# Solid modeling and beyond

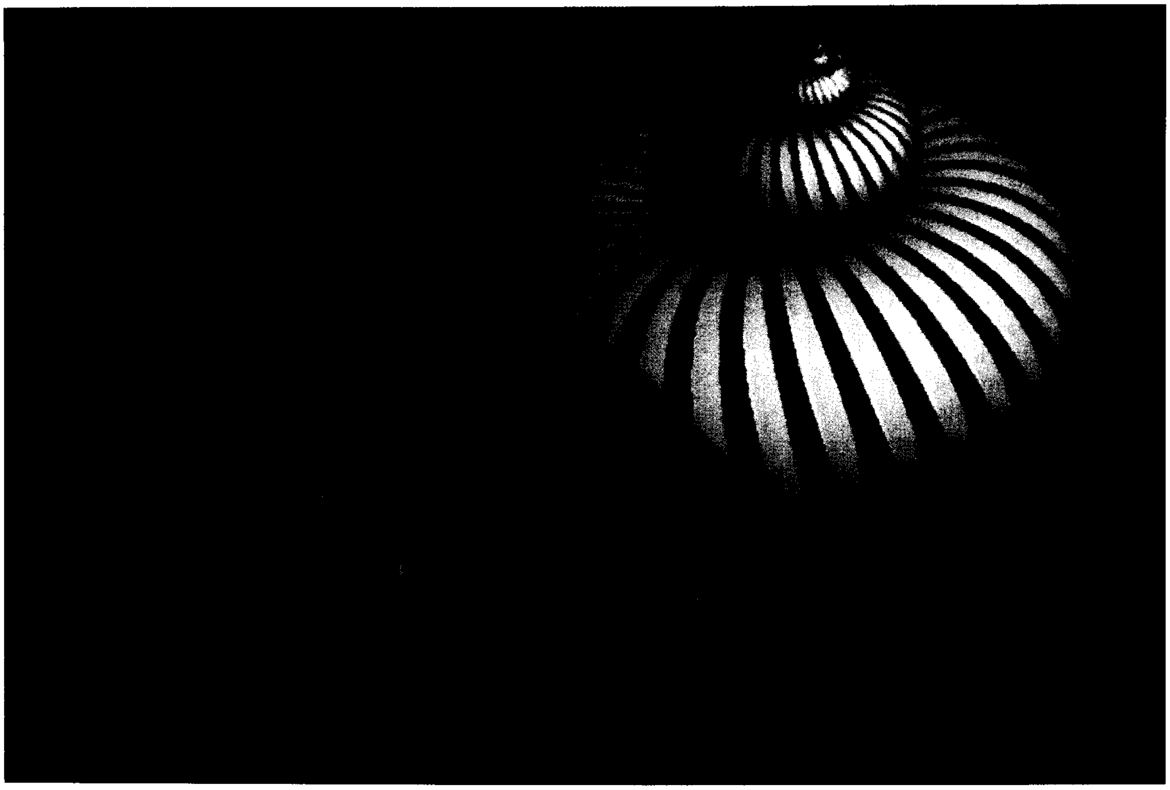
**Article** *in* IEEE engineering in medicine and biology magazine: the quarterly magazine of the Engineering in Medicine & Biology Society · October 1992

DOI: 10.1109/38.156011 · Source: IEEE Xplore



**2 authors**, including:

J. Rossignac
Georgia Institute of Technology



# Solid Modeling and Beyond

Aristides A.G. Requicha
University of Southern California

Jarek R. Rossignac
IBM T.J. Watson Research Center

*We survey the field of solid modeling and assess its strengths and limitations. Our survey covers mathematical foundations, representations, algorithms, applications, user interfaces, and systems.*

A solid model is an unambiguous computer representation of a physical solid object. Research in solid modeling started with a few exploratory efforts in the mid-1960s, but began in earnest in the early 1970s, when several research groups were established in the main industrial nations. Researchers and practitioners recognized that the computer-aided design and manufacturing (CAD/CAM) systems of the time required extensive user intervention to perform seemingly routine tasks. A substantially higher level of automation required that all the geometric information about solid objects be captured in computer representations more powerful than the wireframe models then in vogue. Thus, solid modeling was born. (The history and initial development of the field are summarized in the March 1982 special issue of *IEEE Computer Graphics and Applications*[1] and in two subsequent survey papers.[2,3])

The last decade has seen explosive growth in research and associated publications in a broad area that in this article we call *geometric computation*. Five distinct but overlapping communities have emerged, each with its own primary conference and distinctive technical slant.



0272-17-16/92/0900-0031$03.00 © 1992 IEEE

The theoretical aspects of *design and analysis of geometric algorithms* are covered in the annual Association for Computing Machinery (ACM) Symposium on Computational Geometry. The roots of this research lie in the theory of computation and complexity analysis. (The designation "computational geometry" was first introduced in the late 1960s. Recently, the term has acquired a narrower meaning in the computer science community.)

Work slanted towards *curves and surfaces*, often called computer-aided geometric design, is presented in a biennial conference sponsored primarily by the Society for Industrial and Applied Mathematics (SIAM). Curve and surface modeling has strong ties with numerical analysis and differential geometry.

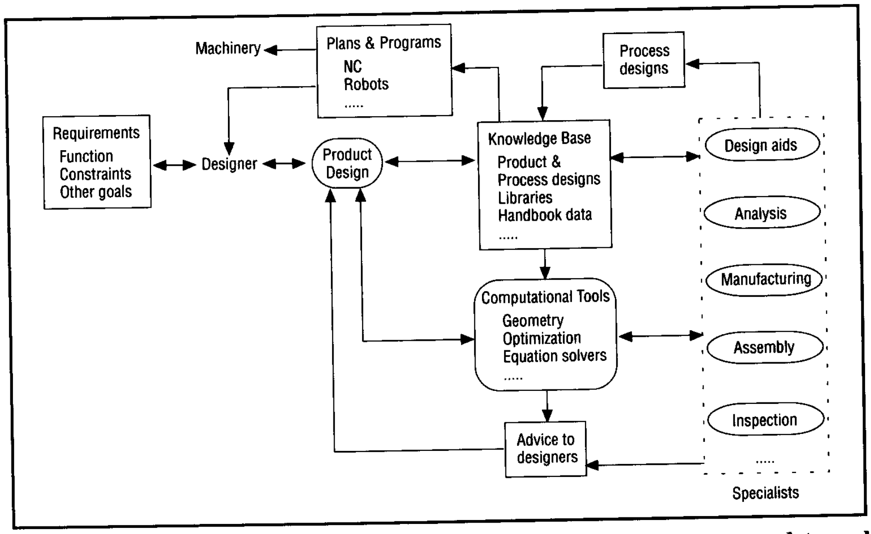



**Figure 1. Engineering environments. Rectangular boxes in the figure represent data, and rounded-corner boxes correspond to computational procedures.**

Research in *solid modeling* and its applications is discussed at the biennial ACM Symposium on Solid Modeling. Solid modeling's mathematical foundations come primarily from topology and algebraic geometry. Its computational aspects blend work on data structures and algorithms from computer science with application considerations from engineering domains, especially design and production of mechanical parts and assemblies.

Spatial reasoning in configuration spaces and other geometric aspects of *robotics* are covered (together with other topics) in the Institute of Electrical and Electronics Enigineers' (IEEE's) annual International Conference on Robotics and Automation. Solid modeling and geometrically oriented robotics have strong intellectual ties, but are studied by separate communities.

*Rendering* of geometric models and issues of user interaction are the focus of much of the work reported at the ACM's annual Siggraph conferences.

Related research sometimes appears at other conferences, such as the Amercian Society of Mechanical Engineers (ASME) Design Theory and Methodology Conference, the International Federation for Information Processing (IFIP) and Eurographics Workshops on Intelligent CAD, the International Conference on Artificial Intelligence in Design, the IEEE Conference on AI Applications, and so on. Taken together, these efforts produce a wealth of rapidly evolving knowledge, which will likely coalesce into a unified field of geometric computation. Much of the recent research surveyed here aims to extend solid modeling in ways that blur the distinctions between subdisciplines.

Solid modeling has a broad range of applications, including architecture and construction, computer graphics and visualization, computer vision, integrated circuits and electronic packaging, and electromechanical CAD/CAM. To establish a specific application context for our survey, we consider a class of systems that we and others are building to support the concurrent design of electromechanical products and processes. We call these systems *engineering environments*, by analogy with programming environments. The ultimate goal is to provide engineers with a set of computational tools for

- designing products,
- assessing their manufacturability, ease of assembly, and other life-cycle characteristics, and
- generating all the information necessary to produce them.

Eventually, we expect such tools to become highly automated and require user input only at the top—the conceptual-design level. Figure 1 illustrates the concept. A designer interacts with a set of product requirements and with a product design subsystem through a suitable human-computer interface. Representations for evolving product designs are stored in a knowledge base. Several specialist programs reason about the product design and generate two types of information: advice to designers on the consequences of design decisions; and process designs, or *plans*, for manufacture and other activities associated with the product's life cycle. Product and process designs are stored in the knowledge base, together with handbook data and other useful information. The plans can be used to drive the robots and numerically controlled machinery that convert raw material into physical products. Observe that the specialists and other computational processes shown in Figure 1 invoke a set of fundamental tools of generic applicability. Solid modeling technology provides many of these tools, as well as facilities for product and, to some extent, process representation.

Engineering environments suggest a high-level structure for the field of solid modeling. We use this structure to organize our article into sections on representations (of products), fundamental algorithms and operations (generic computational tools), applications (specialist algorithms), design tools and user interaction, and systems. In addition, we also discuss the mathematical foundations of the field, and draw some conclusions in a final section.

This article surveys work on solid modeling over the last decade. We did not attempt to be exhaustive because of space limitations, and also because we no longer can follow all the relevant literature. At the request of the editors, we limited the number of references. But we cite several survey papers[1-3] and books[4,5] that contain many more references. For a representative selection of current research in the field, we recommend the proceedings of the last solid modeling symposium.[6]

This survey has several biases. First, it reflects our own view of the field. Second, it focuses on applications in mechanical CAD/CAM. Third, it primarily uses examples drawn from our own research and that of our friends, because that is the work we know best (and also the work most available to us, for practical reasons such as access to data and illustrations). We suspect that some readers will view these biases as virtues, others as defects.

## Mathematical foundations

Semi-algebraic sets were introduced in the geometric modeling literature in the mid-1970s, in a series of reports from the University of Rochester's Production Automation Project. A semi-algebraic set is the result of a finite number of Boolean set operations (union, intersection, difference, complement) applied to half spaces defined by algebraic inequalities, that is, half spaces of the form $H = \{(x, y, z) \mid p(x, y, z) \leq 0\}$, where $p$ is a polynomial function. Bounded and homogeneously three-dimensional (or regular) semi-algebraic sets, called $r$-sets, were proposed as mathematical models for physical solids. (A set is regular if it equals the closure of its topological interior.) $R$-sets are generally accepted as suitable models for solids, although some authors prefer the more restrictive subclass of manifold $r$-sets. Each edge of a manifold solid is shared precisely by two faces, and each vertex has precisely one "cone" of adjacent faces.

The importance of semi-algebraic, but not necessarily regular, sets has been established in many other areas of geometric computation. Three examples follow. The *geometric constraints* associated with mating relationships in a mechanical assembly—for example, prismatic and revolute kinematic joints—define semi-algebraic sets. Potential *contacts* between a moving robot and stationary obstacles define algebraic hypersurfaces in the space of the robot's location and orientation parameters, and give rise to *configuration-space obstacles* that are semi-algebraic. Many elementary geometry theorems can be rephrased in terms of the existence (that is, nonemptiness) of certain semi-algebraic sets, and one can exploit this for (automatic) geometric theorem proving.

Tools from algebraic geometry (for example, resultants and Gröbner bases) have been applied systematically to problems in geometric modeling, robotics, and geometric reasoning in the last 10 years (see Hoffman[4]). They have led to better theoretical understanding, but their practical impact is still unclear, mostly because the associated algorithms tend to be very slow. For example, solids whose faces are subsets of free-form parametric surfaces are semi-algebraic, therefore you can express them as Boolean combinations of half spaces. (Statements to the contrary appear in several papers in the geometric modeling literature.) But the implicitization procedures are slow, the required half spaces often have very high degrees, and general algorithms for automatically computing the appropriate Boolean expressions are unknown.

Many CAD/CAM applications now being developed require a geometric domain that extends beyond solids and encompasses objects that are either dimensionally or materially inhomogeneous. For example, contact regions between solids—important in assembly modeling and planning—are not three-dimensional. Also, many objects important in practice, such as composite-material aircraft parts and semiconductor circuits, consist of several regions with different material properties. Thus, they cannot be represented in a solid modeler as the union of their regions, because the union operator obliterates the "internal boundaries" between the different regions.

You can use nonregular semi-algebraic sets to model dimensionally inhomogeneous objects. But materially inhomogeneous objects must be modeled as sets of sets, called *hypersets* to emphasize that their elements are themselves sets. A semi-algebraic hyperset is a structured entity composed of several regions, much like a complex in algebraic topology. (Do not confuse a hyperset with its underlying space, which is a set defined as the union of the object's regions.) Much of the current work on mathematical foundations, representations, and algorithms addresses the domain of semi-algebraic hypersets. Space partitions—that is, decompositions or subdivisions of Euclidean space into a set of disjoint regions with certain properties (such as connectedness)—play an important role in these studies.

Finally, we see progress in the mathematics of tolerance specifications, which also define sets of sets, called *variational classes*. A variational class encompasses all the solids (that is, $r$-sets) that meet a tolerance specification. Recent work by Boyer and Stewart[7] at the University of Montreal proposes sharp mathematical characterizations for variational classes in terms of regularity in a topology related to the Hausdorff distance between sets.

## Representations

We can group the primary schemes for representing geometric objects into three broad categories: decomposition, boundary, and constructive representations.

Regular decompositions approximate solids by stacks of constant-thickness slices with 2D cross sections, by prismatic columns of square cross sections and parallel axes, or by sets of identical cubical cells called *voxels*. (We do not use "regular" here in the topological sense, but rather to mean uniformly spaced.) Although approximate, these representations have recently gained popularity because their simplicity suits parallelism and hardware implementation (discussed below in the section "Systems"). Column decompositions, also called *ray representations*, correspond to arrays of parallel line segments. They can be computed from other representations by using ray-casting algorithms. Regular decompositions may have a hierarchical structure, as in octree representations. Nonregular decompositions also exist, such as BSP (binary space partition) trees.[8]

Objects can be represented by their topological boundaries. You can represent a solid's boundary by decomposing it into a set of faces, with each face in turn represented by the surface in which it lies and by its bounding edges. Edges are often defined in the 2D parametric space of the surfaces as segments of piecewise polynomial curves. Although a simple enumeration of the solid's faces suffices to unambiguously separate the solid from its complement, most boundary representation schemes store additional information to help in traversing the boundary and in answering topological questions, such as, "How many connected components does the solid have?" This additional information captures connectivity relations between the geometric boundary elements (vertices, edges, and faces).

Boundary data often have to be augmented with neighborhood or orientation information to avoid ambiguities. For example, a circular edge on a spherical surface is the boundary of two complementary faces. You can define a unique face by orienting its edges counterclockwise around the face with respect to the surface normal. Since each edge bounds several faces (two, for manifolds) and must thus be used with different orientations, some representation schemes have multiple entries for each edge. Mäntylä's half edges,[5] Kalay's hybrid edges,[9] and Weiler's edge uses[10] are notable examples.

Ordering relations are stored to facilitate specific traversals of the boundary. Different compromises between storage requirements and traversal performance have produced many variants in the data structures proposed for boundary representations. Examples are the pioneering winged-edge representation developed by Baumgart in 1972,[11] Guibas and Stolfi's quad-edge data structure,[12] and Dobkin and Laszlo's facet-edge data structure,[13] which extends the quad-edge principle to three dimensions.

A *constructive* representation defines an object by providing a recipe for constructing the object through operations on primitives. The best known constructive representation is constructive solid geometry. CSG primitives are themselves solids, and the operations are either rigid motions or regularized Booleans—union, intersection, and difference. Typical CSG representations are rooted, directed, acyclic, binary graphs, where the internal nodes correspond to Boolean operations or rigid-body transformations and where leaves are primitive bounded solids or half spaces. CSG representations are concise, always valid in the $r$-set modeling domain, and easily parameterized and edited. Many solid modeling algorithms work directly on CSG representations through the divide-and-conquer method.

Other constructive representations can be devised by using different sets of operations and primitives. For example, Minkowski operations generate *sweep* representations (discussed further in the section "Fundamental algorithms and operations").

Several boundary representation schemes have been proposed for domains that extend beyond solids. Many of these schemes are based on decomposition into cells or regions. Paoluzzi and his colleagues at the University of Rome in Italy triangulate $n$-dimensional sets and represent the resulting simplices ($n$-dimensional analogs of triangles and tetrahedra) by their vertices.[14] Bieri and Nef,[15] Sobhanpanah,[16] and others also have proposed techniques for representing and manipulating higher dimensional polytopes.

Brisson[17] at Amherst College and Lienhardt[14] at the University Louis Pasteur in France independently extended the facet-edge data structure to higher dimensions. Each cell-tuple in Brisson's structure points to one vertex, one edge, one face, one volume, and so on for higher dimensions. Each cell of dimension greater than zero in a cell-tuple is incident with another cell of immediately lower dimension in the same cell-tuple. Brisson defines the Switch($k$) operator to traverse the data structure. Lienhardt uses $n$-dimensional generalized maps to support analogous operations. The two formulations are equivalent for manifold objects.

Extensions to nonmanifold solids, as well as to inhomogeneous objects or hypersets, also have been addressed. Mäntylä's half edges[5] and Kalay's hybrid-edge[9] data structures cater to dimensionally inhomogeneous objects. Weiler[10] used entities representing incident face-edge and vertex-edge pairs as a basis for his radial-edge data structure, which explicitly captures how faces are ordered around an edge and how edges are ordered around a face. It does not, however, provide direct information on the ordering of cones of faces or of dangling faces and edges incident on a common vertex. This vertex-neighborhood ordering problem is important for a consistent traversal of an object's boundary at nonmanifold vertices. Gursoz, Choi, and Prinz[14] at Carnegie Mellon University addressed the problem.

Selective geometric complexes (introduced elsewhere[18]) provide a general representation for $n$-dimensional collections of disjoint, open, connected sets, called *cells*. (SGC cells need not be homeomorphic to $n$-balls.) SGCs are suitable for representing nonmanifold and inhomogeneous point sets, as well as nonclosed sets and hypersets composed of several regions. SGC representations are decompositions, with each cell represented by its boundary and orientation information. Ordering information, equivalent to Brisson's cell-tuples switches and to Lienhardt's maps, can be added in a very compact form, as a 2D table associated with each element.

Constructive representations for semi-algebraic hypersets composed of disjoint regions are discussed elsewhere.[14] Hypersets are represented by CNRG graphs (short for "constructive nonregularized geometry"), which correspond to expressions involving (nonregularized) Boolean and topological operations on primitive regions. The primitives need not be solids. Unlike SGC cells, CNRG regions can be disconnected, non-open, and dimensionally inhomogeneous.

Solid models also have evolved towards a higher level of representational primitives. Ongoing research efforts on automatic design and manufacturing show a large gap between typical solid modeling primitives (lines, cylinders, cuboids, and so on) and the entities meaningful for design and production, called *features*. No consensus exists on a precise definition for "feature," but many researchers consider features to be solids that can be added to or subtracted from another solid. Additive features include bosses and webs, whereas holes and slots are subtractive features.

Feature-based representations raise delicate issues not fully understood. Suppose, for example, that a boss partially covers a previously defined hole. Is the hole still a valid feature? More generally, how is feature validity defined precisely, and how does it relate to feature interactions?

Mechanical products typically are assemblies of several individual parts. Solid models are being enhanced so as to capture the relationships between mating parts, as well as other information necessary for representing assemblies.[19] This work relates to issues of constraint manipulation, discussed below in the section "Design tools and user interaction." Toward the design end are attempts to devise product models that contain explicit knowledge about function, behavior, and design intent. Still embryonic, these efforts fall largely outside the scope of geometric modeling. Much of the necessary high-level information appears to be associated naturally with regions of the objects, usually called *design features*.

## Fundamental algorithms and operations

Algorithms for constructing representations of solids, and for processing such representations in various applications, build upon a few fundamental procedures that implement geometric operations. A ubiquitous operation, *set membership classification*, segments a candidate set into three subsets inside, outside, or on the boundary of a reference set. For example, by classifying a set of lines with respect to a solid, you can construct ray representations and display objects by ray casting.

The importance of regularized Boolean operations has been recognized since the early times of solid modeling, and virtually all modern modelers support these operations. Algorithms for computing boundary representations of solids defined through Booleans use various forms of set membership classification. Detailed algorithms for polyhedral objects are given elsewhere.[4,5]

The computations associated with Boolean operations involve surface-surface intersections. Reliable intersection techniques for the natural quadrics (planes, cylinders, spheres, and cones) were developed more than a decade ago by the PADL-2 group in Rochester and by others using closed-form parametric equations. Two papers presented at the 1991 Symposium on Solid Modeling Foundations and CAD/CAM Applications discussed the detection of all singular cases where these intersections degenerate into planar curves.[6]

Both the academic community and vendors have invested considerable effort in the development of reliable and efficient intersection techniques for more general parametric surfaces, such as nonuniform rational B-splines (NURBS). The current trend is towards a two-step procedure: loop and branch detection, and curve tracing. Intersection curves may form small closed loops, which sampling techniques cannot reliably detect. Algebraic methods and recursive subdivision techniques based on bounds on the surface and its Gaussian map have been proposed (two papers on loop detection were presented at the 1991 symposium on solid modeling mentioned above[6]). A number of authors have published various curve-tracing techniques. A combination of several approaches is discussed elsewhere.[20]

Algorithms for Boolean operations involve more than just computing the curves of intersection between surfaces. Selecting curve segments that are true edges of the solid (that is, curve classification) and constructing the correct representation of the solid's boundary also involve curve-surface intersections and ordering of surfaces around curves, or of edges around vertices. Ordering is necessary for reasoning about the presence of material in the neighborhood of edges or vertices. Intersection and ordering are difficult tasks, especially for high-degree surfaces.

When CSG models are available, you can perform many fundamental operations using the divide-and-conquer method, that is, recursive descent on CSG trees. Elsewhere we present divide-and-conquer algorithms for a more general domain for CNRG representations.[14]

Several techniques have been devised for quickly updating the boundary of a CSG solid when its definition is edited. Rossignac and Voelcker's active zones[21] and Cameron's S-bounds[14] successfully exploit the structural and spatial locality of editing operations. Several authors have suggested maintaining a decomposition of the Euclidean space into connected 3D cells bounded by the surfaces of the CSG primitives. Cells that satisfy a particular Boolean expression can be selected for display or for other computational purposes. Changes to the Boolean expressions do not require recomputing the subdivision, and local changes to the geometry cause only incremental modifications to the subdivision. Rossignac and O'Connor developed this subdivide-and-select approach for nonregularized Booleans and other operations on selective geometric complexes.[18].

Boolean operation algorithms construct the boundary of a solid by selecting subsets of the boundaries of the operands. Several operations that create new surfaces play an increasingly important role in solid modeling.

Minkowski sums can be defined by sweeping an object over another or, more precisely, by the vector sum: $A \oplus B = \{a + b : a \in A, b \in B\}$. We used the Minkowski sum of an object with a ball to define growing and shrinking operations on solids and to produce constant-radius blends (such as rounds and fillets).[22,23] Requicha used Minkowski operations to define the mathematical meaning of tolerance specifications. Minkowski operations also play a crucial role in robotic collision avoidance[24] and in accessibility and visibility problems.[25]

Recently, Kaul and Rossignac[26] at IBM proposed linear Minkowski combinations for constructing parameterized shapes, $C(t) = (1 - t)A \oplus tB$, that smoothly interpolate any two polyhedra $A$ and $B$. A user controls the shape of $C$ by changing the shapes and orientations of $A$ and $B$ and by selecting values for $t$ (see Figure 2). Real-time graphic feedback using a new parametric graphic representation permits the interactive exploration of the resulting family of shapes for all values of $t$. Several researchers are investigating extensions of Minkowski operation algorithms to curved objects.

The rigid-body transformations used to position objects also extend to more general, nonlinear transformations. For example, bending has been implemented by using the Cartesian coordinates $x$ and $y$ as the radius and angle in a cylindrical coordinate system. Deforming transformations can also be defined by a trivariate polynomial mapping from three-space onto itself and can be edited through control points, as Sederberg and Parry proposed,[27] or through point-displacement constraints (see Figure 3), as Borrel and his colleagues at IBM proposed.[6]

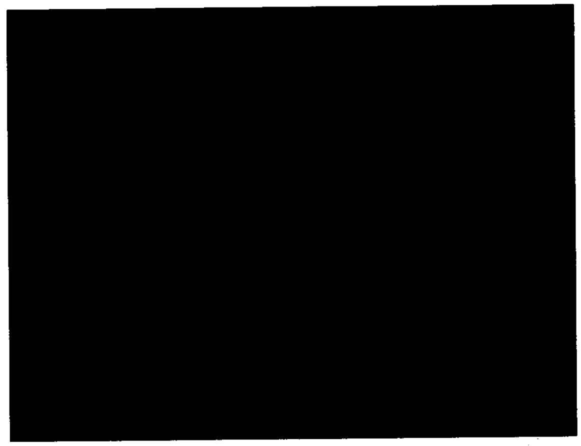


**Figure 3. A solid model of a teapot deformed using a space mapping defined in terms of displacement constraints editable in real time**

Deforming operations are difficult to integrate fully into solid modeling systems because we lack reliable and efficient algorithms for performing set membership classification on deformed solids, for combining them through Boolean and other operations, and even for verifying that the representations of deformed objects are still valid.

Despite some progress, the problem of numerical accuracy in geometric computations remains largely open. Edelsbrunner's "simulation of simplicity"[28] and related approaches avoid processing singularities by perturbing the data or by consistently ordering numerically equal values. This simplifies the code because you do not need algorithms for special cases involving singular geometric configurations, such as three collinear points. But these approaches have several drawbacks:

- They rely on exact geometric calculations requiring extended precision arithmetic or symbolic manipulation and typically are too slow for interactive systems.
- They might destroy singular geometric configurations a designer desires.
- They might produce results incompatible with mathematical properties of the operations simulated by the solid modeling systems. (For example, it is expected that $A \cap B = B \cap A$.)

**Figure 2. A new shape (center) is defined as a Mindowski combination of a hemisphere (left) with a cone (right).**

Approaches that use epsilon tolerances in numeric comparisons to decide when two geometric entities are identical recognize most singularities despite numerical round-off errors, but might also wrongly classify some geometric proximities as singular. They require more complex algorithms to deal with the singularities they introduce, and sometimes they are unreliable. Attempts at using extended precision rational numbers, algebraic symbol manipulation, or logical inference techniques have been reported, but are computationally expensive and limited to simple domains.

Much of the processing cost in solid modeling comes from generating and testing geometric entities that do not contribute to the final result and are immediately discarded. For example, naive algorithms for intersecting two polyhedral solids generate the intersection points between lines and planes for all the edge-face pairs, then test the points for membership in the edges and faces. You can improve performance significantly by avoiding unnecessary computation. Spatial bounds and directories help reduce the number of tested entities. Depending on the application, you can use mini-max boxes or enclosing spheres. Or, more elaborate hierarchical directories might be preferable, such as K-D trees or the extended octrees of Brunet and his colleagues at the Polytechnic University of Catalunia in Spain.[29] You can achieve large speedups with algorithms that use grids or other spatial directories, or with plane-sweep methods (known as scan-line algorithms in graphics).

Graphics and CSG processing have provided a fertile ground for studying performance improvements. Their combination has attracted several research groups, for example, at the Delft Institute of Technology in The Netherlands,[30,31] at Bell Labs,[8] and at the University of North Carolina.[32] The availability of parallel machines has recently added a new twist to the research on performance improvement techniques for solid modeling.

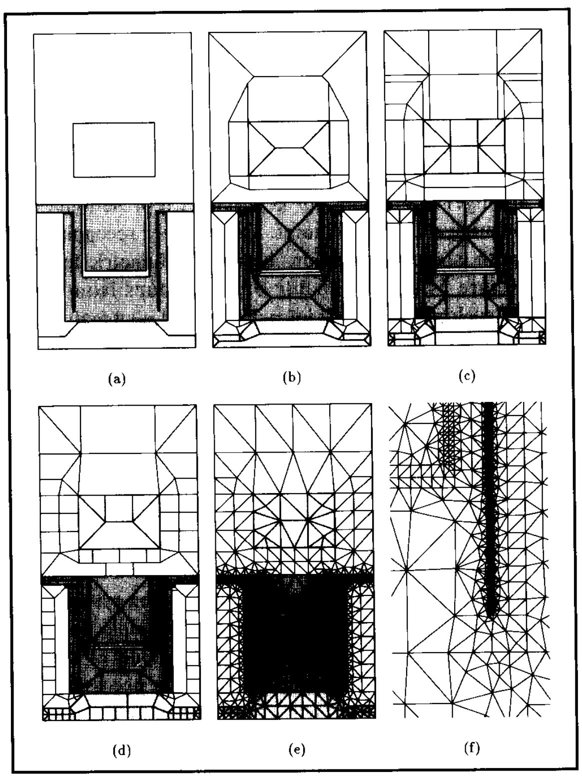



**Figure 4. Multiple domain input (a), its Voronoi diagram with parabolic arcs replaced by line segments (b), an initial partitioning (c), the result of removing slivers (d), the final mesh (e), and a close-up (f).**

## Applications

The development of solid modeling has been motivated primarily by its potential for supporting automated applications. Analytic applications have been successfully tackled, but automation of applications that involve synthesis or planning has proven more difficult. This section surveys applications of solid modeling currently supported or under development. Although not exhaustive, the survey gives an overview of what we can and cannot do today and in the near future.

Algorithms are available for visualization, mass property calculation, static interference analysis, kinematic simulation of mechanisms, and simulation of manufacturing processes such as machining. These application areas are relatively mature. Most commercial solid modelers of the current generation either support them or plan to offer them soon. Research and development in these areas now focuses on performance enhancement and other evolutionary issues.

Work on automatic meshing for finite-element analysis is progressing nicely. Two-dimensional algorithms are reasonably effective, and three-dimensional meshing is emerging. Several approaches are being pursued. Perucchio's group at the University of Rochester has developed meshing algorithms based on octree decomposition and on Delaunay triangulation.[6] Triangulation methods are also being studied at General Motors Research Labs and elsewhere. Octrees and other approaches have been under investigation by Shepard's group at Rensselaer Polytechnic Institute for several years. Nackman, Srinivasan, and their colleagues at IBM are developing decomposition strategies guided by an object's skeleton[33] (see Figure 4). The resulting decompositions resemble manually generated meshes. They tend to follow the general shape of an object and use a relatively small number of cells.

We expect industrial applications of automatic meshing technology within the next few years. (Note that what some vendors currently call "automatic meshing" are computer-aided procedures involving substantial user guidance.)

IBM's Oyster system for integrated circuit modeling has demonstrated simulation of semiconductor fabrication operations and analysis of the resulting 3D structures.[34] Various institutions are pursuing applications in electronic packaging.

Full automation of machining, robotic, and inspection applications remains an elusive goal. In robotic and inspection planning, solid modeling is especially relevant to issues of accessibility, spatial interference, and motion planning. (See Homem de Mello and Lee[19] and Latombe[24] for surveys of robot motion and assembly planning, and Spyridi and Requicha[25] for inspection planning.)

The remainder of this section discusses manufacturing planning for numerically controlled machining. NC machining shares many issues with other planning applications and has been the object of extensive research during the last 15 years. It shows that solid modeling is an important tool for the development of automated applications. However, we also need other tools because solid modeling is largely irrelevant to some of the issues that we must consider.

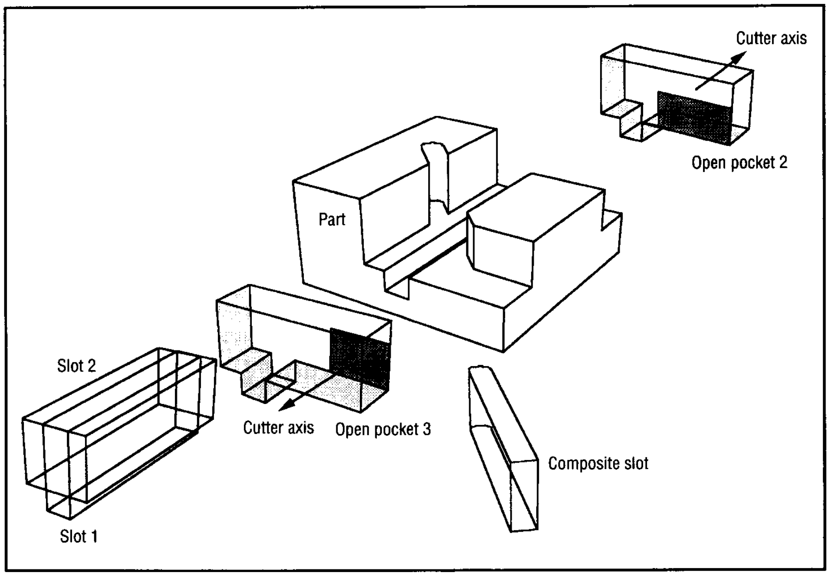



**Figure 5. Recognition of interacting features with Ooff.**

We can state the problem addressed by machining-planning research as follows: Given solid models of the desired part and the raw material, plus tolerancing, material, and other ancillary specifications, generate automatically a plan for manufacturing the part and actual instructions to drive NC machine tools and other manufacturing equipment.

We can decompose automatic machining planning into the following subproblems or tasks.

- Recognition of machinable features such as holes, slots, and pockets. For example, a planner must "know" that a certain surface or surfaces are associated with a hole feature, otherwise it cannot infer the need for drilling and reaming operations. Although some feature information can be captured directly at the design stage, recognition is needed because manufacturing features differ from those used in design. For example, a web is a design feature machined indirectly, typically by removing pockets that share portions of their boundaries with the web.

Feature recognition has been investigated extensively. Most existing recognizers search a solid's boundary for patterns of faces and edges that obey certain topological or geometric relationships. Typically, they rely on linguistic or graph-theoretic methods and only use a modeler to access the solid's boundary representation. An alternative approach, embodied in the Ooff recognizer,[35] uses face patterns as clues for feature existence and attempts to construct the largest volumetric features consistent with the data. (The name Ooff comes from Vandenbrande's exclamation when he finished his thesis.) Ooff analyzes stringent conditions for machinability such as accessibility and nonintrusion. This analysis involves complex geometric computations through calls to a solid modeler's internal procedures. Ooff uses artificial intelligence methods for generating, representing, and manipulating clues and features. Unlike other recognizers, it deals with general interactions between volumetric features (see Figure 5) and is capable of using clues from many sources, such as tolerancing information or design features.

- Process selection for each individual feature of a part. The last decade has seen substantial research on this topic. Most existent process selectors are rule-based expert systems that manipulate primarily "symbolic geometry," that is, sets of geometric attributes and predicates. For example, in a typical system a hole is represented by a data structure containing the feature type, plus attributes such as diameter and tolerances. A process selection rule stipulates that "if the size tolerance is less than a prescribed value $T$ the hole must be drilled and then reamed." The process selection system does not perform geometric computations (for example, intersections). Often it does not even have a global model of the part and its environment. It assumes that all the necessary features and attributes have been precomputed or are entered manually. Choice among competing alternative processes involves search, often guided by cost considerations.

- Process ordering. Selected processes must be partially ordered to reflect precedence relations. Process selection and ordering are closely related and usually addressed by a single system. Features typically contain predicates such as "intersects slot $S$." Sequencing rules stipulate, for example, that "if the depth of hole $H$ is larger than a threshold $T$, and $H$ intersects slot $S$, then $H$ must precede $S$."

• Setup planning. This involves the selection of part orientation and location with respect to machine tools, as well as the selection and placement of clamping devices and other fixtures. It is perhaps the least understood aspect of machining planning. Several experimental setup planners exist; they also are rule-based systems operating on symbolic geometry. Other systems analyze the balance of forces required for holding a workpiece, while recent research at Stanford University takes friction into account. Setup planning is related to grasping for robotic manipulation.

• Cutter and machine selection. Experimental systems for tool selection typically are rule based or use decision tables. They are largely independent of solid modeling concepts.

The outcome of the tasks just described is usually called a *process plan*—a list of machining and setup operations, plus associated resources such as tools and machines. Existent systems other than feature recognizers make essentially no use of solid modelers and operate primarily on feature data and associated attributes. Whether this is desirable or not is unclear at present.

The process plan must be expanded, through operation planning, into executable instructions for machine tools and human operators. Operation planning consists of two main tasks:

1. Parameter and strategy selection. This involves the choice of parameters such as feeds, speeds, depth of cut, and so on, as well as strategies for plunging into the material and for cutting it. For example, you might decide to clear a pocket by using either a zig-zag or a spiral cut. Rule-based systems suit these decisions well, and several such systems are under development.

2. Cutter-path generation. This task produces the actual code for driving the NC machinery. It often involves expensive geometric computations. Traditional cutter-path generation, implemented in most current commercial CAD/CAM systems, is not based on solid models. It uses surface or wireframe models and is based on iterative surface-following methods or offset computations. User guidance is necessary to ensure that features are accessible, no collisions occur, and the part is not over or under cut.

Publicly available literature on commercial algorithms is scant. Recently, algorithms for computing Voronoi diagrams, developed in theoretical computational geometry, were applied to generating cutter paths for pockets. They were developed independently by Chou at the University of Utah, Held at the University of Salzburgh,[36] and a CAD-CAM vendor. Feature-based machining systems, demonstrated in Chan's thesis at the University of Rochester,[37] are beginning to appear commercially. Such systems operate on solid models of features and parts, can ensure program correctness, and do not require user intervention.

## Design tools and user interaction

Computer-aided design is an iterative process that converts a designer's intentional model of functional requirements into an extensional computer representation of the final product. The process typically involves trial and error, editing, and successive refinements.

Solid modelers and other CAD tools have been used mostly for drafting and detailing rather than for the initial, conceptual stages of design—the most important because they largely determine a product's cost and performance. Conceptual design is still carried out primarily using traditional media, for several reasons.

First, the conceptual design process is poorly understood, and it is not clear how to support it with computational tools. Progress in this area has been slow. The annual Eurographics workshop on intelligent CAD systems reports on it.

Second, the design of 3D shapes appears to be inherently difficult, especially if user interaction must be channeled through a 2D medium such as a computer screen. Virtual reality, sometimes characterized as a solution in search of a problem, may provide the answer here and thereby find a major application.

Third, existing user interfaces, although adequate for detailing work, leave much room for improvement. This is, in part, a consequence of the first two problems above.

Fourth, and more prosaically, it has been impossible to provide graphic feedback to users at the necessary speed with acceptable cost. Algorithmic developments, the improved processing power of workstations, and, especially, dedicated graphics boards have solved this last problem for objects represented by tessellations. (A tessellation is a boundary representation in which all the faces are triangles, quadrangles, or other simple geometric entities.) However, computing tessellations remains a bottleneck. Some of the new special-purpose hardware discussed in the next section can break this bottleneck, or bypass it. The remainder of this section focuses on (human) user interaction with solid models.

The design of solids involves the instantiation of primitive geometric entities supported by the modeler. (Primitives need not be solids.) Direct specification by a user of the numeric values of the geometric dimensions, locations, and orientations of the primitives is very inconvenient. Most CAD systems offer construction tools for automatically deriving the desired parameter values from simple constraints, such as tangency between curves or containment of predefined points. Note, however, that the constraints themselves are not stored in most of these systems. They therefore can be broken by subsequent editing operations. For example, tangency between a line and a circle might be lost if the line is moved.

A user can edit models by typing new values for selected size or position parameters, or by linking these to analog input devices and interactively exploring the parametric design space. For example, six-degree-of-freedom joysticks are sometimes used for moving groups of primitives in the viewer's coordinate system, and direct manipulation techniques analyze 2D mouse movements to deduce the intended geometric transformations. Editing complex models in terms of their low-level primitives, such as vertex coordinates, is too tedious for practical use. Grouping, commonly used for editing in 2D drafting systems, is only effective for rigid-body transformations. Designers need higher level primitives for design and editing.

Constructive and feature-based representations provide editable high-level primitives. Constraint-based systems provide facilities for inferring and maintaining the numeric values of the large set of parameters needed to instantiate the primitives. Symbolic parameterization serves to define object families (sometimes called macros, user-defined primitives, or generic objects) and greatly facilitates instantiation of complex parts by specifying a few parameter values.

Symbolic parameterization can be achieved by directly replacing the primitives' parameters with symbolic expressions, which must be reevaluated when editing the parameter values. Expression evaluation is simple, but implies that constraints are unidirectional. The PADL-2 system takes this approach.[38] In indirect parameterization schemes, the parameters of primitives are inferred from their relationships with other, more meaningful parameters such as distances and angles. Indirect parameterization can be supported by sequential evaluation of parameter expressions. Rossignac proposed a method for adjusting the parameters of CSG models according to sequences of transformations specified in terms of the effects they should achieve—for example, in establishing a tangential contact between two curved primitive instances.[39]

Indirect parameterization through bidirectional constraints is more attractive from the users' point of view, but raises implementational problems because it involves systems of simultaneous, nonlinear equations. The variational geometry approach associates a parameterized boundary representation with a set of geometric constraints. Systems based on this approach are becoming popular in 2D drafting packages, where distance, radius, and angle constraints are either explicitly defined by the designer or automatically inferred by the system from singularities in the relative positions of primitive entities.

Variational geometry has not yet proven effective in 3D and suffers from several drawbacks in 2D. It leads to large systems of nonlinear equations in many variables. Iterative techniques for solving these equations are slow and sometimes fail to converge. There may be several solutions, and the system might select a different solution when a value is edited. The behavior is nonintuitive, since a small change might cause a jump to a different configuration. Given too many constraints, a user will have trouble deciding which constraints to enforce and which to relax to obtain the desired results. Users can create invalid boundary representations because no practical techniques exist for computing the bounds on parameter values for which the model remains valid. For more information and references on geometric constraints, see Chapters 2 and 4 of *Computer-Aided Mechanical Assembly Planning*[19] and several papers in *Proceedings of the Symposium on Solid Modeling Foundations and CAD/CAM Applications*.[6]

CSG is the most familiar constructive scheme. You can extend it in several ways to provide more powerful design capabilities. Traditional CSG primitives (solids such as blocks, spheres, cylinders, cones, tori, and simple extrusions) can, in principle, extend to a larger class of parameterized shapes, including generalized cylinders, volumes bounded by sculptured surface patches, or images of the unit cube through trivariate, vector-valued, polynomial functions. You can use surface editing techniques, such as manipulating control points or interpolated curves, to design these primitives.

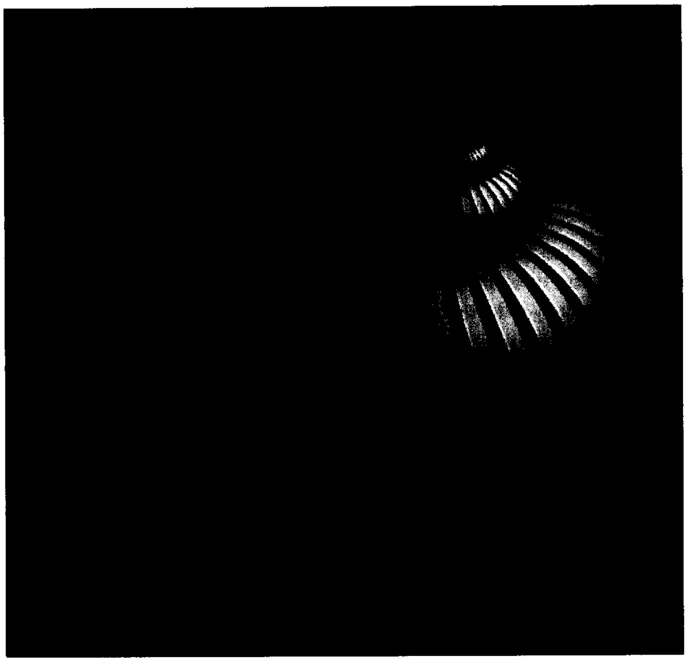


**Figure 6. A shell-like object, $A$, defined by sweeping a variable-radius sphere along a spiral trajectory, is approximated by a union of 600 spheres. The picture shows the union of $A - B$ with a translated version of $A \cap B$, where $B$ is another sphere used to split the object $A$ into two parts.**

Although the high-level algorithms for CSG are independent of geometric domain, their implementation for sculptured primitives remains an active area of research and development. Operations supported in a constructive representation can extend beyond Booleans to include solid offsetting (growing and shrinking), blending (rounding and filleting), bending,[40] Minkowski sums, and other sweeps. For example, we implemented an experimental system that integrated offsets and blends with Boolean operations in a domain of quadrics and tori,[22,23] and Kaul and Rossignac[28] implemented a Minkowski interpolation capability in a polyhedral domain.

Constructive representations are relatively easy to edit because they store the operations used to construct an object. These operations and their arguments can be selectively modified. Log or journal files, saved by many CAD systems, provide similar but much more rudimentary capabilities. Operations (for example, the regularized Booleans) that guarantee the validity of results are especially useful and can readily support symbolic parameterization. If model validity is not a major concern, we can use other shape-editing techniques. For example, we can tweak the entities in a solid's boundary representation by changing their parameters directly, or split a solid in two by cutting it with a patch. Both tweaking and splitting operations might produce invalid results. Several commercial systems offer them, primarily because they are fast. Automatically enforcing the validity of such operations slows them significantly and negates their main advantage.

You can construct feature-based representations by associating features with sequences of operations, typically Booleans, on parameterized primitives. But it is inconvenient to define objects solely by features, and it is unlikely that a system will have all the necessary features for every application domain and industrial environment. Also, engineering changes are sometimes formulated in terms of dimensions that do not correspond to the parameters of the original primitives or design features. Therefore, a user must identify new features on an existing solid model and use them for editing or for extracting data pertinent to a particular application. Typically, the identified features are associated with sets of the model's faces and do not correspond to the shape-modifying operations used to construct the model. This raises several delicate and open issues, discussed elsewhere.[41]

## Systems

Solid modeler architecture has not changed markedly over the last few years. There are (1) CSG modelers, which construct objects by Boolean operations, store CSG trees, and compute boundary representations when needed, and (2) boundary modelers, which store objects' boundaries regardless of construction method. In addition, most modelers store users' input commands for editing purposes and have additional, approximate representations such as octrees for speeding up computations by facilitating spatial search.

Recent advances in special-purpose hardware will probably strongly influence the architecture of future systems. Several interesting hardware projects are under way. Kaufman and his colleagues at SUNY, Stony Brook, designed a multiprocessor voxel machine.[42] Meagher, at the Octree Corporation, developed special-purpose hardware for octree operations and rendering.[43] Voelcker and his team at Cornell use ray representations to exploit the massively parallel Ray Casting Engine (RCE) developed by Kedem and Ellis at Duke.[6] The RCE board operates directly from CSG. In a couple of seconds it produces a ray representation with some 10,000 rays for a CSG tree with 512 quadric primitive half spaces. Figure 6 shows a shaded rendering computed by the RCE.

High-end graphics workstations now support trimmed NURBS patches directly. Special-purpose architectures for performing Boolean operations and detecting interferences were proposed by Yamaguchi at the Kyoshu Institute of Design in Japan[44] and by others. *Z*-buffer technology for visible-surface computation has practically eliminated the hidden-line-removal bottleneck, except for precise plotting applications. To directly render CSG models, Fuchs and his group at the University of North Carolina extended scan-conversion techniques supported by massively parallel Pixel-Planes graphics hardware.[32] Rossignac and his team at IBM

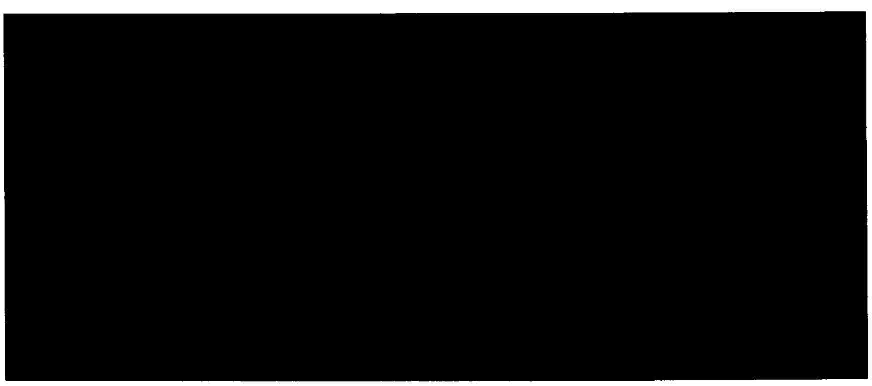


**Figure 7. A multiface cut performed on the union of two interpenetrating tori. The cutting volume and the viewpoint can be moved interactively with a real-time rendering of the hatched cross ssections and the highlighted interference between the solids.**

developed hardware-assisted techniques for displaying cross sections through assemblies of solids and for highlighting interference regions[45] (see Figure 7).

A geometric domain of polyhedral models, or of solids bounded by natural quadrics and tori, is too narrow for many applications. Designers need free-form or sculptured surfaces. NURBS are becoming a de-facto standard for free-form solid modeling. Since NURBS can represent the natural quadric surfaces exactly and can approximate all other surfaces of interest (such as tori and blends), developers can drastically reduce software complexity if they convert all surfaces to NURBS. However, the NURBS formulation produces considerable performance and reliability penalties. For example, algorithms that use closed-form solutions can quickly and accurately compute intersections of quadrics. Heterogeneous approaches, which distinguish between the various surface types, are faster and more reliable than homogeneous, NURBS-based implementations. But, they involve special-case analysis and increase the amount of code.

If hardware were available for direct computation on CSG models with primitives bounded by NURBS, homogeneous NURBS systems might become very attractive. Note, however, that even NURBS representations have descriptive deficiencies. Fixed-degree NURBS cannot describe offset surfaces exactly, and NURBS representations for quadrics hide the nature of the surfaces. For example, to reason about functional issues such as kinematic relationships or to select a peg-in-hole insertion strategy for assembly, it might be important to know that a surface is a cylinder. And deciding that a NURBS surface is a cylinder is a numerically unstable computation.

Traditional turnkey modelers, which are relatively monolithic and hard to extend or customize, will likely give way in the near future to more open systems built upon tool kits for geometric computation. A few tool kits are already commercially available or under development. For example, researchers at the University of Southern California are building a public-domain geometric tool kit in C++. Current modelers function both as geometric engines and as repositories for data. This is due largely to the inability of database technology to provide acceptable performance for geometric modeling applications. But the technology is changing rapidly, and it should soon be possible to build modelers that manage all of their data through object-oriented databases. Parallel object-oriented databases, such as the Omega system developed by Ghandeharizadeh at USC, seem especially promising.

Modelers and other CAD/CAM systems are moving to distributed, heterogeneous environments. Communication between systems requires standards for data exchange. Work on standards for geometric modeling has been under way for over a decade. The International Graphics Exchange Standard (IGES) provides for exchange of wireframe data in file format and has been in industrial use for several years. Presently, much effort is being devoted to the design of the Standard for the Exchange of Product Model Data (STEP), which also includes surface and solid modeling and can be used either for file transfer or database access. The International Standards Organization (ISO) is developing STEP, and the National Institute for Standards and Technology (NIST, formerly the Bureau of Standards) is taking a key role in developing the US version, PDES (product data exchange using STEP). Computer-Aided Manufacturing International's Application Interface Specification (AIS) goes one step further than STEP, proposing standard protocols for invoking modeling procedures. (Computer-Aided Manufacturing International, or CAM-I, is a not-for-profit consortium headquartered in Arlington, Texas.

## Conclusions

Solid modeling theory and technology are maturing rapidly. We have seen explosive growth in the field's scientific literature, and many solid modelers are commercially available. Solid modeling systems, although powerful, consume substantial computational resources. The dramatic increase in the speed-to-cost ratio of computing equipment over the last few years strongly affects the industrial applications of solid modeling. Companies now find it cost effective to provide large numbers of engineers with workstations capable of running solid modelers at interactive speeds for moderately complex objects.

Theoretical understanding of geometric modeling has been enhanced through notions of algebraic geometry. Elegant algorithms and concepts developed in theoretical computational geometry have begun to percolate towards applications—for example, algorithms for cutter-path generation based on Voronoi diagrams.

Significant advances have taken place in extending the modeling domain to cover materially and dimensionally inhomogeneous objects (sometimes called nonmanifold topologies in the jargon) and objects bounded by complex, free-form surfaces. Boolean operations continue to be fundamental in solid modeling, but Minkowski operations (growing and shrinking) and other sweep operations are emerging as perhaps equally important. Whereas Boolean operations do not create new surfaces, sweeps do, and necessitate domain extension. We have seen some progress in robust geometric algorithms. However, enlarging the modeling domain has an adverse effect on robustness because of the inaccuracies introduced by intersection algorithms and other numerical computations.

Despite steady progress, many of the problems identified over a decade ago remain open. Today we still face the following challenges: very large objects (with more than 10,000 CSG primitives, say); better user interfaces for constructing 3D objects; exploiting parallel and distributed system architectures; and robustness and numerical inaccuracies.

Total automation of a wide range of applications—a primary motivation for solid modeling—has proven harder than expected in the field's pioneering times. Applications that involve synthesis or planning are especially difficult. Brute force approaches are doomed by combinatorial explosion, and endowing the application programs with a modicum of intelligence is a nontrivial exercise. Application development remains a major challenge and a driving force in solid modeling progress.

Developers now use techniques from artificial intelligence to build more flexible and automatic CAD/CAM systems. Features, which we define as regions of an object that are meaningful for a specific activity or application, play an important role in current research on intelligent CAD/CAM. (The literature includes many definitions for "feature"; here we use one of the most common ones.) Features provide natural means to associate domain knowledge with object representations. They also serve to "discretize" some problems. For example, given a part representation in terms of manufacturing features, process planning becomes essentially a discrete or combinatorial search problem.

Current trends point away from closed and "pure" solid modeling systems and toward open systems for more general geometric computations, built upon tool kits and (perhaps) object-oriented databases, and distributed over networks. Instead of constructing a monolithic and general-purpose system, it seems advisable to build several more specific systems capable of communicating and cooperating with one another. The development of standards will facilitate this.

Geometric computation underlies many application domains. We see it emerging as a major interdisciplinary research and development area, encompassing design and analysis of geometric algorithms, geometrically oriented robotics, surface and solid modeling, and associated rendering and user-interaction issues. ❑

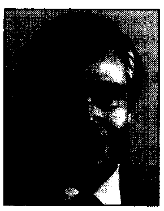
## Acknowledgments

Requicha's work was supported by the National Science Foundation under grants DDM-87-15404 and CDR-87-17322, by the industrial members of the Institute for Manufacturing and Automation Research (IMAR), and by the Industrial Associates of the Programmable Automation Laboratory, Institute for Robotics and Intelligent Systems (IRIS) of the University of Southern California.

## References

1. A.A.G. Requicha and H.B. Voelcker, "Solid Modeling: A Historical Summary and Contemporary Assessment," *IEEE Computer Graphics and Applications*, Vol. 2, No. 2, March 1982, pp. 9-24.
2. A.A.G. Requicha, "Solid Modeling: A 1988 Update," in *CAD Based Programming Sensory Robots*, B. Ravani, ed., Springer Verlag, New York, 1988, pp. 3-22.
3. A.A.G. Requicha and H.B. Voelcker, "Solid Modeling: Current Status and Research Directions," *IEEE Computer Graphics and Applications*, Vol. 3, No. 7, Oct. 1983, pp. 25-37.
4. C. Hoffmann, *Geometric and Solid Modeling*, Morgan Kaufmann, San Mateo, Calif., 1989.
5. M. Mäntylä, *An Introduction to Solid Modeling*, Computer Science Press, Rockville, Maryland, 1988.
6. J.R. Rossignac and J. Turner, eds., *Proc. Symp. Solid Modeling Foundations and CAD/CAM Applications*, ACM Press, New York, 1991.
7. M. Boyer and N.F. Stewart, "Modeling Spaces for Toleranced Objects," *Int'l J. Robotics Research*, Vol. 10, No. 5, Oct. 1991, pp. 570-582.
8. B. Naylor, J. Amanatides, and W. Thibault, "Merging BSP Trees Yields Polyhedral Set Operations," *ACM Computer Graphics* (Proc. Siggraph), Vol. 24, No. 4, Aug. 1990, pp. 115-124.
9. Y.E. Kalay, "The Hybrid Edge: A Topological Data Structure for Vertically Integrated Geometric Modeling," *Computer-Aided Design*, Vol. 21, No. 3, 1989, pp. 130-140.
10. K.J. Weiler, "The Radial Edge Structure: A Topological Representation for Non-Manifold Geometric Modeling," in *Geometric Modeling for CAD Applications*, M. Wozny, H. McLaughlin, and J. Encarnacao, eds., Springer-Verlag, May 1986.
11. B.G. Baumgart, "Winged-Edge Polyhedron Representation," Report No. Stan-CS-320, Artificial Intelligence Laboratory, Stanford University, Calif., 1972.
12. L. Guibas and J. Stolfi, "Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams," *ACM Trans. Graphics*, Vol. 4, No. 2, April 1985, pp. 74-123.
13. D.P. Dobkin and M.J. Laszlo, "Primitives for the Manipulation of Three-Dimensional Subdivisions," *3rd ACM Symp. Computational Geometry*, Waterloo, Canada, June 1987, pp. 86-99.
14. J. Rossignac, guest editor, "Special Issue: Beyond Solid Modeling," *Computer-Aided Design*, Vol. 23, No. 1, Jan./Feb. 1991.
15. H. Bieri and W. Nef, "Elementary Set Operations with *d*-Dimensional Polyhedra," *4th Workshop for Computational Geometry*, Wuerzburg, Germany, March 1988.
16. C. Sobhanpanah, "Extension of a Boundary Representation Technique for the Description of *n*-Dimensional Polytopes," *Computers and Graphics*, Vol. 13, No. 1, 1989, pp. 17-23.
17. E. Brisson, "Representing Geometric Structures in *D* Dimensions: Topology and Order," *5th ACM Symp. Computational Geometry*, Saarbruchen, June 1989, pp. 218-227.
18. J.R. Rossignac and M.A. O'Connor, "SGC: A Dimension-Independent Model for Pointsets with Internal Structures and Incomplete Boundaries," in *Geometric Modeling for Product Engineering*, M. Wozny, J. Turner, and K. Preiss, eds., North-Holland, Rensselaerville, N.Y., Sept. 1989, pp. 145-180.
19. L.S. Homem de Mello and S. Lee, *Computer-Aided Mechanical Assembly Planning*, Kluwer, Boston, 1991.
20. G.A. Kriezis, N.M. Patrikalakis, and F.E. Wolter, "Topological and Differential-Equation Methods for Surface Intersections," *Computer-Aided Design*, Vol. 24, No. 1, 1992, pp. 41-55.
21. J.R. Rossignac and H.B. Voelcker, "Active Zones in CSG for Accelerating Boundary Evaluation, Redundancy Elimination, Interference Detection, and Shading Algorithms," *ACM Trans. Graphics*, Vol. 8, No. 1, Jan. 1989, pp. 51-87.
22. J.R. Rossignac and A.A.G. Requicha, "Constant-Radius Blending in Solid Modeling," *ASME Computers in Mechanical Engineering (CIME)*, Vol. 3, No. 1, July 1984, pp. 65-73.
23. J.R. Rossignac and A.A.G. Requicha, "Offsetting Operations in Solid Modeling," *Computer-Aided Geometric Design*, Vol. 3, No. 2, Aug. 1986, pp. 129-148.
24. J.C. Latombe, *Robot Motion Planning*, Kluwer, Boston, 1991.
25. A.J. Spyridi and A.A.G. Requicha, "Accessibility Analysis for the Automatic Inspection of Mechanical Parts by Coordinate Measuring Machines," *Proc. IEEE Int'l Conf. Robotics and Automation*, May 1990, pp. 1,284-1,289.
26. A. Kaul and J. Rossignac, "Solid-Interpolating Deformations: Constructions and Animation of PIPs," *Computers and Graphics*, Vol. 16, No. 1, 1992, pp. 107-115.
27. T.W. Sederberg and S.R. Parry, "Free-form Deformation of Solid Geometric Models," *ACM Computer Graphics* (Proc. Siggraph), Vol. 20, No. 4, Aug. 1986, pp. 151-160.
28. H. Edelsbrunner, S.T. Tan, and A. Waupotitsch, "A Polynomial Time Algorithm for the Mini-Max Angle Triangulation," *6th ACM Symp. Computational Geometry*, 1990, pp. 44-52.
29. P. Brunet and I. Navazo, "Solid Representation and Operations Using Extended Octrees," *ACM Trans. Graphics*, Vol. 9, No. 2, Arpil 1990; pp. 170-197.
30. W. Bronsvoort, *Direct Display Algorithms for Solid Modeling*, PhD thesis, Delft University Press, The Netherlands, 1990.
31. F.W. Jansen, *Solid Modeling with Faceted Primitives*, PhD thesis, Technische Universiteit Delft, The Netherlands, Sept. 1987.
32. J. Goldfeather et al., "Near Real-Time CSG Rendering Using Tree Normalization and Geometric Pruning," *IEEE Computer Graphics and Applications*, Vol. 9, No. 3, May 1989, pp. 20-28.
33. V. Srinivasan et al., "Automatic Mesh Generation Using the Symmetry Axis Transformation of Polyhedral Domains," IBM T.J. Watson Research Report, RC 16132, 1990.
34. G.M. Koppelman and M.A. Wesley, "Oyster—A Study of Integrated Circuits as Three-Dimensional Structures," *IBM J. Research and Development*, Vol. 27, No. 2, March 1983, pp. 149-163.
35. J.H. Vandenbrande and A.A.G. Requicha, "Spatial Reasoning for Automatic Recognition of Interacting Form Features," *Proc. ASME Int'l Conf. Computers in Engineering*, Vol. 1, Aug. 1990, pp. 251-256.
36. M. Held, *On the Computational Geometry of Pocket Machining*, in the

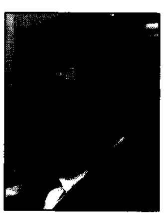
series Lecture Notes in Computer Science, G. Goos and J. Hartmanis, eds., Springer-Verlag, Berlin, 1991.
37. S. Chan, *MPL: A New Machining Process Programming Language*, PhD thesis, Univ. of Rochester, New York, 1987.
38. C.M. Brown, "PADL-2: A Technical Summary," *IEEE Computer Graphics and Applications*, Vol. 2, No. 2, March 1982, pp. 69-84.
39. J.R. Rossignac, "Constraints in Constructive Solid Geometry," in *Proc. 1986 Workshop on Interactive 3D Graphics*, F. Crow and S.M. Pizer, eds., ACM Press, Chapel Hill, N.C., Oct 1986, pp. 93-110.
40. A.H. Barr, "Global and Local Deformations of Solid Primitives," *ACM Computer Graphics* (Proc. Siggraph), Vol. 18, No. 3, July 1984, pp. 21-30.
41. J.R. Rossignac, "Issues on Feature-based Editing and Interrogation of Solid Models," *Computers and Graphics*, Vol. 14, No. 2, 1990, pp. 149-172.
42. A. Kaufman, "The Voxblt Engine: A Voxel Frame Buffer Processor," in *Advances in Graphics Hardware III*, A.A.M. Kuijk and W. Strasser, eds., Springer-Verlag, Berlin, 1990, pp. 85-102.
43. D.J. Meagher, "The Solids Enginer: A Processor for Interactive Solid Modeling," *Proc. Nicograph 84*, Tokyo, Nov. 1984.
44. F. Yamaguchi and T. Tokieda, "A Solid Modeler with a 4x4 Determinant Processor," *IEEE Computer Graphics and Applications*, Vol. 5, No. 4, April 1985, pp. 51-59.
45. J. Rossignac, A. Megahed, and B.O. Schneider, "Interactive Inspection of Solids: Cross-Sections and Interferences," *ACM Computer Graphics* (Proc. Siggraph), Vol. 26, No. 2, July 1992, pp. 353-360.



**Aristides A.G. Requicha** is a professor of computer science and electrical engineering at the University of Southern California, where he directs the Programmable Automation Laboratory, a research unit within USC's Institute for Robotics and Intelligent Systems. He has conducted research in geometric modeling and programmable automation since the early 1970s. He was with the Production Automation Project at the University of Rochester until 1986. While there, he and his colleagues developed the PADL solid modeling systems. His current work blends AI and computational geometry, with an emphasis on spatial reasoning and its applications in intelligent systems for design and manufacturing.

Requicha received the Engenheiro Electrotecnico degree from the Instituto Superior Tecnico, Lisbon, in 1962 and the PhD in electrical engineering from the University of Rochester, New York, in 1970. He is a fellow of the IEEE and a member of AAAI, AAAS, ACM, SME, and Sigma Xi. He serves as editor for *IEEE Transactions on Robotics and Automation*, the Academic Press journal *Graphic Models and Image Processing*, Chapman and Hall's *Journal of Design and Manufacturing*, and the Springer-Verlag book series on computer graphics.

Readers may contact Requicha at Computer Science Dept., University of Southern California, Los Angeles, CA 90089-0781, e-mail requicha@lipari.usc.edu.



**Jarek R. Rossignac** is Manager of Interactive Geometric Modeling in the Computer Science Department of IBM's T.J. Watson Research Center. He currently leads research projects in interactive design techniques for shapes, motions, and deformations; in hardware-assisted methods for interactively visualizing and verifying very complex models of mechanical assemblies; and in the fundamental representations and algorithms that extend the topological domain of solid modelers.

Rossignac received a PhD in electrical engineering from the University of Rochester, New York, in 1985 for his work on offsetting and blending solids. He received best-paper awards from *Computers and Graphics* in 1990 for his work on feature-based design and from Eurographics 91 for joint work on shape interpolation. He serves as associate editor for *Transactions on Graphics* and for *Computer-Aided Design*.

Readers may contact Rossignac at IBM Research, PO Box 704, Yorktown Heights, NY 10598, e-mail jarek@watson.ibm.com.