° with the plane and projects orthographically onto the curve normal, as illustrated in Figure 3.3. Thus, an oriented curve is mapped to

a ruled surface in 3-space which defines the cyclographic map of the curve [20]. A subset of the surface is the graph of the Euclidean distance function, and the MAT is part of the singularity structure of the cyclographic map.

In [5,6,12], it is proposed to compute the medial axis transform based on Danielson’s algorithm and on the dimensionality paradigm; [8,11,16]. Other approaches include approximating the medial axis points from the circum centers of Delaunay triangles when triangulating a point set dense in the domain boundary; e.g., [29]. Note, however, that a topological classification of the Delaunay triangles is required to locate “missing” sections of the MAT not so approximated. When the geometric elements comprising the domain boundary are suitably restricted, other algorithms are possible; e.g., [23,18,27,22].

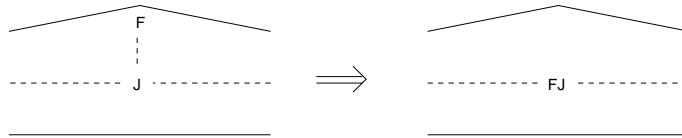
3.2. Armstrong’s Mesh Generation. Armstrong considers quadrilateral mesh generation in 2-space, and hexahedral mesh generation in 3-space. Both algorithms have the same overall structure, but the classification work in 3-space is much more complex than in 2-space. Srinivasan [27] and Patrikalakis and Gürsoy [22] have MAT-based meshing algorithms that differ both in the type of meshing and in the way in which the domain is partitioned. Those algorithms have not been extended to 3D domains.

3.2.1. 2D Meshing. Armstrong’s algorithm proceeds as follows:

1. The domain boundary is discretized and from the points on it a Delaunay triangulation is constructed.
2. By classifying how certain triangles touch the domain boundary, branch points of the MA are determined, as well as how the branch points are interconnected. This classification is used throughout the algorithm.
3. At highly concave corners the domain is subdivided by an internal split.
4. MA branch points and end points of the domain are analyzed, and, depending on type and configuration, the domain is subdivided into subdomains that are 3-, 4-, 5-, or 6-sided.
5. For each type of subdomain, a standard mesh is determined. Compatibility across subdomains is achieved by formulating integer constraint equations and solving an integer programming problem.

Boundary Delaunay Triangulation. In a Delaunay triangulation of a point set, the circumcircle of each triangle does not contain other points of the triangulation. Therefore, when the points are dense in the domain boundary, the circumcircles approach maximal inscribed circles. Note that certain segments of the MA cannot be approximated in this way [12].

Armstrong has his own variant of constructing the triangulation in order to integrate it with a classification of the triangles and determining MA branch points. The boundary discretization is adaptive, and the

FIG. 3.4. *Elimination of Shallow Convex Corners*

triangulation algorithm is incremental.

Triangle Classification. Triangles are classified into one of 5 types. The types are derived based on how the triangles touch the domain boundary, and are related to the type of MA point the triangles' circumcenter is close to.

A triangle is of type J (junction point) if the three vertices touch three different parts of the boundary, and none of the sides is on the boundary. A triangle is of type C (convex corner) if one of its vertices is a convex corner and the adjacent sides are on the boundary. A triangle is of type I (intermediate) if one of its sides is on the boundary and is connected to a concave vertex, and the three vertices are on three different parts of the boundary. A triangle is of type N (normal point) if two vertices and the connecting side are on the same part of the boundary, whereas the third vertex is on a different part of the boundary. A triangle is of type T (topologically redundant) if all three vertices and two of its side are on the same part of the boundary.

After this initial classification, triangles are further classified based on their adjacencies and the local geometry. For example, the type C is reclassified F at shallow corners that are nearly 180° . Ignoring N-triangles, an adjacency graph is constructed that is a topological representation of the medial axis. We call this graph the MAT graph.

Splitting Concave Corners. For the meshing algorithm, a concave corner is one at which the incident sides subtend an interior angle of 216° or more. Incident to the corner are triangles of type I. The sides of such triangles are candidates for splitting the corner. For each choice, the resulting element angles are computed and the chosen split minimizes the deviation from 90° .

Domain Subdivision. A number of transformations are applied to the MAT graph. They include transformations such as Figure 3.4, or Figure 3.5 that eliminate irrelevant parts of the medial axis, and transformations such as Figure 3.6, that break up complex topologies.

Here, E is a type that marks an end point of the MAT at which a specific mesh pattern will be applied. Similarly, F marks a flat corner that is nearly 180° . FJ is a junction with a flat corner, at which topologically the two sides of the flat corner are treated as a single edge.

Eventually, the domain has been partitioned into one of nine types of

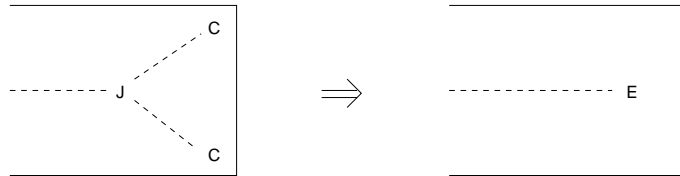


FIG. 3.5. Treatment of MA Ends

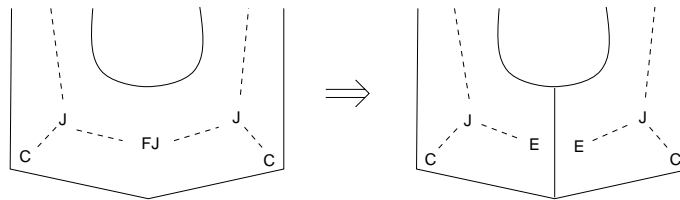


FIG. 3.6. Splitting Chains of J-nodes

subdomains, shown in Figure 3.7. Armstrong calls these subdomains *shape atoms*. Each subdomain is meshed with a standard mesh of quadrilaterals.

Meshing Patterns. Subdomains with fewer than three sides are subdivided. Then, the subdomains are meshed using midpoint subdivision. For a triangular subdomain, the pattern is shown in Figure 3.8. Note that compatibility conditions must be satisfied by the mesh pattern parameters. In the case of the triangular subdomain, the compatibility equations are

$$(3.1) \quad \begin{aligned} m_1 &= n_2 \\ n_1 &= p_2 \\ p_1 &= m_2 \end{aligned}$$

At each internal edge of the domain, the number of elements adjacent to the edge in one subdomain must agree with the number of elements meeting the edge from the adjacent subdomain. This requirement is expressed by a set of equations. For example, if the triangular subdomain of Figure 3.9 is adjacent to the pentagonal subdomain as shown, then we would have to satisfy

$$(3.2) \quad m_1 + m_2 = M_1 + M_2$$

Note that the midpoint subdivisions do not have to align.

The equations of type (3.1) and (3.2) together form a system that defines the constraints of an integer programming problem. Target values for some of the variables can be computed from edge lengths and desired element size. After solving the integer problem, a quadrilateral mesh of the domain has been found.

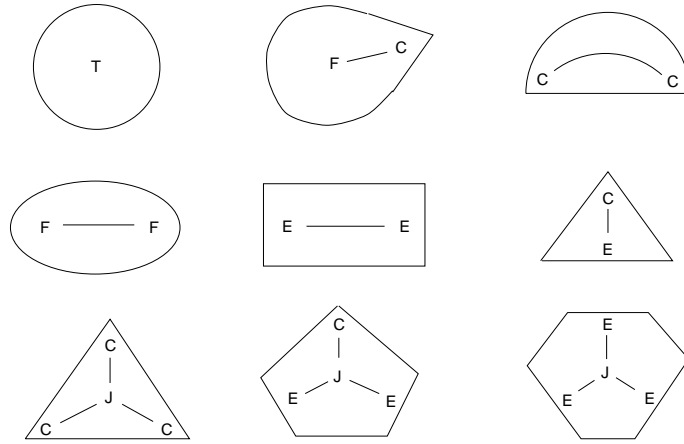


FIG. 3.7. Possible Shapes of Subdomains; from [2]

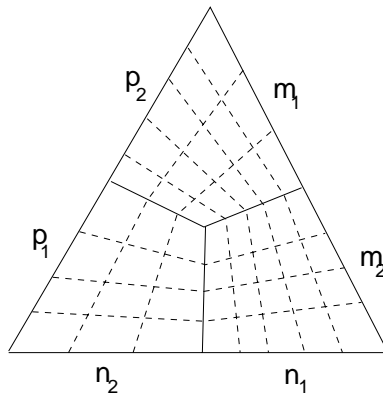


FIG. 3.8. Mesh Pattern by Midpoint Subdivision

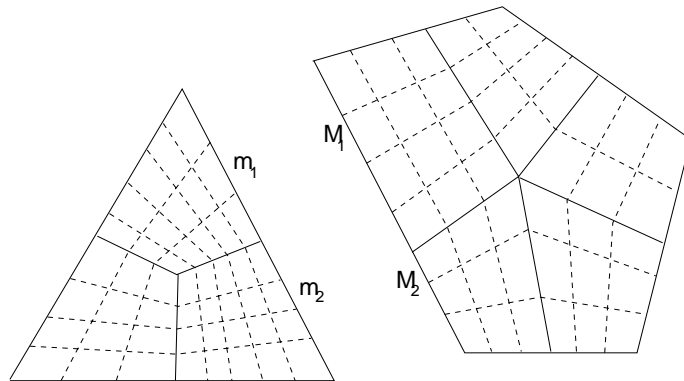


FIG. 3.9. Mesh Compatibility Requires $m_1 + m_2 = M_1 + M_2$

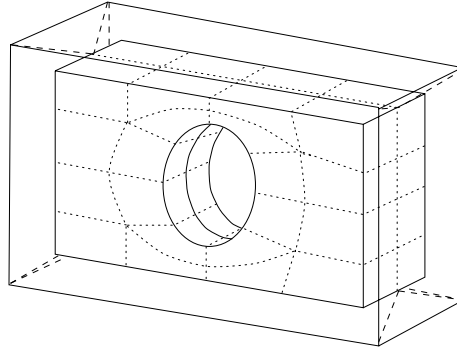


FIG. 3.10. *Extruding the 2D Mesh on an MAT Face; from [1]*

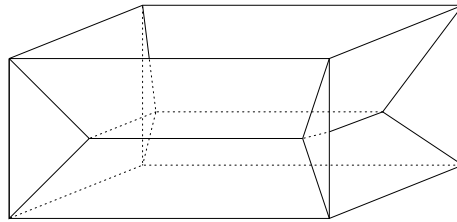


FIG. 3.11. *MA of a Parallelepiped*

3.2.2. 3D Meshing. The 3D Meshing algorithm is structured in the same way as the 2D algorithm; [1]. A central idea is as follows. The MAT of a 3D domain will have faces in addition to edges and vertices. If we mesh an MAT face as if it were a 2D domain, then the mesh can be extruded into columns over each quadrilateral of the face mesh extending to the boundary on either side. Clearly these columns can be divided into hexahedral elements. Figure 3.10 illustrates the idea.

This simple idea ignores some complications: Not all MAT faces are suitable to this idea. For example, the MAT of a parallelepiped has a total of 13 faces, one of which is centrally in the interior, and the other twelve connect to the edges of the parallelepiped, as illustrated in Figure 3.11. Only the central face is amenable to mesh extrusion.

Corresponding to the corner treatment in the 2D case, specific edges of the MAT in the 3D case must be treated specially and some of the adjacent MAT faces should be ignored. In the case of the parallelepiped, this is the 3D analogue of the MAT graph transformation that reclassified a J graph node with two adjacent C nodes as an E node. Vertices of the MAT similarly require special treatment. The resulting cases are considerably more numerous and complex than in the 2D case.

Note also, that meshing an MAT face is more complicated than meshing a 2D domain because the face can be curved, requiring a geodesic version of

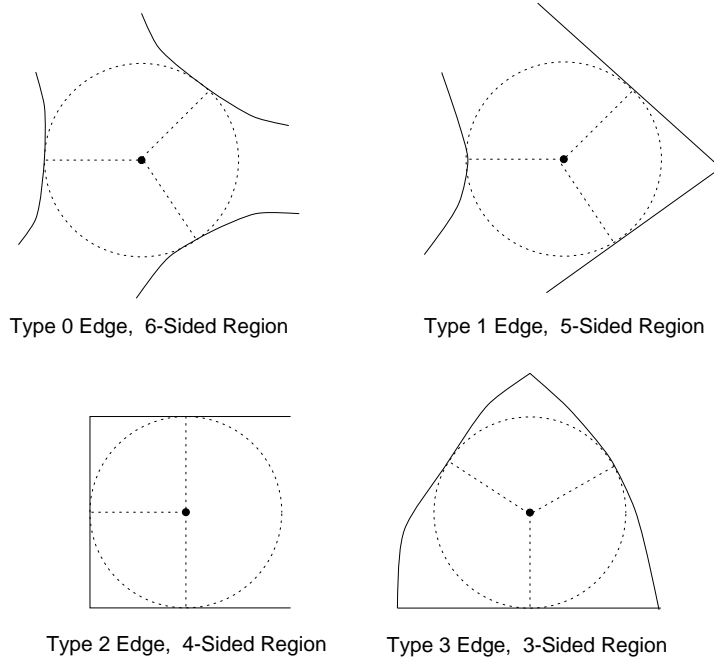


FIG. 3.12. *Major MAT Edge Topologies in Cross Section*

the 2D algorithm. With all these complexities duly noted, the 3D algorithm is overall as follows:

1. From a 3D Delaunay triangulation, determine the topology of the MAT as well as the approximate location of its faces, edges and vertices.
2. Classify the Delaunay tetrahedra, and refine the classification. Determine the adjacency graph of the MAT.
3. Isolate and classify the MAT corners.
4. Isolate and classify the MAT edges.
5. Prepare the remaining MAT faces for 2D meshing.
6. Formulate the compatibility equations and mesh the domain.

Edge Classification. In general, an inscribed maximal sphere centered on an edge point of the MAT touches the domain boundary at three distinct faces. Depending on whether the touched faces are adjacent, four cases arise; Figure 3.12. We see that the MAT edge can be thought of as the spine of a polygonal tube that has between 3 and 6 sides. In consequence, such a region can be meshed by extruding the midpoint subdivision mesh patterns the 2D meshing algorithm would use for the cross sections.

Vertex Classification. In general, an inscribed maximal sphere centered at a MAT vertex touches the boundary at four distinct points. Each

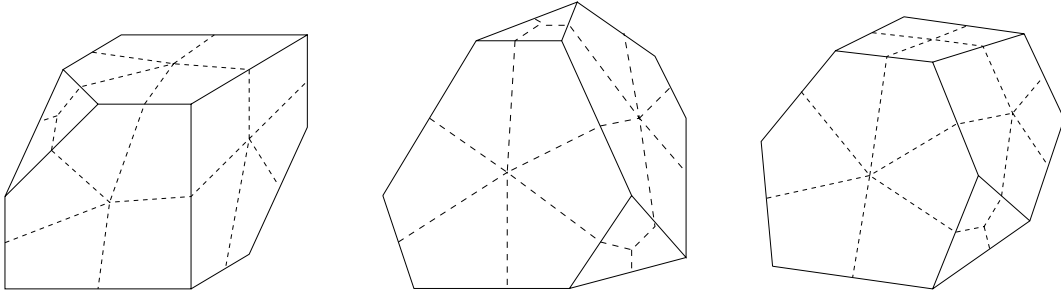


FIG. 3.13. *Some MAT Vertex Topologies and Meshing Patterns*

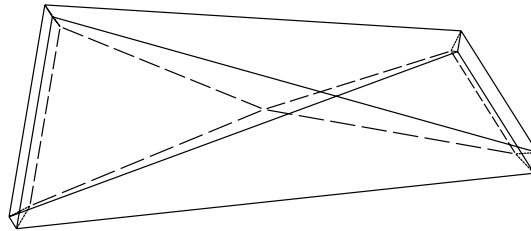


FIG. 3.14. *Truncated Tetrahedral Domain and Medial Axis Faces (C. Armstrong).*

triple of touchings begins an incident edge, and the vertex classification primarily depends on the type of the incident edges. The classification of the adjacent edges defines a subdomain around the common vertex that is topologically characterized by a polyhedron. There are over 20 different vertex polyhedra, and some cases are shown in Figure 3.13. The meshing patterns are also shown.

Face Meshes. Many MAT faces are eliminated from further consideration when edges and vertices have been isolated into vertex polyhedra and polyhedral tubes. The remaining MAT faces are meshed as follows. Compute the approximate MAT of a face, based on shortest distance within the face, and subdivide the face with the 2D algorithm. Extrude the mesh of the face.

The integer equations governing the mesh of each face must be combined with compatibility equations for adjacent edges and vertices. Furthermore, the equations for edge patterns, vertex patterns, and edge/vertex adjacency compatibility are added. The resulting integer programming problem determines the final mesh.

Figure 3.14 shows a simple polyhedron obtained by cutting two opposite edges of a tetrahedron. The central medial-axis faces, adjacent in the center, are two triangles that are at an angle of 90 degrees to each other. The collection of MAT edge-, vertex-, and face-subdomains are shown in Figure 3.15, along with the final composite mesh. Figure 3.16 shows the

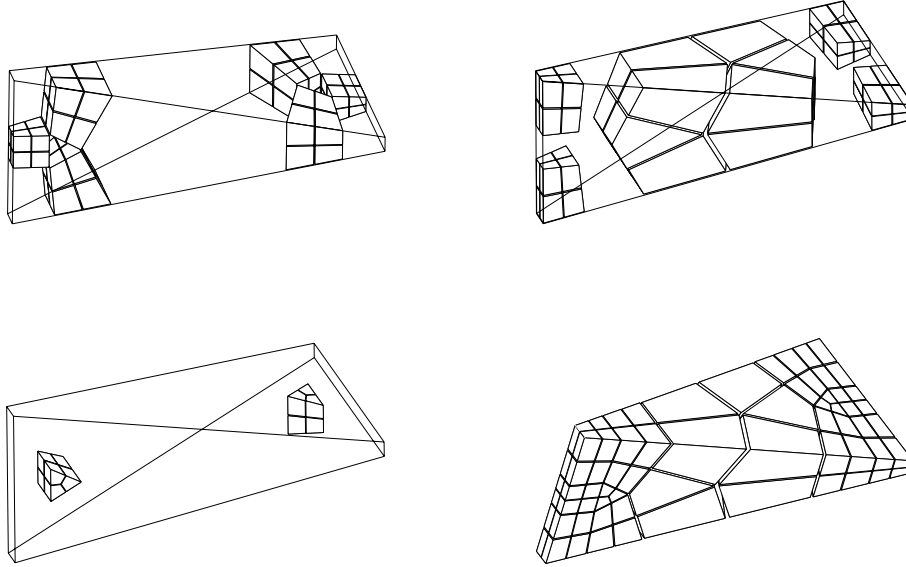


FIG. 3.15. *Meshed Subdomains Belonging to Edges, Vertices and Faces of the MAT, Along With the Final Mesh (C. Armstrong).*

same procedure applied to a more complex object. The mesh is highly structured and follows the object boundary closely. These are desirable attributes for an accurate finite element analysis.

4. Domain Composition. Recall the CSG paradigm in which a 3D domain is constructed using regularized Boolean operations, and the shape primitives are box, cylinder, cone, sphere, and torus. For simple domains, consisting only of one shape primitive, many physical problems are best formulated in a natural coordinate system that simplifies solving the problem. For example, Cartesian coordinates are appropriate for a box domain, whereas cylindrical coordinates are better suited for cylindrical domains. Just as we compose the shape primitives in the geometric design of the domain of interest, we can compose the physical problems in the same way. This idea has been elaborated by Cox in [7] for finite element and finite difference methods. It has a strong technical relationship to the idea of differencing methods working on overlapping grids, as explained in Henshaw's contribution.

Consider the domain shown in Figure 4.1. We mesh separately the disk and the square, with the mesh nodes shown as circles and squares, respectively, in Figure 4.2. For each subdomain, the finite element problem is formulated separately, using the discretization that is most natural. Along the overlapped boundaries of the subdomains, compatibility is enforced by coupling equations. This requires that we express the nodes on

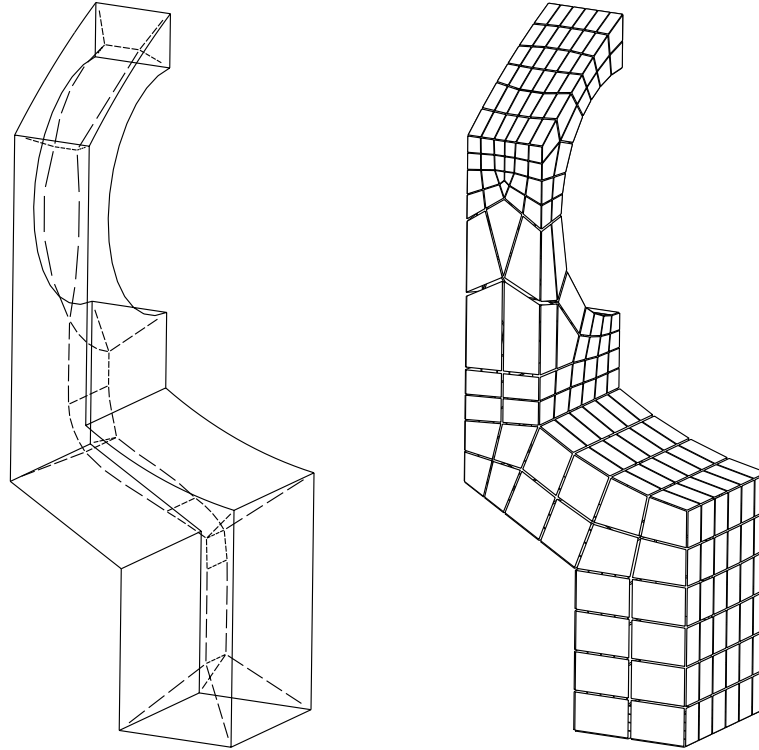


FIG. 3.16. *Complex Object and Mesh (C. Armstrong).*

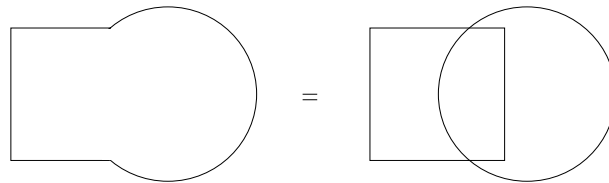


FIG. 4.1. *Domain Composed from Square and Disk*

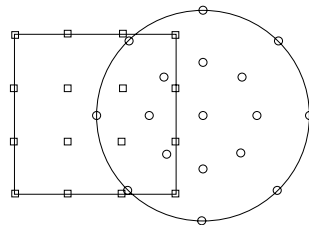


FIG. 4.2. *Meshing a Domain Composed from Square and Disk*

the boundary of one of the subdomains in the coordinates of the other, and vice versa, and require that the solutions on the subdomains agree.

The cross coordinate computations may not be simple to formulate by hand, but the computation is quite straightforward for a geometric modeler. The resulting problems are perhaps more complex to solve numerically, but when sufficiently automated they are obtained quickly, so that the overall design and analysis cycle is sped up.

5. Geometry Compilation to Mesh Representations. Integrating design and analysis is a valuable idea, particularly in light of the present functional barriers between design and analysis that exist in most software systems because of the great differences between the representations used for each task. Work such as the meshing algorithms of Armstrong lower the barriers, but it is clear that the meshing algorithm for 3-dimensional domains involves a considerable amount of detail, and a full implementation is a substantial effort, both in concept as well as in coding. Cox's approach is much simpler, but its scope is narrow because it is based on a narrow range of primitive shapes. We explore next an approach that seeks to combine the strong points of both ideas, combining them with ideas that capture advanced geometric design concepts [15].

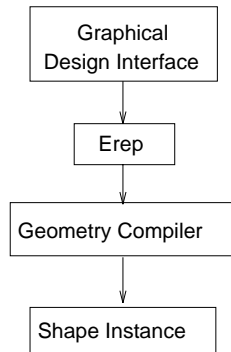
Recent CAD systems have a design interface in which the user composes shapes based on *features*. There is no accepted definition of feature, but it is widely accepted that a feature is a part of a shape that is common, has significance to function or manufacture of the object of which it is part, and can be described in a stereotyped way. For example, in the system Pro/Engineer, features fall broadly into three categories:

1. Volumetric features such as protrusions and cuts that add or subtract predefined or user-defined shapes.
2. Modifying features such as chamfers, rounds and blends that locally alter shape details.
3. Reference structures (datums) that simplify specifying spatial relationships and dimensions using geometric constraints.

In the design process, the user specifies a shape as a hierarchy of features and constraints. By giving values for dimensions and angles, this *generic* design is instantiated and a boundary representation for the instance is created.

Elsewhere, [14,15], we have analyzed this style of design and isolated the shape structures it manipulates. Ignoring the issues that arise in conjunction with instantiating generic design based on constraints, the following shape primitive creation must be considered.

A 2D cross section is drawn, composed of lines, arcs, and spline curves of some type. The cross section is used to define a 3D volume by extrusion (i.e., a sweep along a linear trajectory), revolution (i.e. a sweep along a circular trajectory), or by a sweep along a general space curve. The cross section may be moved as a rigid body in space, or be subjected to

FIG. 5.1. *New CAD System Architecture*

a transformation that alters the shape as a function of the path traversed along the trajectory.

Such shape primitives are then combined using material addition or subtraction operations. They can be implemented as regularized Boolean operations. Furthermore, material can be added or subtracted by stereotyped operations such as chamfering or rounding edges and vertices, shelling (hollowing out a solid volume), or drafting (tapering a generalized cylinder).

This repertoire of design operations is remarkably flexible, for example in mechanical design; [13]. If the design operations are formalized, as proposed in [15], the resulting CAD system architecture looks as shown in Figure 5.1. In this architecture, the design gestures made by the user in the graphical design interface are recorded as a high-level geometry representation, called an *editable representation* (Erep). This shape representation is then translated by a geometry compiler into a specific shape instance, based on constraints and dimensions. The traditional representation of this specific shape would be a boundary representation. But from the user's point of view, the detail representation is quite irrelevant as long as the operations the user wants to do are supported. This suggests translating to other representations, and, in particular, to *analysis representations* in which the shape instance is represented by a finite element mesh. We elaborate on the geometric aspects of this possibility, without regard for convenient ways to specify associated analysis attribute data that further specifies the physical problem and boundary conditions.

The analysis of most engineering parts and assemblies abstracts away a number of shape details of the geometric model. As the user defines features, he or she can tag each one as either essential or inessential for analysis purposes. Furthermore, the high-level Erep feature representation could include rules that define under what conditions the feature should be removed in the analysis. Such conditions would depend on either predefined

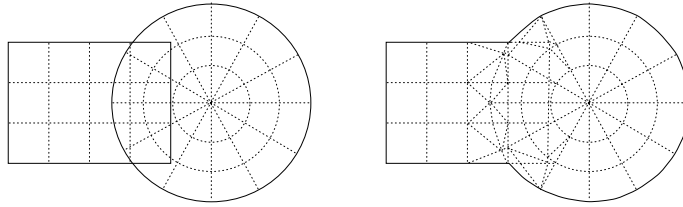
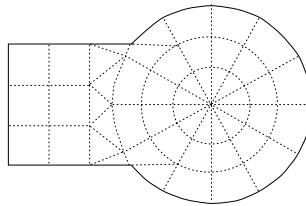
values or on values that would be determined as part of an adaptive analysis process [26].

In many problems, the dimensionality of parts of the domain is reduced. For example, a circular hole may be replaced with its axis. It would be difficult to automate dimensional reduction of features based on the detailed, low-level Brep. In contrast, it would be simple in the Erep: We associate with a feature *simplification rules*. For instance, a hole with a diameter-to-length ratio less than a certain value can be reduced to its axis. Extrusions or sweeps representing beams or plates can be similarly simplified.

In mechanical design optimization, one repeats a cycle that begins with a design or redesign step, and is followed by one or more analysis steps. The results of the analysis steps are used to reconsider design next time round. The traditional implementation of this loop requires translating the Brep that describes the shape design into a mesh representation, interpreting the results of the analysis manually, and repeating the design step. Note that the output of the analysis is not suitable to alter the geometry automatically, mainly because the association of elements with the geometry is weak. It cannot be strengthened when working with a detailed boundary representation. By deriving the analysis representation from the Erep instead, the association between feature and elements is direct, and it becomes possible to process the analysis results to recommend automatically feature modifications and additions. For example, if an area of high stress is along a concave edge that borders two features, we could deduce automatically that the addition of a fillet feature is advisable. So, there are considerable advantages to be realized when approaching the mesh generation problem with a compilation paradigm.

Compiling to a mesh representation is not much different from compiling to ordinary boundary representation. However, different mesh operations arise than are familiar from, say adaptive mesh generation. We proceed as follows: Primitives generated from cross sections are first meshed in cross section using a suitable 2D meshing algorithm. Instead of sweeping the cross section to obtain a volume, we sweep the mesh itself, dividing the polygonal tubes so generated into elements. Just as sweeps along trajectories of varying curvature require special computations to detect self-intersections of the swept boundary, swept meshes must be similarly analyzed and possible self-intersections must be resolved. Boolean operations on meshes can be implemented as described in [9].

In the case of material addition, two meshes are combined. Instead of tolerating overlapping meshes, the overlapped regions are made compatible. This could be done by a refinement, followed by a smoothing step. See Figure 5.2 for a 2D illustration. In the case of cuts, the elements fully overlapped must be removed. Partially overlapped elements are cut and must be *healed*. In both cases the resulting mesh will be denser in and near overlapped regions. This may be desirable from physical considerations,

FIG. 5.2. *Union of Two Meshes by Subdivision*FIG. 5.3. *Mesh Coarsening Applied to Mesh of Figure 5.2*

but where it is undesirable, a *mesh-coarsening* operation must be applied that lowers the mesh density selectively; see also Figure 5.3. This is in a sense the inverse operation of mesh refinement.

Modifying operations such as chamfers and rounds are best implemented as a form of cut, in which the newly interpolated surface cuts through the mesh near the edges and vertices involved. Other modifying operations are analogous.

Acknowledgements. I thank Cecil Armstrong for providing several figures and draft papers.

REFERENCES

- [1] C. Armstrong. Abstraction and meshing of 3d stress analysis models. Progress Report, SERC Directorate, Great Britain, 1991.
- [2] C. Armstrong, T. Tam, D. Robinson, R. McKeag, and M. Price. 2d finite element mesh generation by medial axis subdivision. *Advances in Engr. Software*, 13:313–323, 1991.
- [3] H. Blum. A transformation for extracting new descriptors of shape. In W. Whaten-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, Cambridge, MA, 1967.
- [4] P. Brunet and I. Navazo. Solid representation and operation using extended octrees. *ACM Transactions on Graphics*, 9:170–197, 1990.
- [5] C.-S. Chiang. *The Euclidean Distance Transform*. PhD thesis, Purdue University, Computer Science, 1992.
- [6] C.-S. Chiang, C. M. Hoffmann, and R. E. Lynch. How to compute offsets without self-intersection. In *Proc SPIE Conf Curves and Surfaces in Computer Vision and Graphics*, Volume 1610, pages 76–87. Intl Society for Optical Engineering, 1991.

- [7] J. Cox. *Domain Composition Methods for Combining Geometric and Continuum Field Models*. PhD thesis, Purdue University, Dept. of Mechanical Engineering, 1991.
- [8] P.-E. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14:227–248, 1980.
- [9] V. Ferrucci and A. Paoluzzi. Sweeping and boundary evaluation for multidimensional polyhedra. Technical Report RR 02-90, Università di Roma La Sapienza, Dept. of Informatics and Systems, 1990.
- [10] C. M. Hoffmann. *Geometric and Solid Modeling*. Morgan Kaufmann, San Mateo, Cal., 1989.
- [11] C. M. Hoffmann. A dimensionality paradigm for surface interrogation. *CAGD*, 7:517–532, 1990.
- [12] C. M. Hoffmann. Computer vision, descriptive geometry, and classical mechanics. In B. Falcidieno and I. Herman, editors, *Proc. Eurographics Workshop on Computer Graphics and Mathematics*, Eurographics Series, pages 229–244. Springer Verlag, 1992.
- [13] C. M. Hoffmann. Modeling the DARPA engine in pro/engineer. In *DARPA Workshop Manufacturing, Engineering, Design, Automation*, pages 631–639, Stanford, 1992.
- [14] C. M. Hoffmann. On the semantics of generative geometry representations. In *19th ASME Design Automation Conference*, 1993.
- [15] C. M. Hoffmann and R. Juan. Erep, a editable, high-level representation for geometric design and analysis. In P. Wilson, M. Wozny, and M. Pratt, editors, *Geometric and Product Modeling*. North Holland, 1993.
- [16] C. M. Hoffmann and G. Vaněček. On alternate solid representations and their uses. In *Proc. DARPA Manufacturing Workshop*, Salt Lake City, Utah, 1991.
- [17] A. Kaufman. Modeling in volume graphics. In B. Falcidieno and T. Kunii, editors, *Modeling in Computer Graphics*, pages 441–454. Springer Verlag, 1993.
- [18] D. T. Lee. Medial axis transformation of a planar shape. *IEEE Trans. Pattern Anal. and Mach. Intelligence*, PAMI-4:363–369, 1982.
- [19] M. Mäntylä. *An Introduction to Solid Modeling*. Computer Science Press, 1988.
- [20] E. Müller and J. Krames. *Die Zyklographie*. Franz Deuticke, Leipzig und Wien, 1929.
- [21] B. Naylor. Binary space partitioning trees as an alternative representation of polytopes. *Computer Aided Design*, 22, 1990.
- [22] N. Patrikalakis and H. Gürsoy. Shape interrogation by medial axis transform. Technical Report Memo 90-2, MIT, Ocean Engr. Design Lab, 1990.
- [23] F. Preparata. The medial axis of a simple polygon. In *Proc. 6th Symp. Mathematical Foundations of Comp. Sci.*, pages 443–450, 1977.
- [24] A. Requicha and H. Voelcker. Constructive solid geometry. Technical Report Tech. Memo 25, University of Rochester, Production Automation Project, 1977.
- [25] H. J. Samet. *Applications of Spatial Data structures: Computer Graphics, Image Processing, and GIS*. Addison-Wesley, Redding, MA, 1989.
- [26] M. Shephard. Finite element modeling within an integrated geometric modeling environment: Part ii – attribute specification, domain differences, and indirect element types. *Engineering with Computers*, 2:72–85, 1985.
- [27] V. Srinivasan and L. Nackman. Voronoi diagram of multiply connected polygonal domains. *IBM Journal of Research and Development*, 31:373–381, 1987.
- [28] V. Srinivasan, L. Nackman, J. Tang, and S. Meshkat. Automatic mesh generation using the symmetric axis transformation of polygonal domains. Technical Report RC 16132, IBM Yorktown Heights, 1990.
- [29] T. Tam, M. Price, C. Armstrong, and R. McKeag. Computing the critical points on the medial axis of a using a Delaunay point triangulation algorithm. *IEEE PAMI*, 1991. to appear.
- [30] J. Thompson, Z. Warsi, and W. Mastin. *Numerical Grid Generation*. North Holland, 1985.

- [31] K. Weiler. *Topological Structures for Geometric Modeling*. PhD thesis, Rensselaer Polytechnic University, Dept. of Comp. and Syst. Engineering, 1986.
- [32] X. Yu, J. A. Goldak, and L. Dong. Constructing 3D discrete medial axis. In *Proc. ACM Symp. Solid Modeling Found. and CAD/CAM Applic.*, pages 481–492, Austin, 1991.