56 SOLID MODELING

Christoph M. Hoffmann

INTRODUCTION

The objective of solid modeling is to represent, manipulate, and reason about the three-dimensional shape of solid physical objects, by computer.

Solid modeling is an application-oriented field that has a tradition of implementing systems and algorithms. Major applications include manufacturing, computer vision, graphics, and virtual reality. Technically, the field draws on diverse sources including numerical analysis, symbolic algebraic computation, approximation theory, point set topology, algebraic geometry, and computational geometry.

First, the major representations of solids are reviewed in Section 56.1. They include constructive solid geometry, boundary representation, spatial subdivision, medial surface representations, and procedural representations. Then, major layers of abstraction in a typical solid modeling system are characterized in Section 56.2. The lowest level of abstraction comprises a substratum of basic service algorithms. At an intermediate level of abstraction there are algorithms for larger, more conceptual operations. Finally, a yet higher level of abstraction presents to the user a functional view that is typically targeted toward solid design.

Solid design paradigms work with form features and constraints. Often, they define classes of shape instances, and venture into territory that has yet to be plumbed mathematically and computationally. Concurrently, there is also a shift in the system architecture toward modularized confederations of plug-compatible functional components. We explore these trends lightly in Section 56.3.

Open problems are gathered in Section 56.4.

56.1 MAJOR REPRESENTATION SCHEMATA

GLOSSARY

Solid representation: Any representation allowing a deterministic, algorithmic point membership test.

Constructive solid geometry (CSG): The solid is represented as union, intersection, and difference of primitive solids.

Boundary representation (Brep): The solid surface is represented as a quilt of vertices, edges, and faces.

Spatial subdivision: The solid is decomposed into a set of nonintersecting primitive volumes.

0-8493-8524-5/97/$0.00+$.50

Medial surface transformation: Closure of the locus of centers of maximal inscribed spheres, and a function giving the minimum distance to the solid boundary. Usually called the ***MAT*** for "medial axis transformation."

Procedural representation: The solid is described by a scripting language or a notational schema that must be evaluated.

A solid representation must allow the unambiguous, algorithmic determination of point membership: given any point $p = (x, y, z)$, there must be an algorithm that determines whether the point is inside, outside, or on the surface of the solid. Moreover, restrictions are placed on the topology of the solid and its embedding, excluding, for example, fractal solids.

These restrictions are eminently reasonable. Increasingly, however, solid modeling systems depart from this strict notion of solid and permit representing a mixture of solids, surfaces, curves, and points. The additional geometric structures are useful for certain design processes, for interfacing with applications such as meshing solid volumes, and for abstracting solid features, to name a few.

56.1.1 CONSTRUCTIVE SOLID GEOMETRY

GLOSSARY

Primitive solids: Traditionally block, sphere, cylinder, cone, and torus. More general primitives are possible.

Sweep: Volume covered by sweeping a solid or a closed contour in space.

Extrusion: Sweep along a straight line segment.

Revolution: Circular sweep.

Regularized Boolean operation: The closure of the interior of a set-theoretic union, intersection, or difference.

Algebraic halfspace: Points such that $f(x, y, z) \leq 0$ where f is an irreducible polynomial.

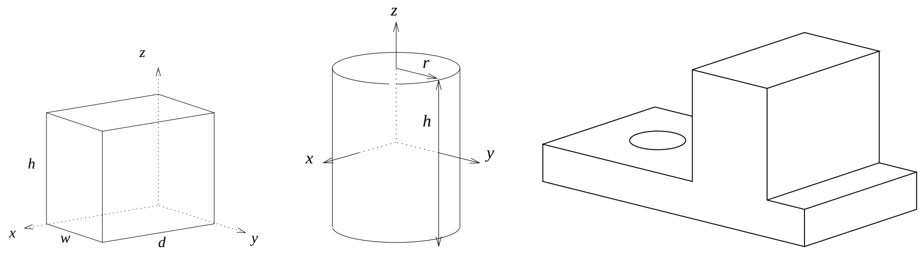
Irreducible polynomial: Polynomial that cannot be factored over the complex numbers.

Classical Constructive Solid Geometry (CSG) represents a solid as a set-theoretic Boolean expression of *primitive* solid objects, of a simpler structure. Both the surface and the interior of the final solid are thereby defined, albeit implicitly. The CSG representation is valid if the primitives are valid. A solid's surface is closed and orientable and encloses a volume. The traditional CSG primitives are block, sphere, cylinder, cone, and torus.

A solid is represented as an algebraic expression that uses rigid motions and regularized set operations. The traditional operations are regularized union, intersection, and difference. A regularized set operation is obtained by taking the closure of the interior of the set-theoretic result. The effect is to obtain solids that do not contain lower-dimensional parts, such as interior (or dangling exterior) faces, edges, and vertices.

Each solid has a default coordinate system that can be changed with a rigid body transformation. A Boolean operation identifies the two coordinate systems of the solids to be combined and makes it the default coordinate system of the resulting solid.

FIGURE 56.1.1
Left and middle: CSG primitives `block`(w, d, h) *and* `cylinder`(r, h) *with default coordinate systems. Right: T-bracket as union of two blocks minus a cylinder.*



As an example, consider Figure 56.1.1. Using the coordinate system conventions shown, the CSG representation of the bracket is the expression

$$\begin{aligned}&\texttt{block}(8,3,1) \cup^{*} \texttt{move}(\texttt{block}(2,2.5,3),(0,4.5,1))\\ &\qquad -^{*} \texttt{move}(\texttt{cylinder}(0.75,1),(1.5,1.5,-0.5))\end{aligned}$$

where the * indicates a regularized operation. (See also Figure 37.4.1.) ref!

The basic operations one wishes to perform on CSG representations are classifying points, curves, and surfaces with respect to a solid; detecting redundancies in the representation; and approximating CSG objects systematically.

More general primitives are obtained by considering the volume covered by sweeping a solid along a space curve, or sweeping a planar contour bounding an area. Defining a sweep is delicate, requiring many parameters to be exactly defined, but simple cases are widely used. They are extrusion, i.e., sweep along a straight line; and revolution, i.e., a sweep about an axis. The evaluation of general sweeps can be done by a number of methods. An even more general set of primitives is algebraic halfspaces, point sets defined by

$$P = \{(x, y, z) \in \mathbb{R}^3 \mid f(x, y, z) \leq 0\},$$

where $f(x, y, z)$ is an irreducible polynomial in x, y, and z.

More general operations are obtained by using nonregularizing Boolean operations or by defining a nonstandard semantics for Boolean operations on surfaces and curves.

56.1.2 BOUNDARY REPRESENTATION

In boundary representation (Brep) the solid surface is represented as a quilt of faces, edges, and vertices. A distinction is drawn between the topological entities, vertex, edge, and face, related to each other by incidence and adjacency, and the geometric location and shape of these entities. See also Figure 56.1.2. For example, when polyhedra are represented, the faces are polygons described geometrically by a face equation plus a description of the polygon boundary. Geometrically, the entities in a Brep are not permitted to intersect anywhere except in edges and vertices that are explicitly represented in the topology data structure. In addition to the classification operations mentioned for CSG, Boolean union, intersection, and difference operations are usually implemented for Brep systems. Both regularized and nonregularizing Boolean operations may occur.

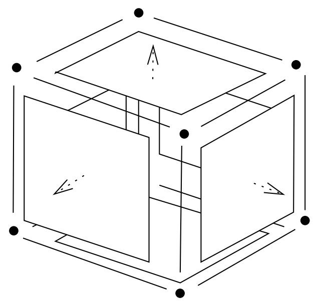


FIGURE 56.1.2
Topological entities of a box. Adjacency and incidence are recorded in Brep. Dotted arrows indicate face orientation.

Different Brep schemata appear in the literature, divided into two major families. One family restricts the solid surfaces to oriented manifolds. Here, every edge is incident to two faces, and every vertex is the apex of a single cone of incident edges and faces. The second family of Brep schemata allows oriented nonmanifolds in which edges are adjacent to an even number of faces. When these faces are ordered radially around the common edge, consecutive face pairs alternatingly bound solid interior and exterior. See Figure 56.1.3 for examples.

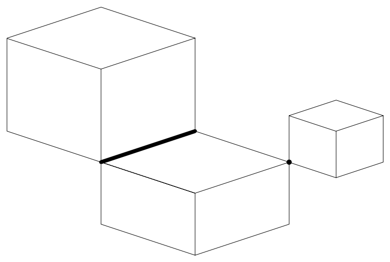


FIGURE 56.1.3
A nonmanifold solid without dangling or interior faces, edges, and vertices; the nonmanifold edges and vertices are drawn with a thicker pen.

More general nonmanifold Breps are used in systems that combine surface modeling with solid modeling. In such representation schemata, a solid may have interior (two-sided) faces, dangling edges, and so on. The current trend is to incorporate surface modeling capabilities into solid modelers.

The topology may be restricted in other ways. For instance, the interior of a face may be required to be homeomorphic to a disk, and edges required to have two distinct vertices. In that case, the Brep of a cylinder would have four faces, two planar and two curved. This may be desirable because of the geometric surface representation, or may be intended to simplify the algorithms operating on solids.

56.1.3 SPATIAL SUBDIVISION REPRESENTATIONS

GLOSSARY

Boundary conforming subdivision: Spatial subdivision of a solid that represents the boundary of the solid exactly.

Boundary approximating subdivision: Spatial subdivision that represents the boundary of the solid only approximately.

Regular subdivision: A subdivision whose cells are congruent. Grids are regular subdivisions.

Irregular subdivision: A subdivision with noncongruent cells.

Octree: Recursive selective subdivision of a cuboid volume into eight subcuboids.

Binary space partition (BSP) tree: Recursive irregular subdivision of space, traditionally by halfplanes. See also Section 37.5. ref!

Spatial subdivision decomposes a solid into cells, each with a simple topological structure and often also with a simple geometric structure. Subdivision representations are divided into *boundary conforming* and *boundary approximating*.

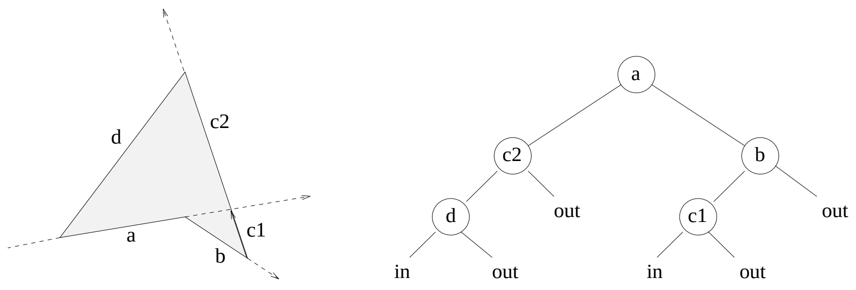
Important boundary conforming subdivision schemata are *meshes* and the *BSP tree*. Mesh representations are used in finite element analysis, a method for solving continuous physical problems. The mesh elements can be geometric tetrahedra, hexahedra, or other simple polyhedra, or they can be deformations of topological polyhedra so that curved boundaries can be approximated exactly. See Sections 24.4–5. ref!

Binary space partition trees are recursive subdivisions of 3-space. Each interior node of the tree separates space into two disjoint point sets. In the simplest case, the root denotes a separator plane. All points of $\mathbb{R}^3$ below or on the plane are represented by one subtree, all points above the plane are represented by the other subtree. The two point sets are recursively subdivided by halfplanes at the subtree nodes. The leaves of the tree represent cells that are labeled IN or OUT. The (half) planes are usually face planes of a polyhedron, and the union of all cells labeled IN is the polyhedron. For an example in $\mathbb{R}^2$ see Figure 56.1.4. Note that algebraic halfspaces can be used as separators, so that curved solids can be represented exactly.

Boundary approximating representations are *grids* and *octrees*. In grids, space is subdivided in conformity with a coordinate system. For Cartesian coordinates, the division is into hexahedra whose sides are parallel to the coordinate planes. In cylindrical coordinate systems, the division is into concentric sectors, and so on. The grids may be regular or adaptive, and may be used to solve continuous physical problems by differencing schemes. Rectilinear grids that are geometrically deformed can be boundary-conforming. Otherwise, they approximate curved boundaries.

An octree divides a cube into eight subcubes. Each subcube may be further subdivided recursively. Cubes and their subdivision cubes are labeled white, black, or grey. A grey cube is one that has been subdivided and contains both white and black subcubes. A subcube is black if it is inside the solid to be represented, white if it is outside. Quadtrees, the two-dimensional analogue of octrees, are used in many geographical information systems. See Figure 37.5.1. ref!

FIGURE 56.1.4
A polygon and a BSP tree representing it.



56.1.4 MEDIAL SURFACE REPRESENTATIONS

GLOSSARY

MAT: Medial axis transform, the two-dimensional version of the medial surface representation. Some authors use "medial axis transform" regardless of the dimension of the domain.

Maximal inscribed disk: Disk inscribed in a domain and not properly contained in another inscribed disk.

Medial axis and medial surface can unambiguously represent two-dimensional domains and three-dimensional solids, respectively. The representations are not widely used for this purpose at this time; more frequently they are used for shape recognition (see Section 50.4). However, as explained below, some sophisticated meshing algorithms are based on the medial axis and the medial surface. ref!

The medial axis of a two-dimensional domain is defined as the closure of the locus of centers of disks inscribed within the domain. A disk is maximal if no other disk properly contains it. An example is shown in Figure 56.1.5 along with some maximal disks.

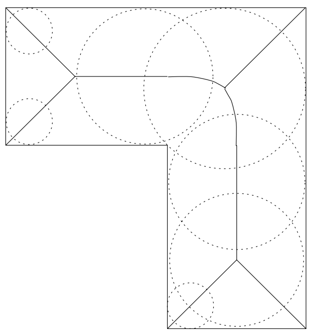


FIGURE 56.1.5
L-shaped domain and associated medial axis. Some maximal inscribed circles contributing to the medial axis are also shown.

The medial surface of a solid is the closure of the locus of centers of maximal inscribed spheres. When we know the radius (the limit radius in case of closure points) of the corresponding sphere for each point on the medial surface, then an unambiguous solid representation is obtained that is sometimes called the medial axis transform (MAT). The MAT has a number of intriguing mathematical properties. For example, by enlarging the radius values by a constant, the MAT of a dilatation of the solid is obtained.

Originally, solid modeling has investigated the MAT for the purpose of constructing shell solids (obtained by subtracting a small inset), for organizing finitie element meshing algorithms, and for recognizing form features. More recently, the role of the MAT in surface reconstruction has begun to impact solid modeling; see Section 30. Surface reconstruction arises in solid modeling for its application in ref! reverse engineering where a model is to be constructed from a physical object by an automated measuring strategy.

56.1.5 PROCEDURAL REPRESENTATIONS

Procedural solid representations fall into two families: script language representations that have a strong programming language character, and descriptive representations evaluated by a program or system.

The PADL system used FORTRAN as script language to specify solids. CSG expressions and directives were embedded into the Fortran program. The solid was evaluated into an internal format. Alpha_1 originally used Lisp as script language and evaluated the solid so described into a boundary representation. Subsequently, a direct manipulation interface was added to the system. The recent SGDL system uses Scheme as script language, evaluating it into an internal proprietary data format. Since such script languages are based on a general programming language, the solid evaluation can be highly complex and may include any computation. Unless the evaluated solid is represented in one of the other representation schemata, it is in general not possible to reason about solids using the procedural representation directly.

Descriptive representations, including the Erep notation are data representations by nature. Their procedural nature derives from the need to evaluate and instantiate parameters, based on (computed) geometric relationship and, in many cases, geometric constraints. Once the parameters are determined, the shape is evaluated in steps, where the major steps typically correspond to form features. Usually, an entire family of solids can be so described and instance solids are obtained by valuating parameters and dimensional constraints. A semantic characterization of the family remains largely an open problem, as discussed later.

56.1.6 CONVERSION BETWEEN REPRESENTATIONS

Most solid modeling systems use Brep. Conversion from CSG to Brep is well understood and is implemented as regularized Boolean operations on Brep solids. An extensive literature addresses these complex algorithms.

The conversion from Brep to CSG is not completely understood. In the polyhedral case, the conversion is essentially the same as the conversion from Brep to BSP tree. Pure CSG solids, using the PADL primitives, can also be converted. Conversion involving higher degree surfaces is largely open.

Some progress has been made by Naylor and Rogers in the case of Bézier curves and B-splines (for definitions, see Section 52.1). Roughly speaking, a coarse BSP ref!

tree is constructed that encloses sections of the curve in convex polygonal regions. On demand, the tree can be extended dynamically, thereby refining the enclosing regions. In this way, points may be classified efficiently with respect to the curve to a required resolution.

There are several algorithms for converting from CSG or Brep to the MAT. Some are based on geometric principles, some on a Delaunay triangulation of an approximated boundary, and some on a grid subdivision of ambient space. Because simple boundary geometry elements can produce very complicated curves and surface elements in the MAT, approximation approaches are favored in practice. The conversion from MAT to Brep has been addressed by Vermeer and later on by Amenta. Note that a polyhedral MAT produces a solid boundary that can contain spherical, conical, and cylindrical elements.

The conversion from CSG or Brep to mesh representations is a partially solved problem when the conversion is done for finite element analysis or other numerical treatment of continuum problems. In that context, the problem is not a geometric problem alone: the quality of the subdivision must also be judged by nongeometric criteria that come from the nature of the physical problem and the numerical algorithm used to solve it. Many approaches are based on octree subdivision, on Delaunay triangulation, and on MAT computations.

The conversion relationships are summarized in Table 56.1.1.

TABLE 56.1.1 Representation conversion.

CONVERSION	REMARKS
CSG $\rightarrow$ Brep	Many methods, e.g., [Chi88, Hof89, Män88]. Active research seeks better tradeoff between speed, accuracy, and geometric coverage.
Brep $\rightarrow$ CSG	Largely open. Polyhedral case similar to BSP tree construction [Hof93b]; quadric cases treated in [Sha91, SV93]. See also [NR95] for parametric case.
Brep, CSG $\rightarrow$ MAT	[CHL91] uses grid approximation, [SAR95] uses Delaunay approximation of domain.
MAT $\rightarrow$ Brep	[Ver94] converts polyhedral MAT.
Brep, CSG $\rightarrow$ spatial subdivision	Many approaches; see, e.g., [Hof95, TWM85, SERB99]. Active research seeks improved techniques.

GEOMETRIC COVERAGE

The range and geometric representation of solid surfaces is referred to as *geometric coverage*. Polyhedral modeling restricts to planes. Classical CSG allows only planes, cones, cylinders, spheres, and tori. Experimental modelers have been built allowing arbitrary algebraic halfspaces. SGDL uses implicit algebraic surfaces of degree up to 4.

Most commercial and many research modelers use B-splines (uniform or nonuniform, nonrational or rational) or Bézier surfaces. The properties and algorithmic treatment of these surfaces is studied by computer-aided geometric design. See Chapter 52 of this Handbook, as well as the monographs and surveys [Far88, Hos92, ref!

HL93].

Subdivision surfaces have also been proposed but have, thus far, not gained wide acceptance in solid modeling. There are many connections between certain kinds of subdivision curves and surfaces and certain classes of spline curves and surfaces. See also 53. ref!

SPATIAL RELATIONSHIPS

In many applications one would like to understand spatial relationships. Some of the solid representations reviewed have been considered for this purpose. For instance, the MAT has been used to guide meshing algorithms globally and some attempts have been made to devise simplifications for isolating specific features of a shape. Attempts have been made to define suitable simplifications and variations of the MAT; e.g., [FLM03].

Shape simplification is fundamental for many tasks, including in collision detection reviewed in chapter 35. When a shape is offset by a large distance, smaller ref! features tend to disappear, hence offsetting, a close relative of the MAT, can be used to explore shape simplification; [BDG97]. Other approaches have constructed hierarchical representations in which shape is approximated by a hierarchy of simple bounding volumes that at the tree root enclose the entire shape, and in the interior refine the shape estimate by alternatingly subtracting and adding smaller bounding volumes; e.g., [GLM96, KGL$^+$98]. Such trees of bounding volumes have similarity with CSG trees.

56.2 LEVELS OF ABSTRACTION

GLOSSARY

Substratum: Basic computational primitives of a solid modeler, such as incidence tests, vector arithmetic, etc.

Algorithmic infrastructure: Major algorithms implementing conceptual operations, such as surface intersection, edge blending, etc.

Graphical user interface (GUI): Visual presentation of the functionality of the system.

Application procedural interface (API): Presentation of system functionality in terms of methods and routines that can be included in user programs.

Substratum problem: Unreliability of logical decisions based on floating-point computations.

Large software systems should be structured into layers of abstraction. Doing so simplifies the implementation effort because the higher levels of abstraction can be compactly programmed in terms of the functionality of the lower levels. Thereby, the complexity of the system is reduced. A solid modeling system spans several levels of abstraction:

1. On the lowest level, there is the substratum of arithmetic and symbolic computations that are used as primitives by the algorithmic infrastructure. This level contains point and vector manipulation routines, incidence tests, and so on.
2. Next, there is an intermediate level comprising the algorithmic infrastructure. This level implements the conceptual operations available in the user interface, as well as a wide range of auxiliary tools needed by these operations. There is often an application programming interface available with which programs can be written that use the algorithmic infrastructure of the modeling system.
3. A graphical user interface (GUI) presents to the user a view of the functional capabilities of the system. Interaction with the GUI exercises these functions, for instance, for solid design. Tools for editing and archiving solids are included.

Ideally, the levels of abstraction should be kept separate, with the higher levels leveraging the functionality of the lower levels. However, this separation is fundamentally limited by the interaction of numeric and symbolic computation.

56.2.1 THE SUBSTRATUM

The substratum consists of many low-level computations and tests; for example, vector computations, simple incidence tests, and computations for ordering points along a simple curve in space. Ideally, these operations create an abstract machine whose functionality simplifies the algorithms at the intermediate level of abstraction. But it turns out that this abstract machine is unreliable in a subtle way when implemented using floating-point arithmetic. Exact arithmetic would remedy this unreliability, but is held by many to be unacceptably inefficient when dealing with solids that have curved boundaries. See Section 40.4. Problems include input accuracy.

ref!

To illustrate how inexact arithmetic at the substratum level can impact the geometric computation, consider modeling polyhedral solids, the simplest possible situation for solid modeling. All computational decisions that arise in the course of a regularized Boolean operation on polyhedra can be reduced to determining the sign of 4×4 determinants. Geometrically, this is a test of whether a point is above, on, or below a plane. When the determinant's value is nearly zero, floating-point evaluation will decide based on a tolerance. But the decision is unreliable because logically equivalent tests may arise as different determinants in the course of the algorithm: some of the determinants could have small, others large values, thus necessitating different tolerances to arrve at consistent decisions. This gives an opportunity for the algorithm to build inconsistent data structures and fail. The problems are magnified when dealing with curved solids.

Recent academic solid modeling systems adopt exact arithmetic either outright, or use exact arithmetic on demand. In the latter approach, an error bound is evaluated along with the predicate on whose value a logical decision depends. If the decision is unambiguous based on floating-point arithmetic, no further action is taken. Otherwise, an exact evaluation is done. If an exact evaluation is to be made, the input is understood to be exact as given, and the predicate must be evaluated from the input data without using intermediate, possibly inaccurate, data. The assumption of exact input data is problematic for Brep solids. Unless the input solid is very simple, or it was computed using exact arithmetic, it is an open problem how to interpret the data such that a valid solid is obtained. For a deeper evaluation of the problem, and for some approaches to solving it, see Chapter 40. ref!

56.2.2 ALGORITHMIC INFRASTRUCTURE

Algorithmic infrastructure is a prominent research subject in solid modeling. Among the many questions addressed is the development of efficient and robust algorithms for carrying out the geometric computations that arise in solid modeling. The problems include point/solid classification, computing the intersection of two solids, determining the intersection of two surfaces, interpolating smooth surfaces to eliminate sharp edges on solids, and many more. See the reference section for a sampling of the literature.

Recent academic work considers structuring *application procedural interfaces* (API's) that encapsulate the functional capabilities of solid modelers so they can be used in other programs; [ABC+00]. Such API's play a prominent role in applications because they allow building on existing software functionality and constructing different abstraction hierarchies than the one implemented by a full-service solid modeling system. The work attempts to give a system-independent specification of basic API functionality for solid modeling.

An important consideration when devising infrastructure is that the algorithms are often used by other programs, whether or not there is an API. Therefore, they must be ultra-reliable and in most cases must not require user intervention for exceptional situations.

The major geometric computations implemented at the infrastructure level have to balance the conflicting goals of efficiency, accuracy, and robustness. For this reason, many operations continue to be researched in efforts to seek new perceived optima. Moreover, new variants of surface representations continue to be devised that necessitate different approaches. Some of the major operations on which research continues are the following.

Surface intersection. Given two bounded areas of two surfaces, determine all intersection curve components. A major difficulty of the problem is to identify correctly all components of the intersection, including isolated points and singularities. Since this computation is done in $\mathbb{R}^3$, classical algebraic geometry is of limited help. The other difficulty is to address properly the substratum unreliabilities.

Surface intersection remains a key problem with continuing attempts at balancing efficiency, accuracy and stability of the algorithms.

Offsetting. Given a surface, its offset is the set of all points that have fixed minimum distance from the surface. Offsets can have self-intersections that must be culled, and there is a technical relationship between offsetting and forming the MAT. Namely, when offsetting a curve or surface by a fixed distance, the self-intersections must lie on the medial axis. Offsetting is used to determine certain blending surfaces, and is also used in the solid operation of *shelling* that creates thin-walled solids.

Blending. Given two intersecting surfaces, a third surface is interpolated between them to smooth the intersection edge. A simple example is shown in Figure 56.2.1. A locally convex blend surface is often called a *round*, and a locally concave one a *fillet*. The blend surface in Figure 56.2.1 is a fillet.

Blending has been considered almost since the beginning of solid modeling, and some intuitive and interesting techniques have been developed over the years. For example, consider blending two primary surfaces f and g. Roll a ball of fixed radius r along the intersection such that it maintains contact

FIGURE 56.2.1
Left: two cylinders intersecting in a closed edge. Right: edge blended with a constant-radius, rolling-ball blend; the bounding curves of the blend are shown.

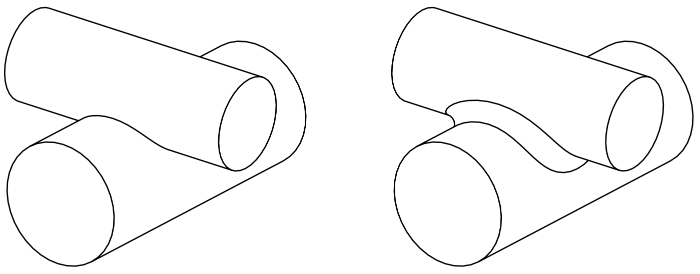


FIGURE 56.2.2
Global blend interference [Bra97]: The round of the front edge overlaps with the fillet of the cylinder edge on top (left). Without further action, the two blends do not connect, leaving a gap in the surface. The solution shown in the middle modifies the front round. Other possibilities include modifying the fillet or inserting a separate blend in the overlap region (right).

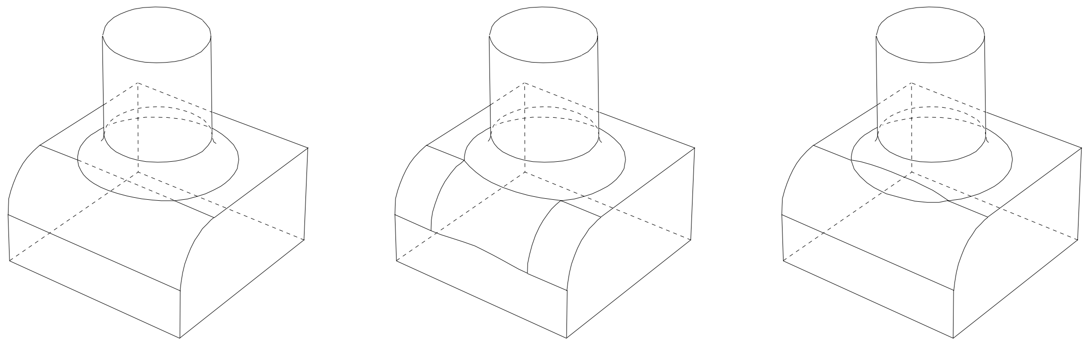
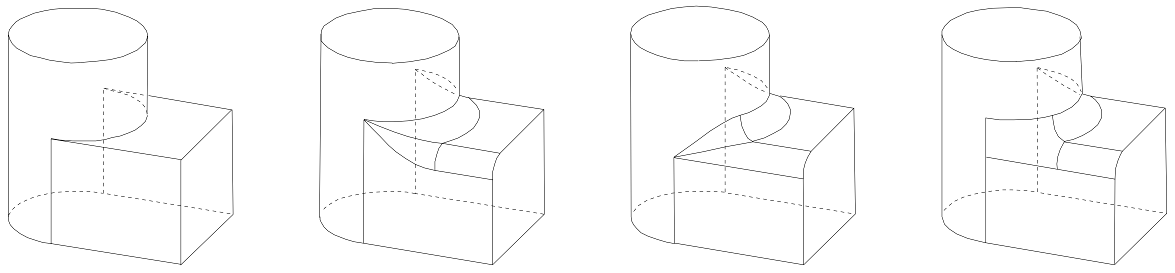


FIGURE 56.2.3
Global blend interference [Bra97]: At ending vertex, the round and the fillet must be merged into a compatible structure. Several solutions are illustrated.



with both f and g. Then the surface of the volume swept by the ball can be used as a blending surface, suitably trimmed. Note that the center of the ball lies on the intersection of the offsets, by r, of both f and g. In more complicated schemes the radius of the ball is varied along the intersection.

A less well-understood issue for blending solids arises from the global problem of how to devise the contact curves and blending surfaces, so that the surfaces connect properly at adjacent faces, behave correctly at vertices, and so on. Figure 56.2.2 shows the problem of overlapping blends. The fillet and round constructed separately do not meet in the region of overlap. The problem is that then there is no closed surface defining the blended solid. A resolution could modify the round, or the fillet, or could insert a separate surface in the overlap region after suitably cutting back both primary blends.

When the primary surfaces meet at a vertex tangentially, blending surfaces must "dissipate." Figure 56.2.3 shows several ways how to dissipate round and fillet at the end vertices. The examples are from [Bra97] and point out the dimensions of the global problem.

Deformations. Given a solid body, deform it locally or globally. The deformation could be required to obey constraints such as preserving volume or optimizing physical constraints. For example, we could deform the basic shape of a ship hull to minimize drag in fluids of various viscosities.

Shelling. Given a solid, hollow out the volume so that a thin-wall solid shape remains whose outer surface is part of the boundary of the input solid. The wall thickness is a parameter of the operation. Variations include designating parts of the solid surface as "open." For instance, taking a solid cylinder and designating both flat end faces as open the operation creates a hollow tube of the same outside diameter. Conceptually, the operation subtracts an inset of the solid, obtained by shrinking the original solid, an offset operation.

56.2.3 USER INTERFACES

Ultimately, the functional capabilities of a solid modeling system have to be presented to a user, typically through a *graphical user interface* (GUI). It would be a mistake to dismiss GUI design as a simple exercise. If the GUI merely presents the functionality of the infrastructure literally, an opportunity for operational leveraging has been lost. Instead, the GUI should conceptualize the functionalities an application needs. As in programming language design, this conceptual view can be convenient or inconvenient for a particular application. Research on GUI's therefore is largely done with a particular application area in mind.

For example, in mechanical engineering product design, an important aspect of the GUI might be to allow the user to specify the shape conveniently and precisely. This might be accomplished using geometric constraints and constraints of length, radius, and angle. In GUI's for virtual environment definition and navigation, on the other hand, approximate constraints and direct manipulation interfaces would be better.

56.3 FEATURES AND CONSTRAINTS

GLOSSARY

Form feature: Any stereotypical shape detail that has application significance.

Geometric constraint: Prescribed distance, angle, collinearity, concentricity, etc.

Generic design: Solid design with constraints and parameters without regard to specific values.

Design instance: Resulting solid after substituting specific values for parameters and constraints.

Parametric constraint solving: Solving a system of nonlinear equations that has a fixed triangular structure.

Variational constraint solving: Solving a system of nonlinear simultaneous equations.

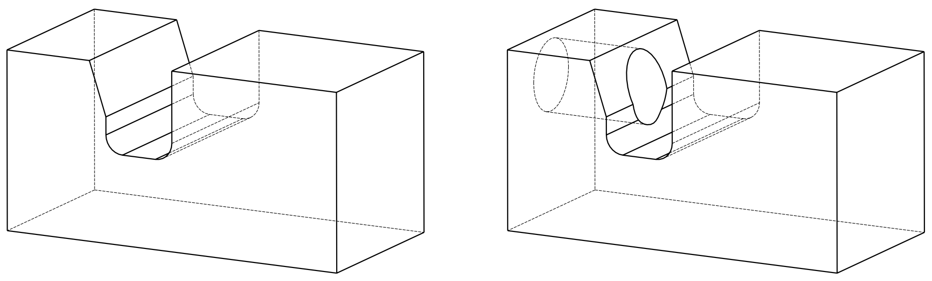
In solid modeling, two design paradigms have become standard for manufacturing applications, *feature-based design* and *constraint-based design.* The new paradigms expose a need to reconsider solid representations at a different level of abstraction. The representations reviewed before are for individual, specific solids. However, we need to represent entire *classes* of solids, comprising a generic design. Roughly speaking, solids in a class are built structurally in the same way, from complex shape primitives, and are instantiated subject to constraints that interrelate specific shape elements and parameters. How these families should be defined precisely, how each generic design should be represented, and how designs should be edited are all important research issues of considerable depth.

56.3.1 FEATURE-BASED DESIGN

Feature-based design is usually understood to mean designing with shape elements such as slots, holes, pockets, etc., that have significance to manufacturing applications relating to function, manufacturing process, performance, cost, and so on. Focusing on shape primarily, we can conceptualize solid design in terms of three classes of features: generative, modifying, and referencing features. A feature is added to an existing design using attachment attributes and placement conditions. Subsequent editing may change both types of attachment information.

As an example, consider the solid shown to the right in Figure 56.3.1. A hole was added to the design on the left, and this could be specified by giving the diameter of the hole, placing its cross section, a circle, on the side face, and requiring that the hole extend to the next face. Should the slot at which the hole ends be moved or altered by subsequent editing, then the hole would automatically be adjusted to the required extent.

FIGURE 56.3.1
Left: solid block with a profiled slot. Right: After adding a hole with the attribute "through next face," an edited solid is obtained. If the slot is moved later, the hole will adjust automatically.

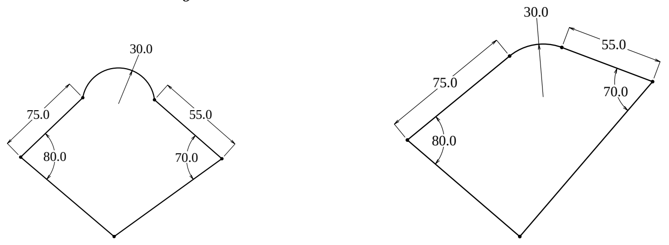


56.3.2 CONSTRAINT-BASED DESIGN

Constraint-based design refers to specifying shape with the help of constraints, when placing features or when defining shape parameters. For instance, assume that we are to design a cross section for use in defining a solid of revolution. A rough topological sketch is prepared (Figure 56.3.2, left), annotated with constraints, and instantiated to a sketch that satisfies the constraints exactly (Figure 56.3.2, right). Auxiliary geometric structures can be added, such as an axis of rotation. There is an extensive literature on constraint solving, from a variety of perspectives.

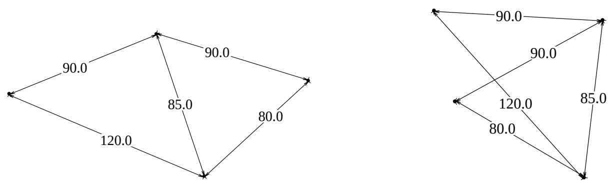
Most solid modeling systems use both features and constraints in the design interface. Often, the constraints on cross sections and other two-dimensional structures are unordered, but the constraints on three-dimensional geometry are usually considered in a fixed sequence. Solving systems of unordered constraints is sometimes referred to as *variational constraint solving*. Mathematically, it is equivalent to solving a system of nonlinear simultaneous equations. Solving constraints in a fixed sequence is also known as *parametric constraint solving*. The latter is equivalent to solving a system of nonlinear equations that has a fixed, triangular structure where each equation introduces a new variable.

FIGURE 56.3.2
Geometric constraint solving. Input to the constraint solver shown on the left. Here, the arc should be tangent to the adjacent segments, and the two other segments should be perpendicular. Output of the constraint solver shown on the right.



A well-constrained geometric constraint problem corresponds naturally to a system of nonlinear algebraic equations with a finite set of solutions. In general, there will be several solutions of a single, well-constrained geometric problem. An example is shown in Figure 56.3.3. This raises the interesting question of exactly how a constraint solver should select one of those solutions efficiently, and why.

FIGURE 56.3.3
The well-constrained geometric problem of placing 4 *points by* 5 *distances has two distinct solutions.*



From symbolic computation we know that there are algorithms to convert a nontriangular system of nonlinear equations into a triangular system. The distinction between parametric and variational constraint solving is therefore artificial in theory. However, full-scale triangularization of systems of nonlinear equations is not tractable in many cases, so the distinction is relevant in practice. Moreover, a predetermined sequential evaluation of constraints is simple to implement and can be interfaced easily with conditional constraint evaluation, thereby increasing the expressive power of the constraint system without raising new semantic issues. For these reasons, many developers of solid modeling systems leverage core modeling capabilities by such (simple) extensions.

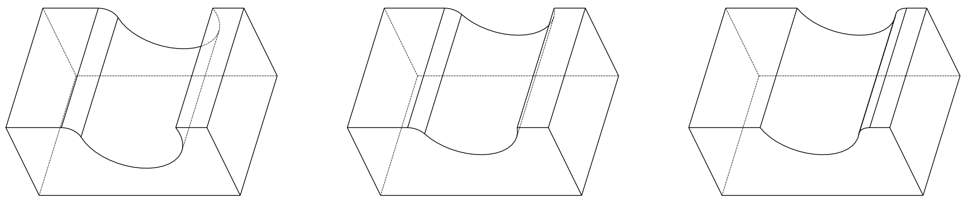
Spatial constraint solving is very much more demanding than planar constraint solving. In the planar case, simple (simultaneous) subsystems can be identified and isolated using straightforward graph algorithms, and result in practically important solvers. Furthermore, in the planar case, there are not many such subsystems needed. In contrast, no simple simultaneous spatial subsystems exist. When lines are allowed as geomtric primitives, then the systems become very much harder and there are many such subsystems even when restricting to only 5 or six geometric elements. The number of basic cases number in the hundreds; [GHY02]. This structural barrier seems to preclude the emergence of truly spatial constraint solvers, and with it, of spatial design paradigms. In practice, CAD systems fudge the issue by building interrelated planar constraint problems which are variational in each plane but follow a clear, parametric sequence for elaborating the spatial relationship between the various planar problems.

56.3.3 SEMANTIC PROBLEMS

When constraints and parameters are used in solid design, a ***generic design*** is obtained. Generic designs are instantiated by constraint values, and may be edited by changing the constraint values, the constraint schema, and the feature attributes. A design so edited can then be automatically re-instantiated by the solid modeler. A central difficulty in implementing this scenario, however, is that the generic design is usually defined visually on the basis of a particular instance, and when the design changes, the instance geometry is no longer present. Thus, visually identified instance structures must be suitably described, so that re-instantiation can be carried out correctly.

As an example, consider the solid shown in Figure 56.3.4, left. It was constructed as follows. First, a rectangle was drawn and extruded into a block. On the front face of the block, a circle was drawn as a profile of a slot across the top of the block. Then, an edge was visually identified for rounding. This design is edited by altering the position of the circular slot profile. The edge to be rounded is not an explicit design entity, however. Hence, the edge has to be described implicitly, perhaps by the intersection of the circle and the top edge of the face on which the circle has been drawn. This description does not distinguish between the two straight edges of the slot, however, so additional information has to be used. Such information would have to allow a consistent identification under all possible constraint values, and is called the ***persistent naming*** problem.

FIGURE 56.3.4
A block with a slot and round on the left edge is shown left. After editing, in this case decreasing the depth of the slot, re-instantiation should produce the solid shown in the middle. However, some systems may re-instantiate as shown to the right, an error.



There has been a small stream of academic work on this topic, although it is of intense interest in applications. In particular, the formalization of the design information has profound implications on system architectures because it formalizes, in effect, the information flow between functional components. Whenever such formalization seeks independence from the specific implementation of the system components, system modularization is facilitated. Ultimately, this will accelerate the current trend of decomposing solid modeling systems into standardized components that can function interchangeably and can be combined in a variety of ways.

56.4 OPEN PROBLEMS

Most major problems in solid modeling contain a conceptualization aspect. That is, a precise, technical formulation of the problem commits to a specific conceptualization of the larger context that may be contentious. For example, consider the following technical problem. *Given an implicit algebraic surface S and a distance d, find the "offset" of S by d.* Assuming a precise definition of offset, and a restriction to irreducible algebraic surfaces S, the problem statement ignores the fact that a solid model is not bounded by a single, implicit surface, and that implicit surfaces of high algebraic degree may cause severe computational problems when used in a solid modeler.

CONSTRAINT SOLVING

Geometric constraint solvers trade efficiency for generality. Some very interesting techniques have been developed that are fast but not very general, for planar problems. They could be extended in various ways without substantially impacting on efficiency. Such extensions, for constraint solving in the plane, include the incorporation of parametric curve segments as geometric elements, more general constraint configurations, relations among distances, and angles.

Spatial geometric constraint solving poses a number of open problems, including determining whether a constraint problem is generically well-constrained. The problem of how many lines can be found at prescribed distance from four fixed points has been solved, one of the sequential construction problems for lines. Other construction problems for lines are not completely solved. The smaller simultaneous problems involving points and planes have been solved. Most simultaneous problems involving lines require numerical treatment, however, and are not well understood.

FEATURES

Manufacturing applications need cogent definitions of features to accelerate design. Such definitions ought to be in terms of generic mechanisms of form and of function. Also needed are mapping algorithms interrelating different feature schemata.

A set of features, say those conceptualizing machining a shape from stock, is called a *design view*. In manufacturing applications there are many views, including machining, tolerancing, design view, etc. Work has begun to address the problem of altering a design in one view with an automatic update of the other views. To do so requires reasoning about shape and is a hard problem. Some approaches have been based on subdividing the shape by superimposing all feature boundaries, and then tracking how the subdivision is affected by changes to one of the features.

SEMANTICS OF CONSTRAINT-BASED DESIGN

A solid shape design in terms of constraints can be changed simply by changing constraint values. To date, all such changes have been specified in terms of the procedures and algorithms that effect the change. What is needed is an abstract definition of shape change under such constraint changes to obtain a semantic definition of generic design and constraint-based editing. Such a definition must be visually intuitive.

MODEL RECTIFICATION

Because of the substratum problem, Brep data structures can be invalid in the sense that the geometric description does not agree fully with the topological description. For instance, there may be small cracks between adjacent faces, the edge between two adjacent faces may not be where the curve description would place it, and so on. This has motivated work to "heal" the defective surface by closing cracks, eliminating overlaps, and so on. Some approaches sew up cracks with smaller faces, and in the case of polyhedra with triangles. Optimal healing is known to be NP-hard.

An intuitive idea is to assign a thickness to faces, edges and vertices, and enlarge the thickness so that the surface closes up. The difficulty is to work out what happens when nonadjacent faces merge into adjacent ones. The natural geometric enlargement creates mathematically difficult surfaces; for instance, the offset surface of an ellipsoid increases the algebraic degree by a factor of 4. So, an interval based approach has also been proposed in which there is no closed-form description of the enlarged geometric elements.

56.5 SOURCES AND RELATED MATERIAL

FURTHER READING

Monographs on solid modeling. Monographs and surveys provide an excellent entry into solid modeling. Major monographs on solid modeling are [Chi88, Hof89, Män88]. Books on the related field of CAGD (computer-aided geometric design) may also contain material on solid modeling but concentrate primarily on curve and surface design and manipulation. Surface interrogation from a solid modeling point of view is explored in [Hos92, PM02].

Solid representations and conversion. There is a large and diverse literature on representations and representation conversion. Classical work focused primarily on the semantic foundations of CSG and Brep and includes [Req77, Wei86]. Maintaining Brep and CSG simultaneously has been explored in [RS00]. The mesh and octree representations are treated in [BN90, Hof95, Sam89a, Sam89b, TWM85], including the associated conversion problems. The medial axis representation of solids, and how to compute with it, are considered in [Hof92, SAR95, Ver94]. Implicit algebraic halfspaces as solid primitives are discussed in [BDL$^+$91]. The conversion between boundary representation and CSG can be considered a generalization of the binary space partition tree and is explored in [Hof93b, Nay90, NR95, Sha91, SV93]. Curve and surface representations, and their manipulation, are the subject of [Far88, Hos92, HL93, PM02]. More specialized treatment of offsets and sweeps is found in [BL90, CHL91]. Procedural script language representations are discussed in [Bro82, oU94, Sys01] for PADL, Alpha_1, and SGDL. Data representations that neutrally describe form features and constraints are developed in [HJ92].

ref! *Substratum, infrastructure and user interfaces.* The substratum robustness issue is presented in greater depth in Chapter 40; [SI89, For97] explores the use of exact arithmetic in polyhedral modeling. Manocha and Keyser work with exact arithmetic for curved solids; [KKM99a, KKM99b]. A recent survey is found in [Hof01].

Infrastructure work is traditionally quite extensive. Surface intersection is treated in [Hoh92]; this thesis contains an excellent summary of previous work. A recent monograph on the subject is [PM02]. Global solid operations are considered in [BW89, For95, PS95, RSB95]; local solid operations are discussed in [HH87, Pet92]. Much work has been done in blending. The local problem ios often addressed in the context of CAGD, and the monographs on that subject contain much material. The global blending problem is treated extensively in [Bra97]. ref! Work in symbolic algebraic computation (Chapter 32) has foundational importance, for instance in regard to converting between surface representations. Some of the applications of symbolic computation are explored in [BCK88, Cho87, Hof90].

Features and constraints. Neither topic is new, so there is a sizable literature on both. The confluence of the two issues in recent solid modeling systems, however, is new. It raises a number of questions that have only recently been articulated and addressed. [SHL92, KRU94] discuss feature work. Constraints are the subject of [BFH$^+$95, HV94, Kra92]. The confluence of the two strands and some of the implications are discussed in [HJ92]. Some of the technical issues that must be addressed are explained in [Hof93a, CH94], and there is more work emerging on this subject. In particular, Shapiro and Raghothama propose several criteria for defining a family of solids; [RS02, RS98].

RELATED CHAPTERS

ref! Chapter 24: Triangulations and mesh generation
ref! Chapter 32: Computational real algebraic geometry
ref! Chapter 37: Geometric Intersection
ref! Chapter 40: Robust geometric computation
ref! Chapter 48: Computer graphics
ref! Chapter 50: Pattern recognition
ref! Chapter 52: Splines and geometric modeling

REFERENCES

[ABC$^+$00] C. Armstrong, A. Bowyer, S. Cameron, J. Corney, G. Jared, R. Martin, A. Middleditch, M. Sabin, and J. Salmon. *Djinn, a geometric interface for solid modelling.* Information Geometers, Ltd., Winchester, UK, 2000.

[BCK88] B. Buchberger, G. Collins, and B. Kutzler. Algebraic methods for geometric reasoning. *Annual Reviews in Computer Science*, 3:85–120, 1988.

[BDG97] G. Barequet, M. Dickerson, and M. Goodrich. Voronoi diagrams for polygon-offset distance functions. In *Workshop on Algorithms and Data Structures*, pages 200–209. Springer LNCS 1272, 1997.

[BDL$^+$91] A. Bowyer, J. H. Davenport, D. A. Lavender, A geometric algebra system. In D. Kapur, editor, *Integration of Symbolic and Numeric Methods.* MIT Press, 1991.

[BFH$^+$95] W. Bouma, I. Fudos, C. Hoffmann, J. Cai, and R. Paige. A geometric constraint solver. *Computer Aided Design*, 1995. to appear.

[BL90] D. Blackmore and M. Leu. A differential equations approach to swept volume. In *Proc. Rensselaer 2nd Intl Conf on Computer-Integrated Manufacturing*, pages 143–149, Troy, NY, 1990.

[BN90] P. Brunet and I. Navazo. Solid representation and operation using extended octrees. *ACM Transactions on Graphics*, 9:170–197, 1990.

[Bra97] I. Braid. Non-local blending of boundary models. *CAD*, 29:89–100, 1997.

[Bro82] C. M. Brown. PADL-2: a technical summary. *IEEE Comput. Graph. and Applic.*, 2:69–84, 1982.

[BW89] M. Bloor and M. Wilson. Generating blending surfaces with partial differential equations. *Computer-Aided Design*, 21:165–171, 1989.

[CH94] X. Chen and C. Hoffmann. Editing feature based design. *to appear in CAD*, 1994.

[Chi88] H. Chiyokura. *Solid Modeling with Designbase.* Addison-Wesley, 1988.

[CHL91] C.-S. Chiang, C. Hoffmann, and R. 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.

[Cho87] C.-S. Chou. *Mechanical Theorem Proving.* D. Reidel Publishing, 1987.

[Far88] G. Farin. *Curves and Surfaces for Computer-Aided Geometric Design.* Academic Press, 1988.

[FLM03] M. Foskey, M. Lin, and D. Manocha. Efficient computation of a simplified medial axis. In *Proc. 8th ACM Symp. on Solid Modeling and Applications*, pages 96–107. ACM Press, 2003.

[For95] M. Forsyth. Shelling and offsetting bodies. In *Proc 3rd ACM Symp on Solid Modeling.* ACM Press, 1995.

[For97] S. Fortune. Polyhedral modeling with multi-precision integer arithmetic. *CAD*, 29:123–133, 1997.

[GHY02] X.-S. Gao, C. Hoffmann, and W.-Q. Yang. Solving spatial basic geometric constraint configurations with locus intersection. In *Proc. 7th ACM Symp. on Solid Modeling and Applications.* ACM Press, 2002.

[GLM96] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: A hierarchical structure for rapid interference detection. *Computer Graphics*, 30(Annual Conference Series):171–180, 1996.

[HH87] C. Hoffmann and J. Hopcroft. The potential method for blending surfaces and corners. In G. Farin, editor, *Geometric Modeling*, pages 347–365. SIAM, 1987.

[HJ92] C. Hoffmann and R. Juan. Erep, an editable, high-level representation for geometric design and analysis. In P. Wilson, M. Wozny, and M. Pratt, editors, *Geometric Modeling for Product Realization*, pages 129–164. North Holland, 1992.

[HL93] J. Hoschek and D. Lasser. *Computer Aided Geometric Design.* A. K. Peters, 1993.

[Hof89] C. Hoffmann. *Geometric and Solid Modeling.* Morgan Kaufmann, 1989.

[Hof90] C. Hoffmann. Algebraic and numerical techniques for offsets and blends. In S. Micchelli M. Gasca, W. Dahmen, editor, *Computations of Curves and Surfaces*, pages 499–528. Kluwer Academic, 1990.

[Hof92] C. Hoffmann. Computer vision, descriptive geometry, and classical mechanics. In B. Falcidieno and I. Herman, editors, *Computer Graphics and Mathematics*, Eurographics Series, pages 229–244. Springer Verlag, 1992.

[Hof93a] C. Hoffmann. On the semantics of generative geometry representations. In *Proc. 19th ASME Design Automation Conference*, pages 411–420, 1993. Vol. 2.

[Hof93b] C. Hoffmann. On the separability problem of real functions and its significance in solid modeling. In *Computational Algebra*, pages 191–204. Marcel Dekker, 1993. Lecture Notes in Pure and Applied Mathematics, 151.

[Hof95] C. Hoffmann. Geometric approaches to mesh generation. In I. Babuska, J. Flaherty, W. Henshaw, J. Hopcroft, J. Oliger, and T. Tezduyar, editors, *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations*. Springer Verlag, 1995.

[Hof01] C. Hoffmann. Robustness in geometric computations. *J of Computing and Info. Sci. in Engr.*, 1:143–155, 2001.

[Hoh92] M. Hohmeyer. *Surface Intersection*. PhD thesis, University of California, Berkeley, Dept. of Computer Science, 1992.

[Hos92] M. Hosaka. *Modeling of Curves and Surfaces in CAD/CAM*. Springer Verlag, New York, 1992.

[HV94] C. Hoffmann and P. Vermeer. Geometric constraint solving in R^2 and R^3. In D. Z. Du and F. Hwang, editors, *Computing in Euclidean Geometry*. World Scientific Publishing, 1994. second edition.

[KGL$^+$98] S. Krishnan, M. Gopi, M. Lin, D. Manocha, and A. Pattekar. Rapid and accurate contact determination between spline models using ShellTrees. *Computer Graphics Forum*, 17(3):??–??, 1998.

[KKM99a] J. Keyser, S. Krishnan, and D. Manocha. Efficient and accurate B-rep generation of low degree sculptured solids using exact arithmetic: I – representations. *CAGD*, 16:841–859, 1999.

[KKM99b] J. Keyser, S. Krishnan, and D. Manocha. Efficient and accurate B-rep generation of low degree sculptured solids using exact arithmetic: II – computation. *CAGD*, 16:861–882, 1999.

[Kra92] G. Kramer. *Solving Geometric Constraint Systems*. MIT Press, 1992.

[KRU94] F.-L. Krause, E. Rieger, and A. Ulbrich. Feature processing as kernel for integrated CAE systems. In *Proc IFIP Intl Conf: Feature Modeling and Recognition in Advanced CAD/CAM Systems Vol II*, pages 693–716, Valciennes, 1994.

[Män88] M. Mäntylä. *An Introduction to Solid Modeling*. Computer Science Press, 1988.

[Nay90] B. Naylor. Binary space partitioning trees as an alternative repressentation of polytopes. *Computer Aided Design*, 22, 1990.

[NR95] B. Naylor and L. Rogers. Constructing binary space partitioning trees from piecewise Bézier curves. In *Graphics Interface '95*, 1995.

[oU94] University of Utah. *Alpha_1 advanced experimental CAD modeling system*, 1994. www.cs.utah.edu/gdc/projects/alpha1/.

[Pet92] J. Peters. Joining smooth patches around a vertex to form a c^k surface. *Computer Aided Geometric Design*, 9:387–411, 1992.

[PM02] N. Patrikalakis and T. Maekawa. *Shape interrogation for computer-aided design and manufacture*. Springer Verlag, 2002.

[PS95] A. Pasko and V. Savchenko. Algebraic sums for deformation of constructive solids. In *Proc 3rd ACM Symp on Solid Modeling*. ACM Press, 1995.

[Req77] A. Requicha. Mathematical models of rigid solids. Technical Report PAP Tech Memo 28, University of Rochester, 1977.

[RS98] S. Raghotama and V. Shapiro. Boundary representation deformation in parametric solid modeling. *ACM Trans on Graphics*, 17:259–286, 1998.

[RS00] S. Raghotama and V. Shapiro. Consistent updates in dual representation systems. *CAD*, 32:463–477, 2000.

[RS02] S. Raghotama and V. Shapiro. Topological framework for part families. In *Proc ACM Symp on Solid Modeling and Applic*, 2002.

[RSB95] A. Rappoport, A. Sheffer, and M. Bercovier. Volume-preserving free-form solids. In *Proc 3rd ACM Symp on Solid Modeling*. ACM Press, 1995.

[Sam89a] H. J. Samet. *Applications of Spatial Data structures: Computer Graphics, Image Processing, and GIS*. Addison–Wesley, 1989.

[Sam89b] H. J. Samet. *Design and analysis of Spatial Data Structures: Quadtrees, Octrees, and other Hierarchical Methods*. Addison–Wesley, 1989.

[SAR95] D. Sheehy, C. Armstrong, and D Robinson. Computing the medial surface of a solid from a domain Delaunay triangulation. In *Proc 3rd ACM Symp on Solid Modeling*, 1995.

[SERB99] A. Sheffer, M. Etzion, A. Rappoport, and M. Bercovier. Hexahedral mesh generation using the embedded voronoi graph. *Engineering with Computers*, 15:248–262, 1999.

[Sha91] V. Shapiro. *Representations of Semialgebraic Sets in Finite Algebras Generated by Space Decompositions*. PhD thesis, Cornell University, Sibley School of Mechanical Engineering, 1991.

[SHL92] J. Shah, D. Hsiao, and J. Leonard. A systematic approach for design-manufacturing feature mapping. In P. Wilson, M. Wozny, and M. Pratt, editors, *Geometric Modeling for Product Realization*, pages 205–222. North Holland, 1992.

[SI89] K. Sugihara and M. Iri. A solid modeling system free from topological inconsistency. *Journal of Information Processing*, 12:380–393, 1989.

[SV93] V. Shapiro and D. Vossler. Separation for boundary to CSG conversion. *ACM Transactions on Graphics*, 12:35–55, 1993.

[Sys01] SGDL Systems. *The SGDL language*, 2001. www.sgdl-sys.com.

[TWM85] J. Thompson, Z. Warsi, and W. Mastin. *Numerical Grid Generation*. North Holland, 1985.

[Ver94] P. Vermeer. *Medial Axis Transform to Boundary Representation Conversion*. PhD thesis, Purdue Univ., 1994. Comp. Sci.

[Wei86] K. Weiler. *Topological Structures for Geometric Modeling*. PhD thesis, Rensselaer Polytechnic Inst., 1986. Comp. and Syst. Engr.