# Machine Interpretation of CAD Data for Manufacturing Applications

QIANG JI AND MICHAEL M. MAREFAT

*University of Arizona, Tucson*

Machine interpretation of the shape of a component from CAD databases is an important problem in CAD/CAM, computer vision, and intelligent manufacturing. It can be used in CAD/CAM for evaluation of designs, in computer vision for machine recognition and machine inspection of objects, and in intelligent manufacturing for automating and integrating the link between design and manufacturing. This topic has been an active area of research since the late '70s, and a significant number of computational methods have been proposed to identify portions of the geometry of a part having engineering significance (here called "features").[1] However, each proposed mechanism has been able to solve the problem only for components within a restricted geometric domain (such as polyhedral components), or only for components whose features interact with each other in a restricted manner. The purposes of this article are to review and summarize the development of research on machine recognition of features from CAD data, to discuss the advantages and potential problems of each approach, and to point out some of the promising directions future investigations may take. Since most work in this field has focused on machining features, the article primarily covers those features associated with the manufacturing domain. In order to better understand the state of the art, methods of automated feature recognition are divided into the following categories of methods based on their approach: graph-based, syntactic pattern recognition, rule-based, and volumetric. Within each category we have studied issues such as the definition of features, mechanisms developed for recognition of features, the application scope, and the assumptions made. In addition, the problem is addressed from the perspective of information input requirements and the advantages and disadvantages of boundary representation, constructive solid geometry (CSG), and 2D drawings with respect to machine recognition of features are examined. Emphasis is placed on the mechanisms for attacking problems associated with interacting features.

Categories and Subject Descriptors: 1.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*boundary representations; constructive solid geometry (CSG); curve, surface, solid and object representation; geometric algorithms, languages, and systems;* 1.3.8 [**Computer Graphics**]: Applications;

---

[1] Although literature on this subject uses the terms form feature, semantic feature, shape feature, semantic shape feature, and semantic form feature, we use the term *feature* exclusively throughout this article.

This research has been partially supported by the National Science Foundation under grant DDM-9210018 to M. M. Marefat, and partially by funds provided by the Electrical and Computer Engineering Department at the University of Arizona. Qiang Ji has been supported in part by funds from Paul Lever. The support is gratefully appreciated.

Authors' address: Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721.

1.5.1 [**Pattern Recognition**]: Models—*geometric;* 1.5.2 [**Pattern Recognition**]: Design Methodology—*pattern analysis;* 1.5.0 [**Pattern Recognition**]: General; 1.2.1 [**Artificial Intelligence**]: Applications and Expert Systems—*industrial automation;* 1.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding—*shape;* 1.2.9 [**Artificial Intelligence**]: Robotics; J.2 [**Computer Applications**]: Physical Sciences and Engineering—*engineering*

General Terms: Design, Performance, Reliability

Additional Key Words and Phrases: Artificial intelligence, automated process planning, computer-aided design, computer-integrated manufacturing, feature recognition, flexible automation

CONTENTS



## 1. INTRODUCTION

Design is a set of important processes that occur at different life-cycle stages of a product. Computer-aided design (CAD), in general, refers to using computers to assist with the various functions in the design process. Engineers consider CAD data to be the data that represent a product or component: in the domain of mechanical components these are often represented as a set of engineering drawings or a solid model of a component.

Although CAD has been used to assist with various design tasks, CAPP (computer-aided process planning) has usually referred to the collection of activities that convert a part design into manufacturing instructions that describe how to produce the part or how to build an assembly to satisfy the design specifications. In the domain of machined components, process planning involves finding the sequence of processes with which parts are to be machined (such as milling, grinding, drilling, etc.), the fixturing configuration to set up the part for each process to be carried out, and the tools to be used to carry out each operation in the sequence. In order to achieve this task for a component, process planners interpret the design data (the shape, surface finish, tolerances, etc.) based on process and tool capabilities.

Computer-integrated manufacturing (CIM) systems attempt to integrate design, process planning, and other functions (material handling, factory management, etc.) in a production environment. However, developing truly integrated manufacturing systems has proved not to be a trivial undertaking. One important reason has been that CAD data consisting of annotated engineering drawings or the solid model of a component are not manufacturing-specific and generally represent geometry by a low-level description of edges, vertices, and faces of a component, whereas process planners work with primitives such as slots and holes (and properties of the primitives such as dimensions and surface finish) that are shapes produced by processes and tools.

In order to overcome the integration barrier between design and process planning, a task which previously relied upon a manual interpretation process by an engineer, several conscious efforts have been made, all using the concept of features. As explained later, the strategy has been either to incorporate features in the CAD data during the design process or to extract the features from CAD data, or a combination of both. In the next section, we first consider features and what they refer to in the remainder of this article.

### 1.1 Features

There is no universally accepted definition of features. In fact, this has been one of the difficulties researchers have faced in this area. However, two recent books [Shah and Mäntylä 1995; Shah et al. 1994] have described features as groupings of topological entities from a component that are semantically significant in its production and thus need to be referenced together. Clearly this description implies that feature definitions are domain-dependent and application-oriented. For varied applications such as design, manufacturing, stress analysis, and the like, the part is therefore to be viewed in terms of different sets of features [Mäntylä et al. 1996].

From the point of view of process planning, a feature set can be visualized as consisting of shapes and technological attributes associated with manufacturing operations and tools [Shah 1991]. For example, pockets, slots, holes, and steps are examples of common machining features, instances of which are shown in Figure 1(a). Pockets and slots may be produced by milling and grinding process operations, and holes may be achieved by drilling processes. Although some researchers have broadened the notion of features to include such entities as tolerance features, surface finish features, material features, and the like, in this article the term is restricted to "shape" features [Shah and Mäntylä 1995] or groupings of geometric and topological entities from a component that correspond to primitive shapes produced by given manufacturing operations and tools. Henceforth, we concentrate on common machining features such as pockets, slots, bosses, holes, and so on because these have been the predominant features discussed in the literature: from here on, the term features refer to them unless otherwise stated.

In order to develop systems-handling features, existing research has approached the definition of features differently. Some work has regarded features as (closed) volumes with certain characteristics (rectangular block, with two opposite ends open, etc.), whereas other researchers have regarded them as a group of topologic entities (faces, edges, etc. that are not necessarily closed volumes) satisfying certain geometric relationships (four faces, pairwise parallel, etc.). Examples of research in each category include Sakurai and Gossard [1990], where a feature is defined as a collection of faces, and Vandenbrande and Requicha [1990, 1993], who define a feature as a volume. Figure 1(a) shows the same set of common features regarded as volumes or as a group of topologic entities. It is important to recognize how features can be defined differently because the proposed algorithms for feature recognition from solid models cannot be used interchangeably between the alternate approaches. Algorithms for feature recognition can deal only with a specific definition of features.

Although there are infinite possible shape patterns for features, it may still be possible to categorize them into groups or classes. Such a classification would be useful for feature support, for developing a standard terminology, and for data exchange. For example, features may be classified as polyhedral or nonpolyhedral. Features may also be classified as prismatic or rotational. The attributes associated with features may include dimension, orientation, toler-

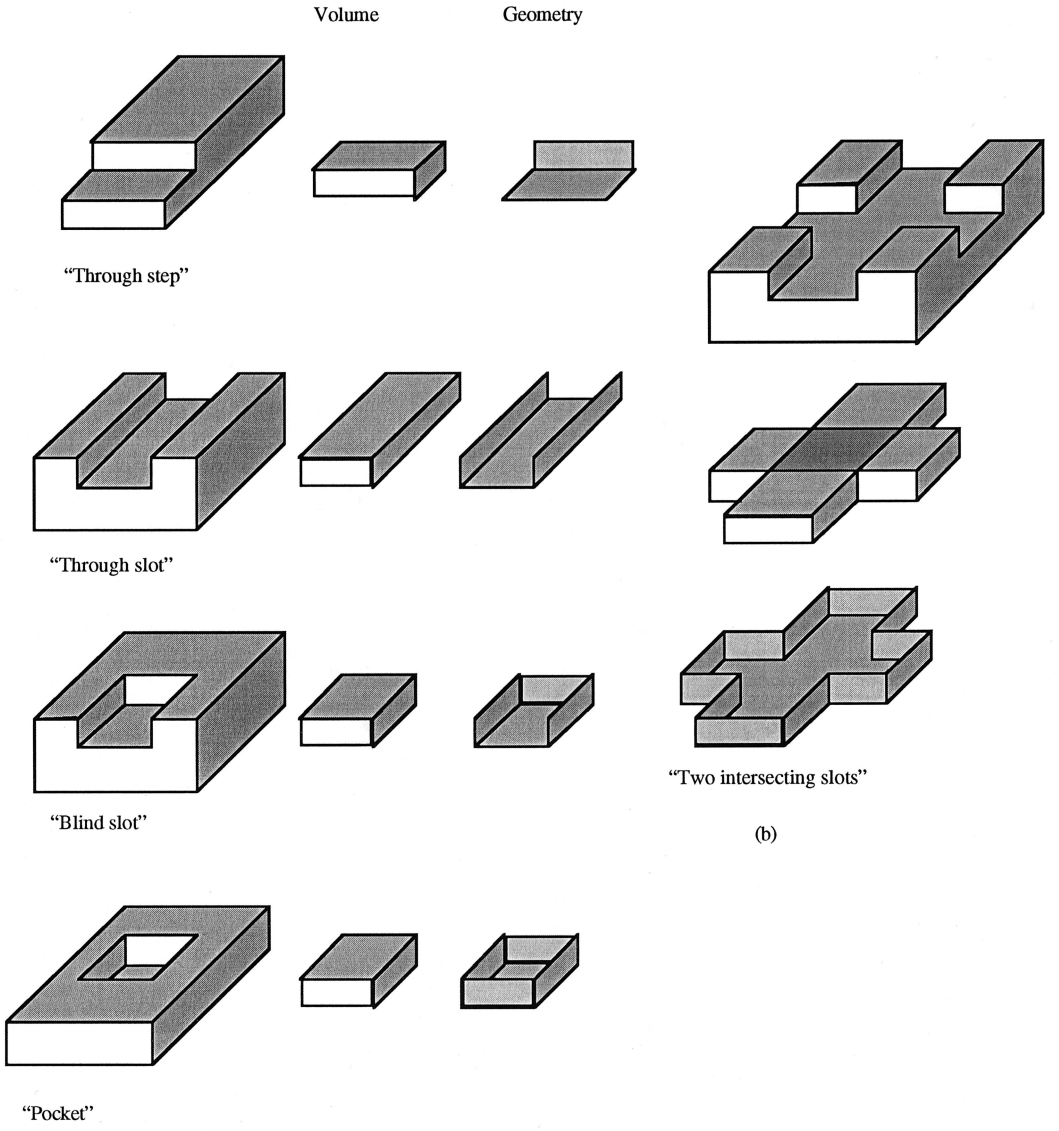


**Figure 1.** (a) Several common prismatic manufacturing features which can be defined volumetrically or geometrically; (b) part with interacting features.

ance, spatial relationship, and topologic components.

### 1.2 Interpreting Geometric Models to Obtain Features

An active area that has received much attention in integrating CAD and CAPP has been the development of an intelligent interpreter of CAD data (geometric models) to obtain features. Such an interpreter would serve to translate the low-level entities (faces) in the geometric models produced by a CAD system into a set of features suitable for manufacturing by means of an automatic feature recognition process (AFR) that would determine the features from an existing CAD-produced geometric model of a component such as a boundary rep-

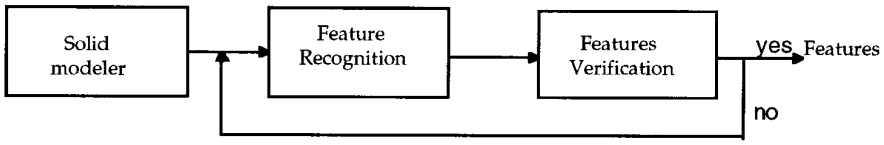


**Figure 2.** Component modules in automatic feature recognition.

resentation (B-Rep), a constructive solid geometry representation (CSG), engineering drawings, and so on. We discuss geometric models of components in the next section.

Figure 2 shows the component modules of automatic feature recognition [Shah 1991]. The constructed features constitute the high-level primitives that contain the semantic manufacturing information used for process planning or assembly. Once features are hypothetically found, they must be verified for correctness. This is generally done through either a set of rules, which determine if any given feature is not valid, or through volume-checking methods, which ensure that each feature generated is within the volume to be removed from the machined raw stock. The feedback loop shown in the figure is in place in case the verification fails, in which case the feature recognition process must be repeated to search for a different set of features and the new features must be verified in turn. We note that for a number of systems, some or all of the tasks of feature recognition are included within the duties of (computer-aided) process planning. For example, feature verification or even feature recognition itself may be considered a duty of the process planner. For clarity, we separate the feature recognition task from other process planning tasks (e.g., tool selection), and focus only on it.

A number of techniques for automatic feature recognition have been proposed in the past decade, but there have been difficulties associated with a lack of standard definitions. The multiplicity of feature definitions has sometimes contributed to different classifications for the same shape within the literature. For example, Figure 3 has been classified as a slot [Marefat and Kashyap 1990; Joshi and Chang 1988], a pocket [Gupta et al 1994], or a depression [De Floriani 1989].

Another difficulty in studying and proposing feature-recognition techniques has been robustness in handling feature interactions. When two or more features intersect geometrically, open into one another, and so on, this produces what is generally termed a feature interactions. The definition of feature interaction depends on the approach taken in defining features. As mentioned in the previous section, some work has regarded features as (closed) volumes with certain characteristics, and other studies have regarded them as groups of topological entities (not necessarily volumes) satisfying certain geometric relationships. For volumetric feature definitions, a feature interaction corresponds to an intersection of the volumes of two (or more) features. For topology-based feature definitions, an interaction corresponds to modification of the topological elements and the relationships between the elements that define each feature involved in the interaction. For example, Figure 1(b) shows a component with interaction between its two features, that is, two slots.

Regardless of whether features are regarded as volumes or as particular groups of topological elements, the difficulty of feature interactions for proposed techniques has been that the geometric interaction of features often produces a different version of a feature, and the characteristics of this new instance are different from the representational characteristics used by the technique to define the given class of

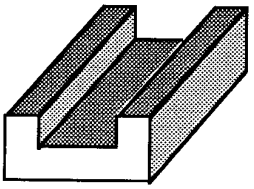

**Figure 3.** Example of simple part with slot.

features. The new version might have a different number of faces, a different number of edges, different geometric constraints (perpendicularity, etc.) between adjacent faces, a raw stock bounded volume that does not correspond to the expected volume for that class of feature, and so on. Because of these potential differences, feature-recognition techniques cannot therefore expect direct matches between the characteristics expected to represent a feature and the characteristics of all instances of that feature. This leads to nonuniqueness in characteristics defining instances of a feature and a need for capability and/or intelligence to cope with nonuniqueness and perform robustly in spite of it.

Another natural outcome when features interact is that there are inevitably alternative ways to describe a component according to its features, either by having alternative sets of matches among the geometric model of the component and the representations for features or, if there is no set of complete matches, by having alternative reasoning paths leading to alternative partial matches. For example, according to the features shown in Figure 1(a), the component in Figure 1(b) can be interpreted as having two interacting through slots, but an equally valid interpretation (based on geometric instances of shown features) would be four blind slots and a pocket in the middle. Being able to generate alternate interpretations systematically is very useful, because it allows manufacturing process planners to benefit from the information and generate process plans that are superior in terms of cost, quality, or both. Interpretations that provide access to all features from fewer access directions may produce better machining process plans, because it is estimated that as much as 80% of production time is spent in establishing different setups.

In the remainder of this article, we first review schemes for representing components via CAD geometric models and then discuss a variety of solutions to the problem of automatic feature recognition. We study advantages and disadvantages, as well as the application scope of each solution. The mechanisms for automatic feature recognition are divided into several categories of methods based on the overall approach: syntactic pattern recognition, graph-based, rule-based, volume- and cell-based, and evidence-based reasoning. Such an informal grouping is useful to better understand the state-of-the-art technology related to feature recognition from CAD geometric models. Associated with each technique are important issues related to feature recognition including how features are represented, algorithms developed for feature recognition, application scope of the proposed technique, underlying assumptions, and prototype systems developed. One emphasis of the article is on mechanisms for attacking problems associated with interacting features since, as we have noted, they have posed difficulties in feature recognition.

## 2. CAD AND GEOMETRIC MODELING

Design is an iterative process that involves proposing a design solution, testing and evaluating the design solution, modifying the proposed solution, and finally optimizing the solution. Within CAD, the graphics capabilities of a computer are substituted for the work that traditionally would have been done with pencil and paper. Furthermore, the simulation capabilities of the computer help the designer test and evaluate a proposed design solution. CAD can reduce the design cycle, increase design accuracy,

and free workers from tedious and repetitive work.

With the rapid development in database, simulation, and artificial intelligence technology, CAD's functions have evolved from simple computer graphics and computer-aided drawing and drafting to advanced 3D graphical representation, analysis, and simulation. Current CAD systems allow a user to design a 3D part, study the mechanical action of the part through simulation, and automatically produce engineering drawings of the part. The user can also analyze stresses and deflection of the part using finite element analysis techniques. The generic functions of a CAD system may include geometric modeling, engineering analysis, and automated drafting, as well as kinematics analysis. For the purposes of this article, we are concerned with the geometric modeling aspect of CAD systems. Such geometric models are either searched via automatic feature-recognition techniques or are augmented with feature information in feature-based design. Therefore, it is important to cover the basic aspects of geometric modeling.

### 2.1 Geometric Modeling and Representation Schemes

Geometric models are represented using *wireframes*, which represent the part shape with interconnected edge segments, or by using 3*D solid models*, which model the volume enclosed by the shape of the physical design. Since solid models carry more information than wireframe representations, most research on features has used solid models as input. In a typical solid-modeling system, the user constructs a model with building blocks of elementary solid shapes called *primitives*. The user may generate and/or modify a model by sizing, adding, and subtracting geometric solid primitives from a base component. The base component is typically a solid rectangular block called the stock.

A range of commercial solid modelers and design packages are available, but an important distinction must be made between solid-modeling packages and design packages. At the heart of a solid-modeling package is a kernel that supports solid-modeling operations such as intersections, differences, center of mass calculations, and the like. A design package (or standard CAD tool) generally allows the construction of design models (which may or may not include solid models), but may not allow access to any of the preceding solid-modeling operations. There are packages that combine a solid-modeling kernel and a design tool as well. ACIS (Spatial Technologies) is an example of a commercial solid modeler and Pro/ENGINEER (Parametric Technology Corporation) is an example of a commercial CAD package or design tool.

A representation scheme for solid modeling is defined as a mapping $S$ that maps physical objects from a domain $M$ into representations in a model space $R$; that is, $S$: $M$ $R$. In unambiguous representation schemes, each representation in the model space corresponds to one physical object in the domain. Unique representation schemes ensure that each physical object admits (can be mapped to) only one syntactically correct representation. Six major techniques (schemes) used to represent and maintain a 3D model by a CAD solid-modeling system are:

—pure primitive instancing (PPI),
—spatial occupancy enumeration (SOE),
—cell decomposition (CD),
—sweeping (S),
—constructive solid geometry (CSG), and
—boundary representation (B-Rep).

PPI involves reusing already stored descriptions of solids, such as blocks, brackets, and the like, and applying a transformation to them by instantiating certain parameters, to generate new objects. Although the original descriptions are referred to as *generic primitives*, the individual objects created through this transformation are called *primitive*

*instances*. PPI is a unique representation scheme, but because it lacks restrictions on the generic primitives, it is not necessarily unambiguous. PPI provides no way to combine object instances to create structures that represent new and more complex objects.

SOE subdivides 3D space into small volumes called *voxels* (an abbreviation for volume elements). To represent an object, it classifies these volumes as either empty or containing a solid. This is a variation of octrees in which 3D space is recursively subdivided into octants and classified as full, empty, or partially full. (Juan-Arinyo and Sole [1995] present a conversion from SOE to a variant of octrees.) SOE is unique and unambiguous but has the drawback of being potentially verbose, due to its enumerative nature.

Cell decomposition (CD) is similar to SOE, but it starts the decomposition with the object, not with 3D space. CD subdivides the object into primitive components that are either disjoint or meet precisely at a common face, edge, or vertex. The object is then thought of as being these primitive components glued together. Figure 4(a) shows the CD representation for a simple example object with a slot. Since engineering objects may be decomposed into constituting components in different ways, CD is, in general, not unique.

Sweeping is very closely related to the concept of a generalized cylinder. It is based on the notion of moving a two-dimensional set (a cross-section) along a three-dimensional space curve (axis) to sweep out a solid volume (Figure 4(b)). A sweep can generally be described by $W = W(\mathbf{r}, f)$, where $\mathbf{r}(s) = [x(s), y(s), z(s)]$ is a vector function representing the axis parametrically in terms of the arc length variable $s$. Normally, $f$ represents the boundary of the cross-section curve; that is, $f = (x(u, s), y(u, s))$, although $f$ may also be a set membership function describing the interior points of the cross-section at any point along the axis. The dependence of the cross-section on the arc length of the axis allows the cross-section to shrink, expand, or change in other fashions as it is swept along the axis. Although sweeping is intuitively appealing, like cell decomposition, it is not a unique representation scheme.

The last two solid-modeling techniques (CSG and B-Rep) have received the most attention and therefore are considered the most significant representation schemes. Constructive solid geometry (CSG) is a volumetric representation scheme in which solids may be represented as compositions, via (regularized) set Boolean operations, of primitive shape entities positioned properly in space via rigid motion operations. The primitive shape entities used in CSG are typically parameterized blocks, cylinders, cones, and spheres, and the Boolean operations include regularized union, difference, and intersection [Requicha 1980]. CSG represents and maintains objects as trees, the leaves of which are the primitive entities involved in constructing the object and the interior nodes of which are Boolean operations and motion transformations (Figure 4(c)). The "*" in the figure refers to *regularized* Boolean operations so as to prevent dangling edges and faces.

The concepts of adding and subtracting elementary volumes, called primitives, at an abstract level can be likened to manipulating features during the design process (similar to feature-based design). These CSG primitives may also translate into machining operations, which originally provided further motivation for the scheme. For example, drilling a hole can be interpreted as subtracting a cylinder from a base part. One drawback of the CSG representation scheme is that, in general, it is not unique.

Boundary representations (B-Reps) model objects by hierarchically storing their boundaries. The object boundary is segmented into a set of nonoverlapping faces. Each face is specified by describing the surface it is embedded on and its bounding edges. Each edge is, in turn,

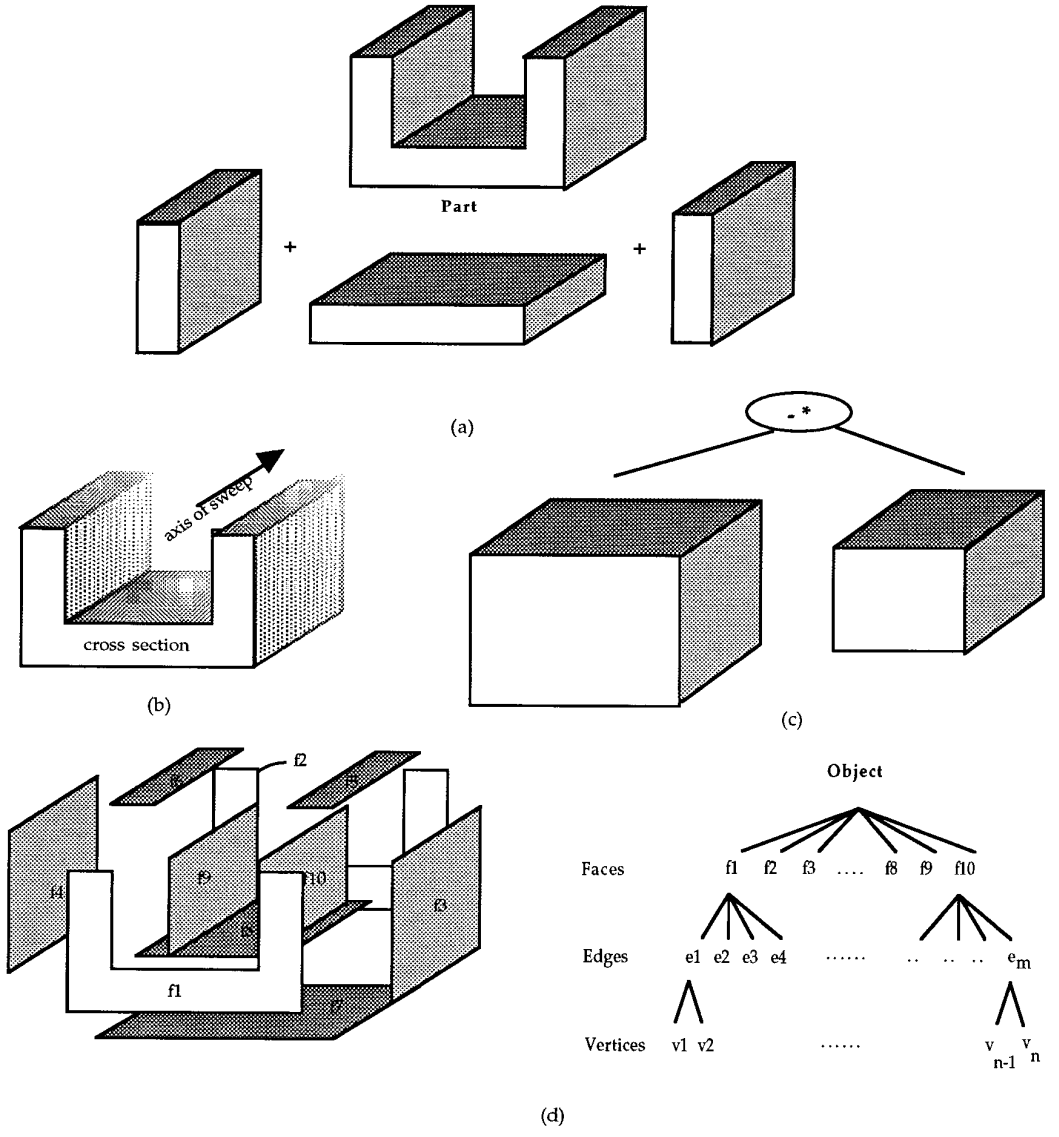


**Figure 4.** Solid model representation schemes for example object with a slot: (a) cell decomposition subdivides the object into (glued-together) primitive components that meet precisely at a face; (b) sweeping represents the object by a cross-section ($f$), and a sweep axis ($r(s)$); (c) CSG represents the object as a tree whose nodes are primitive volumes, Boolean operations, and rigid motions; (d) B-Rep scheme represents the object by describing faces, edges, and vertices forming its boundary in a hierarchical structure.

represented by the curve it lies on and any associated vertices. Vertices are simple three-dimensional coordinate points. The example in Figure 4(d) shows the boundary representation for a simple component with a slot. One of the earlier influential systems adopting this approach [Baumgart 1972] proposed "winged-edge" data structures for representing this information.

Boundary representation schemes are unambiguous [Requicha 1980]. They are

also unique when the boundary of the object is partitioned into maximal and connected faces. This aspect is desirable, because it makes B-Rep, unlike CSG, independent of the operations (or their order) used in constructing the model. Conversion of CSG representations to boundary representations is relatively well understood [Requicha and Voelcker 1985]. However, the inverse problem has not been well addressed until recently [Shapiro and Vossler 1993].[2] Due to CSG's nonunique nature, B-Rep is more commonly used for geometric modeling and feature recognition, but more important, nearly all automatic feature recognition mechanisms rely on B-Rep or CSG for their input.

Many of these geometric modeling systems use procedures based on the Euler theorem to test and help ensure validity of the models. Regardless of the solid-modeling scheme used, the output of current solid-modeling systems describes part topology and geometry in terms of low-level surface entities such as faces, edges, and vertices, possibly along with such information as surface finish, dimensions, density, tolerances, and so on.

The schemes previously discussed are "manifold" models with strict rules for their topological correctness. However, some applications require "nonmanifold" models and some of the rules on topological correctness need to be relaxed. Readers are referred to the selective geometric complexes (SGC) model [Rossignac and O'Connor 1990] for a further discussion.

### 2.2 STEP: The International Standard for the Exchange of Product Model Data

Finally, it should be noted that there is a developed standard for the exchange of solid model data, although it has been limited commercial usage thus far. STEP (the international Standard for the Exchange of Product Model Data) is being developed by the International Standards Organization (ISO) (STEP itself is ISO Standard 10303). STEP Application Protocol 203 (AP203) is the standard for the exchange of mechanical part and assembly data. PDES is the organization responsible for the testing and support of STEP within the United States. STEP consists of schemes to store and transmit geometric primitives data (solid models) using the EXPRESS language. These sequential file formats are defined to permit the transfer of product data between different CAD (or solid modeling) systems. The format is intended to be independent of the manner in which the information is created and stored within any particular CAD system. The information contained in this format may include geometric and topological information such as vertices of an edge, edges of a face, and faces for an object as well as certain manufacturing information such as density and dimension.

## 3. AN INFORMAL CLASSIFICATION OF FEATURE RECOGNITION MECHANISMS

All AFR algorithms include two important components: the definition of the features and the feature-recognition mechanism. Various approaches have been developed for automatic feature-recognition mechanisms that can be informally classified into the categories:

—syntactic pattern recognition,
—graph-based methods,
—rule-based methods,
—volumetric methods (including "cell-based" techniques), and
—evidence-based reasoning methods.

Syntactic pattern recognition characterizes the overall part shape as the composition of certain geometric primitives. Feature recognition proceeds by parsing the input syntactic expression of a part using grammar rules to identify the syntactic patterns representing fea-

[2] Due to the nonuniqueness of CSG, conversion to CSG representations is a difficult problem. Recently, however, Chirehdast and Papalambros [1994] developed a routine to convert SOE to CSG.

tures. In graph-based feature recognition, the topological shape of a part is represented as a graph (generally with nodes of the graph corresponding to the faces of the object and the arcs of the graph corresponding to the edges of the object. However, other graph representations, for example, Chuang and Henderson [1990], represent the object with a graph whose nodes are vertices of the object and whose arcs correspond to its edges). This graph representation is then searched for certain properties to identify the features embedded in the part. In rule-based methods, rules attempt to specify a set of necessary and sufficient preconditions for the patterns found in a feature. Recognition is carried out through an inference control mechanism that determines how to apply these rules to the input data. This includes forward chaining, backward chaining, or opportunistic rule firing. In volumetric methods, the finished material is represented as a combination of a set of volumes. If a CSG representation of the part is the input, the nonuniqueness of the representation would be a hurdle—there are many ways to define the same feature by different combinations of Boolean operations on the CSG primitives. Thus, the nonunique representation must be simplified before it can be recognized through pattern matching. In the convex-hull method, the part (or the volume to be removed from the part) is partitioned into subvolumes. The convex-hull algorithms compute the difference between the object and its convex hull recursively until the difference is a null set. The convex hulls are then rearranged to obtain removal volumes for machining. Cell-based techniques also decompose the part (or the volume to be removed from the part) into primitive volumes (cells). These cells are then recombined to provide the features of the original part. In the evidence-based reasoning method only hints or evidence, and not full-fledged features, are generated at first. Then the patterns are elaborated through hint recombination or evidence accumulation. Finally, there are a few methods that do not entirely fall into any of these categories, and we have classified them as "other" due to their unique or hybrid nature. (Note that many of the classified techniques still utilize several different feature-recognition strategies. In this sense, even the classified techniques could be called hybrid.)

## 4. MACHINE FEATURE IDENTIFICATION AND RECOGNITION TECHNIQUES

In this section, we survey various AFR techniques and summarize each technique's approach to feature definition, mechanisms developed for feature recognition, application scope, assumptions made, and limitations. AFR techniques are classified into three groups based on the their input information: B-Rep, CSG, and 2D models.

### 4.1 AFR Using B-Rep

The majority of AFR techniques proposed by researchers use B-Rep as their input information. These methods include syntactic, graph-based, rule-based, cell-based, and evidence-based reasoning methods, as well as some hybrid methods.

4.1.1 *Syntactic Pattern Recognition.* The syntactic pattern approach for feature recognition from boundary representations received much attention in the early '80s [Fu 1982]. Prior to that, it had been successfully applied to 2D recognition in computer vision. Syntactic pattern recognition is a formalized technique for representing complex patterns in terms of simple subpatterns and relations among subpatterns. A given pattern is decomposed recursively into simpler subpatterns called primitives. Just as an alphabet in a formal language may be combined into words and sentences, sequences of geometric primitives can be combined to form an expression that represents the complex patterns of features. The possible combination sequences can be organized ac-

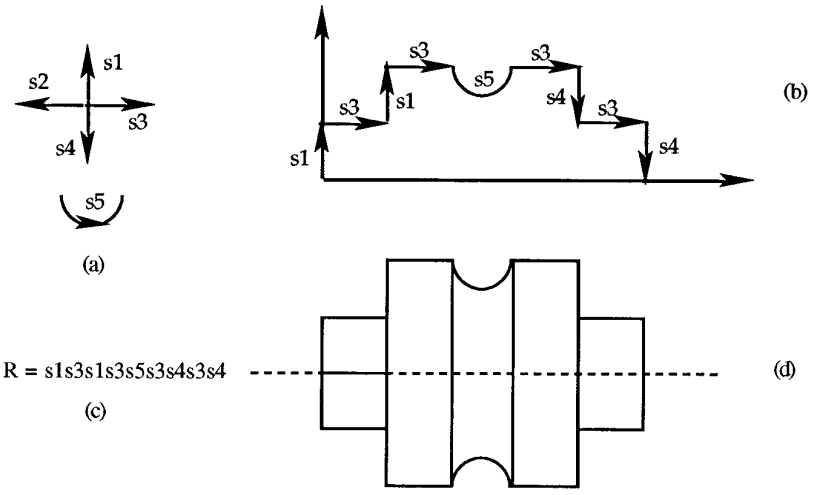


**Figure 5.** (a) Examples of basic primitives for line segments and curve segments; (b) described cross-section; (c) regular expression for the cross-section; (d) part corresponding to the contour.

cording to syntactic language rules, just like the grammar for a formal language. The resulting language (or expression) is called a pattern description language. The rules that define valid compositions of primitives into patterns are specified by the grammar of the pattern description language. The grammar is defined by the ordered tuple ($V_t$, $V_n$, $P$, $S$), where $V_t$ is the set of terminal symbols, $V_n$ is the set of nonterminal symbols, $P$ is the set of productions, and $S$ is the start symbol of the language. Correct sentences in this language are constructed by sequentially using productions from $P$ after the start symbol $S$ until a terminal symbol is reached. The recognition process proceeds by first constructing an expression for a pattern or object based on the defined primitives and the grammar rules. This expression, consisting of the string of primitives, is then parsed using the set of productions in $P$ to identify the feature pattern it represents. In most applications, patterns refer to contours of the cross-sections of a part and the primitives are usually line segments and curve segments (such as edges) that form the contours.

Jakubowski [1982] uses the syntactic approach for automatically interpreting rotational parts—parts that have an axis of symmetry—and polyhedral parts. With this approach, machine parts are described by such contour primitives as line segments (e.g., edges), curve segments, and surface segments, coupled with a set of operators such as revolution and sweeping, to describe the operations for generating a 3D part from its 2D cross-sections. Figure 5 shows an example based on this method. Figure 5(a) shows some of the basic geometric primitives. Groups of similar parts can be described by instantiating a generic description with certain parameters (e.g., the part shown in Figure 5(d) is represented (Figure 5(b)) by the instantiation of several of the $s1$–$s4$ primitives). A part is described by organizing the primitives into an expression according to grammar rules. Two grammar rules have been applied to construct the expressions for a part and to develop their parsers: extended context-free grammar and regular right part grammar. The most important task of the grammar rules is to distinguish the repetitive parts of contours from the nonrepetitive ones [Jakubowski 1982]. Figures 5(c) and (d) show a regular expression for the cross-section of a class of rotational

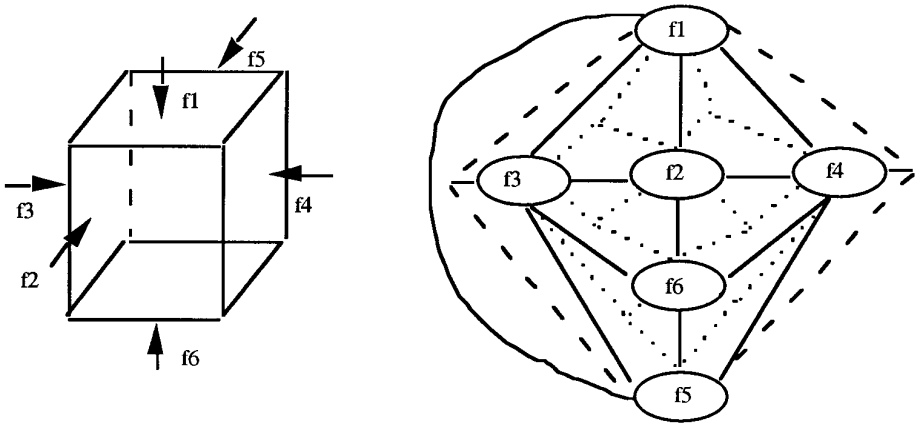


**Figure 6.** Cube and its face adjacency hypergraph representation.

parts, along with the contours represented by the expression and a part instance having such contours. To identify the features represented by the expression, parsers are constructed using the same grammar rules and are subsequently applied to the input expressions for the parts.

Similar techniques have been developed for more specific classes of parts. For example, Li [1986] uses a similar, but simpler, approach to identify turning features from the cross-sections of a rotational part. Choi [1982] focuses on an approach identifying hole geometries. Staley et al. [1983] developed a method designed for primitive depressions or protrusions whose cross-sections consist entirely of the chosen set of 2D primitive patterns, but cannot be extended to a feature for which the cross-section is inconsistent along different sections on the sweep axis. This is a common thread among all of these methods—since they all interpret a 3D part from its cross-section, they fail if the cross-section varies. All the methods use similar primitives from the part cross-section, and if the pattern does not match the symbols in the given grammar the methods do not work.

To recognize features in a general 3D part, Kyprianou [1980] proposed one of the earliest methods, which directly used syntactic pattern recognition coupled with a graph representation of the part. In this approach, the B-Rep of a part is first converted into a face-edge graph with nodes of the graph representing faces and arcs representing edges of the part. The arcs are labeled with the edges' concavity. The features are generated by parsing the face-edge graph. The significance of this work is that surfaces and edges are the basis for the pattern primitives instead of line and curve segments. Hence, it is possible to recognize 3D features directly without first converting to a 2D representation. The three basic structural primitives used in constructing the feature grammar are convex loop, concave loop, and smooth. A loop is an ordered set of faces, such that the first and last faces, and each pair of adjacent faces, each share an edge. A loop is defined as convex if its successive faces are separated by only convex edges. It is concave if it contains at least one concave edge and smooth if it contains edges on which there is no change in surface normal between the adjacent faces. Based on these primitives, a depression is identified if convex edges are on the inner loop of a face, and, similarly, a protrusion is identified by concave edges on the inner loop of a face. It is possible to identify different categories of features such as slots, pockets, simple or nested depressions, protrusions, and

the like, using this technique. The recognition consists of several steps. First, all edges and loops of the graph for a part are examined and the edges are classified as either concave, convex, or smooth. Then a list of faces to which features are attached is created. Finally, all the faces belonging to a feature are identified and the recognized feature is classified as either a depression or a protrusion of the part. Since the syntactic pattern definition of a feature may not define a closed volume, the recognized features are then completed by adding dummy entities such as edges, faces, and vertices to form closed solid volumes.

Choi et al. [1984] describe a similar approach except that they do not retain edge-concavity information in the graph of the part. Consequently, this approach can only recognize prismatic depression features. More recently, Falcidieno and Giannini [1989] extended the work in Kyprianou [1980] to parse structured face adjacency hypergraphs (SFAHs) rather than simple face-edge graphs. A face adjacency graph (FAH) is first defined as a triple $G = (N, A, H)$, where $N$ is the set of nodes in the graph corresponding to the faces in the object, $A$ is the set of all arcs in the graph corresponding to the edges in the object, and $H$ is the set of hyperarcs in the graph that, for each vertex of the object, connects all the nodes corresponding to the faces incident on the vertex. Figure 6 shows a cube and its FAH representation.

The features recognized by parsing the FAH graph are arranged into a hierarchical graph, the SFAH, which describes the hierarchy through adjacency relationships among the extracted features. That is, the SFAH is a graph whose nodes are the features of the part, and the arcs represent adjacency (or containment) of the features. The advantage of such a hierarchical description of the extracted features is that it provides a more global view of the part. The limitation of this method is that the system can only recognize features that define loops on the boundary of the object, that is, prismatic protrusion and depression features. As a result, it is not applicable to feature entities such as bevels, chamfers, and steps (area-features), since the loops for such features defined on the boundary of the object coincide with the external loop of a face.

To summarize, the syntactic pattern approaches have successfully been applied to 2D prismatic parts, rotational parts with turning features, and axis-symmetric volumes. However, success for nonaxis-symmetric 3D parts or rotational parts with nonturning features has been limited. This may be partially due to lack of a suitable language for 3D objects. Another limitation has been the ambiguity of the syntactic patterns [Wang 1992]. The primitives involved in the syntactic approach usually cannot represent certain geometric properties, such as the size of the primitive, relative orientation, edge concavity, and the like, that are important for distinguishing among features. This aspect may lead to one syntactic expression's corresponding with several different features and may, therefore, cause invalid shape constructs to be identified. Subsequent validation rules in many cases may be too difficult to derive and these rules may not be sufficient to filter out these spurious constructs. Since using the 2D patterns in syntactic pattern recognition severely limits the ability of the language to deal with the world of CAD parts (which are generally 3D), it is possible to write the rules as general inference rules. However, this technique will make the method more like a rule-based approach than syntactic pattern recognition and, therefore, largely defeats the purpose of syntactic parsing. The majority of syntactic pattern recognition techniques were developed almost a decade ago, and few current feature-recognition schemes employ this strategy.

Neural networks have been used recently in Prabhakar and Henderson [1992] to attack the problem of feature

recognition. The neural networks in this research perform pattern matching on a modified B-Rep input to recognize features. (Although this pattern matching could also be considered a type of rule firing, it is justifiable to discuss this approach here.) For this approach the B-Rep model of a part is converted to an adjacency matrix that describes the adjacencies between each pair of faces in the part. The matrix is then parsed for patterns (i.e., pattern matching) with a different neural network for each defined feature. If the neural network for a given feature finds a match within the adjacency matrix, then that particular feature is recognized.

4.1.2 *Graph-Based Approaches.* The graph-based approach, which gained momentum toward the end of the '80s, is currently one of the most prevalent feature-recognition techniques. Although many of the very recent feature-recognition works have moved away from graph-based techniques, they still present an important school of thought on the feature-recognition problem. One reason the graph-based approach is popular is that it can directly use many developed concepts and algorithms from applied mathematics, especially in graph theory and topology. In graph-based feature recognition, the B-Rep of a part is translated into a graph representing its topology. Primitive features are also represented by graphs as templates. Usually, the graph representation consists of nodes and links corresponding, respectively, to the faces and edges of the part. Additional information may be incorporated into the graph to represent the properties of geometric entities such as concavity and face orientation. The graph representation is then searched, using subgraph isomorphism, for certain properties to identify the features embedded in the part that match the templates of the primitive features. The identified subgraphs are subsequently extracted as the features embedded in the part.

Graph-based feature extraction was first introduced in Joshi & Chang [1988] through the attributed adjacency graph (AAG) for a part. The AAG makes use of the B-Rep information on faces and edges. An AAG is defined as a triple $G = (N, A, T)$, where $N$ is the set of nodes, $A$ is the set of arcs (links), and $T$ is a set of attribute values for arcs in $A$. The arc values in $T$ are either 0 or 1, marking the arcs as concave or convex, respectively. Figure 7(a) shows a part (an object with a slot feature) and its AAG. The nodes $F1$–$F10$ represent the faces and the arcs represent the edges. An arc is labeled 0 if the corresponding edge is concave and 1 if the edge is convex.

In this method, the topological and geometric relationships for a depression feature are first translated into a local AAG that represents feature faces and has only concave links. Unique properties for a particular feature are subsequently extracted from the local AAG of a feature and represented in terms of heuristics that define the feature. For example, Figure 7(b) shows the AAG for a pocket and a generic rule for the definition of a pocket may be given as:

—its graph is cyclic,
—it has exactly one node "$n$" with the number of incident 0-arcs equal to the total number of nodes $-1$,
—all other nodes have degree three, and
—after deleting node "$n$" the number of 0-arcs is greater than the number of 1-arcs.

Such a definition of a pocket is general enough to identify a wide range of instances of this feature. It also allows the recognition of nested features.

Instead of directly applying subgraph isomorphism to the AAG of the part, which is computationally intensive, a heuristic method is proposed to divide the part graph into components that could contain features. The heuristic is based on the following observation: a face that is convexly adjacent to all its neighboring faces (a convex node) is not likely to belong to a depression feature.

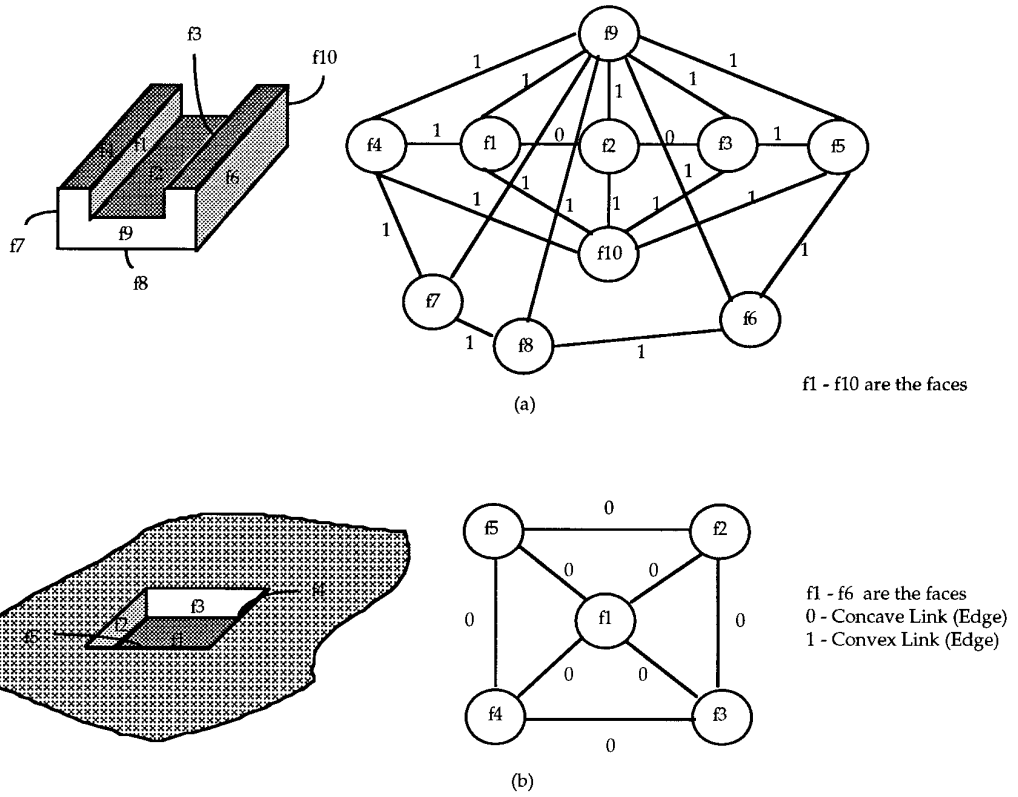


**Figure 7.** Examples of AAGs for (a) part with a slot, and (b) pocket.

The removal of such convex nodes will separate the AAG into several connected components, and if there are no interactions between features, each of these connected components (a subgraph) will represent an isolated feature or a depression in the original part. The recognition method, which is based on subgraph isomorphism and graph-based heuristics, is then applied to each component to identify the kind of feature represented by the component. To handle interacting features, Joshi implemented a heuristic rule that splits arcs and nodes to form complete feature subgraphs. However, this heuristic splitting can handle only some interactions between features. The difficulty arises when there are interactions that can possibly destroy an adjacency between two faces. This in turn means that the pattern of the feature within the AAG is lost, and hence unrecognizable. Therefore, more complex interactions in which the graph patterns of the features involved are modified are beyond the capability of this method, which requires more information than simple heuristics.

The significance of this research is in being among the first to propose the graph-based approach for extracting machining features. The advantage of Joshi's method is its use of both forward chaining and a heuristic method to reduce the computational effort. Figure 8 shows a simple part whose constituting primitive features are not correctly identified by the previous approach. The heuristic eliminates face 1 and thus the feature formed by faces 1 and 2 (a step as defined by Joshi) will not be recognized. Another limitation of this approach is that the derived rule set rep-

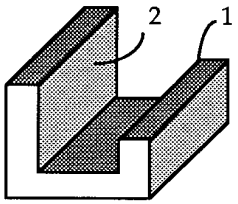


**Figure 8.** Interacting feature not correctly recognized using AAG.

resentation used for the description of a feature may be ambiguous. Specifically, the same rule may describe two different features as shown in Figure 9, where two different features are represented by the same rule set due to the ambiguity of the feature definition rule.

De Floriani [1989] introduces a similar graph-based approach to identifying the topological features of an object. The graphs used in this technique are called edge-face graphs (EFG). In an edge-face graph, a cut node corresponds to a face in the object that splits the graph into two or more connected parts described as biconnected components. Similarly, separation pairs correspond to pairs of faces on the object which, when removed, may split the object into connected pairs. Thus, the feature extraction algorithm decomposes the EFG into biconnected and triconnected components. The components are subsequently organized into what is called an object decomposition graph (ODG). Based on arcs incident on a component in the ODG, an entity is classified as a DP-feature or an H-feature. DP-features are protrusions or depressions on a set of faces of the object and H-features are through-holes, handles, or bridges. Figure 10 shows examples of some objects with their associated ODGs. The method provides a global understanding of the object shape. Construction of the ODG provides not only a list of features but also their relationships in the global shape of the part. Unlike local extraction based on geometric information, this method can identify "compound" features formed by a combination of through-holes and protrusions or depressions. However, the identification does not provide a sufficiently detailed classification for manufacturing and engineering purposes. In order to be appropriate for automation and planning functions, additional mechanisms to identify manufacturing-related primitives, such as pockets, slots, and holes that comprise the DP- and H-features, would be necessary.

Henderson et al. [1990] proposed a similar graph-based approach based on the assumption that distinct feature subgraphs can be isolated from their environments (the body graph) by detection of so-called *cut vertex nodes*. Cut vertex nodes represent the entrance faces linking the features to the body part. Another graph-based algorithm for feature extraction suggested by the same research group is called vertex-edge pattern extraction [Chuang and Henderson 1990]. According to this technique, the topological and geometric properties around a vertex can be used for its classification. Several types of vertices are suggested. Based on this vertex classification, each feature can be represented by a vertex-edge (V-E) graph, in which the nodes represent the type of vertices and the links represent the edges. This approach follows the concepts of bi- and triconnected components proposed earlier in De Floriani [1989]. Pattern-matching algorithms are then used to find the subgraphs of an object V-E graph that match V-E graphs for features.

The limitation of the cut-vertex method is that there is no proof that a cut vertex node always represents an entrance face that links a feature to the body of the part. This aspect is further complicated by situations in which a feature may have more than one entrance face in the part.

Sakurai and Gossard [1990] proposed a different approach to graph-based feature definition and recognition. A feature can be a cavity, a protrusion, or neither. In this technique, a feature is represented by a *feature graph*. A feature graph is a B-Rep of the feature's

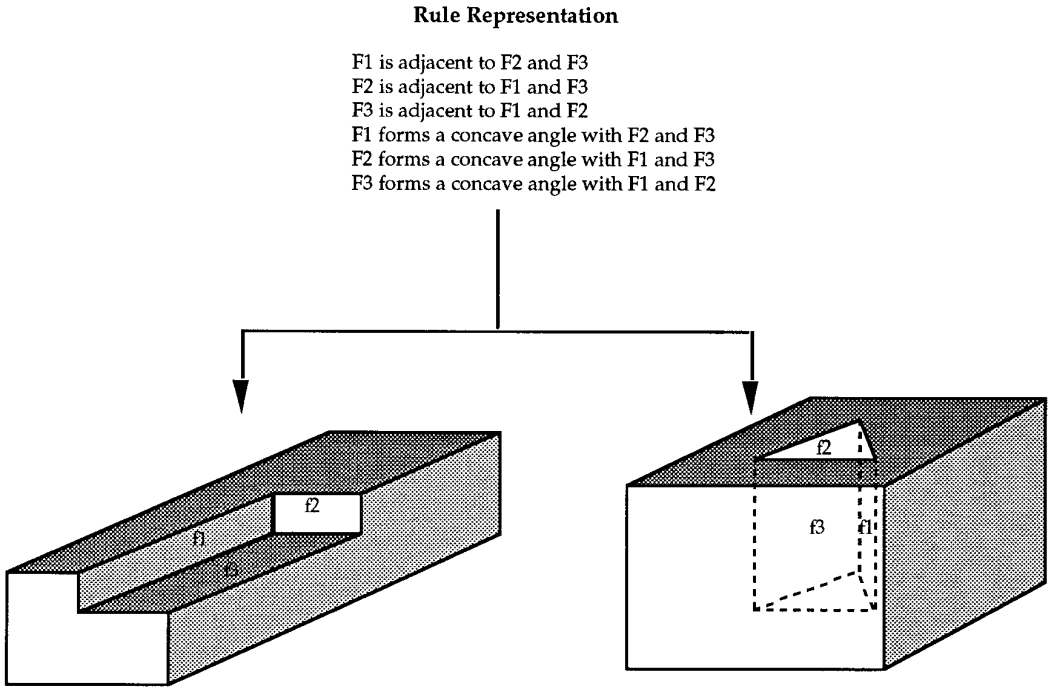


**Figure 9.** Ambiguity of feature description using rules.

faces augmented with user fact nodes and parameter nodes. Both user fact nodes and parameter nodes can refer to the same entity in the B-Rep graph. Although parameter nodes specify an attribute of a feature such as width, length, and the like, user fact nodes may supply additional geometric information such as parallelism, coaxiality, and perpendicularity among faces. Figure 11 shows a generic feature graph. A *template feature* is the generic definition of a feature, and a recognized feature is an instance of a template feature. The template feature is defined by interactively selecting a set of faces in a graphical display of the solid model and coupling them with user facts and parameters. Feature recognition is accomplished through graph matching by searching the entire solid model to find instances for each template feature. The graph matching is implemented by comparing each face of the template feature with every face of the solid model to determine whether the two faces have the same geometric type, a matching number of loops, and a matching number of edges.

To recognize interacting features, the algorithm first removes the recognized features from the original object. When the recognized feature is a cavity, the feature removal generates a volume that fills the cavity and it is added to the solid model. On the other hand, when the recognized feature is a protrusion, the procedure generates the volume of the protrusion and subtracts it from the solid model. The purpose of such removal is to simplify the shape of the solid model so that other yet unrecognized features can be identified.

The advantage of this method is that each feature can be defined easily and interactively without requiring a sophisticated language. However, there are interacting situations in which the features are not correctly identified, as when intersecting features have coincident faces when features have volumes that split each other completely (i.e.,

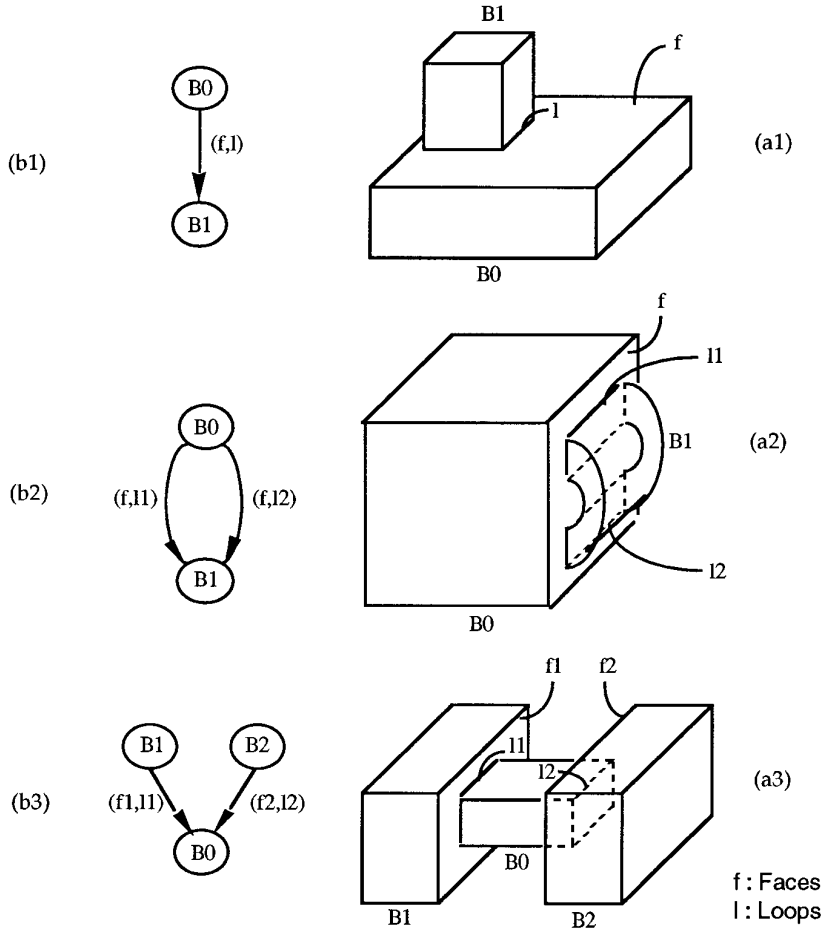


**Figure 10.** Biconnected components and their object decomposition graphs: (a) protrusion, handle, and bridge, with associated ODGs depicted in (b) [De Floriani 1989].

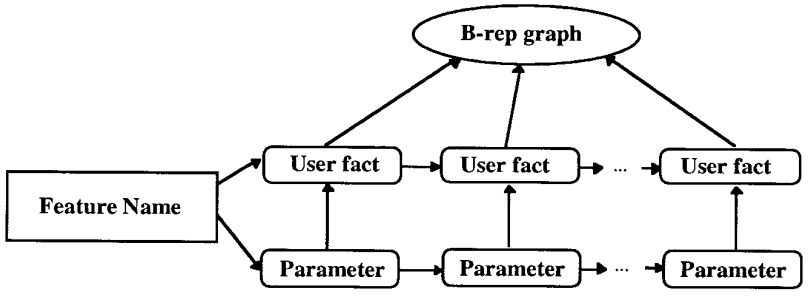


**Figure 11.** Schematic representation for feature graph [Sakurai and Gossard 1990].

the volumetric difference between the two features is two strictly disjoint regions of space). A general problem with such face-based pattern recognition methods is that it is difficult to create a feature volume from a set of faces that

do not enclose a volume. The removal of a volumetric feature also tends to complicate further recognition in cases of arbitrary interactions between features because it tends to generate extraneous elements or alterations that may complicate recognition of the involved features.

Corney and Clark [1991] developed a graph-based algorithm for identifying holes and pockets with interactions in the form of nonunique entrance faces. The method first constructs an aspect face-edge graph from the edge-face graph by deleting nodes that are parallel or antiparallel to a particular direction of view. Simple cycles in this aspect graph are identified and line segments linking the vertices associated with every pair of adjacent faces are drawn. By considering the loop formed by a projection of these line segments the partially destroyed face adjacency is recovered. The projection is taken on a plane normal to the direction of view. In recent related work, Clark and Corney [1994] described similar steps for recognizing general depressions and protrusions of a part, which can then be mapped into domain-specific features.

Fields and Anderson [1994] introduced an oriented face adjacency graph (OFAG), similar to the face adjacency hypergraph, to address the problem of feature recognition. As with the FAH, the OFAG for a part has nodes that correspond to the faces of the part and arcs that correspond to the edges of the part. However, between any two nodes (faces) there are two directed arcs, each augmented with concavity/convexity information as well as information about where each face intersects with another. One face can intersect the exterior or interior of another face. Using this new graph, Fields presents a linear-time algorithm for matching templates within it for the recognition of features. The sacrifice involved in a time-efficient algorithm is that this approach classifies sets of faces in the part into categories of features. Although such a categorization is not necessarily a drawback, it does mean the features extracted are further removed from domain-specific features than in the approaches previously described.

It has been pointed out [Marefat et al. 1990] that one shortcoming of many graph-based techniques is that the representations carry insufficient information for unique identification and recognition of features. Important information such as the relative orientation between object faces is not effectively used, which may lead to ambiguity and nonuniqueness in the representation and recognition of features, especially in the cases involving interacting features. Another problem that needs to be addressed is related to the mechanisms needed to validate the extracted features.

The success of graph-based methods has focused for the large part on polyhedral objects. In addition, most graph-based approaches consider only a limited set of patterns for the definitions of features. Searching for more general definitions becomes complicated due to the large number of patterns. Also, an important inherent problem in all graph-based techniques is computational complexity. Graph construction for both the primitives and parts and subgraph isomorphism can both be computationally expensive. As discussed in Sedgewick [1984], the problem of subgraph isomorphism corresponds to determining whether two graphs can be made identical by renaming the vertices. The general problem remains computationally intractable, although efficient algorithms for special types of graphs are known. To ease this problem, research is now being directed towards reducing the required subgraph matching time by applying heuristics. Peters [1993] presents an example for the domain of sheet metal parts to show that engineering knowledge can be used to reduce a theoretical combinatorial explosion to tractable bounds. Researchers have also succeeded in dividing the graph representation of a part into feature and body components so that pat-

tern-matching only applies to feature components, thus reducing the search space [Joshi and Chang 1988; Henderson 1984; Henderson et al. 1990; Marefat and Kashyap 1990; Marefat et al. 1990]. However, simply reducing the search space does not necessarily expand the domain of parts that graph-based methods can handle. To expand the scope of this work, mathematicians have developed efficient algorithms [Wang 1992] for identifying the graph nodes that separate feature subgraphs from body subgraphs.

Computational complexity is not the only drawback of graph-based approaches. Graph-based approaches are typically weak at recognizing features that intersect (interacting features). The feature graph representations are built from the topology of the part. However, in feature interactions the face adjacencies in the topology that are changed as a result of interactions make this information nonunique. Therefore, when exact matches are required in graph-based approaches, there is often difficulty in correctly identifying features within interactions.

Another potential limitation of graph-based approaches is verbosity in terms of the number of graphs required to represent the features. As illustrated in Gadh and Prinz [1992], for each primitive and its variants, a new set of graph representations must be created. Therefore, mechanisms must be developed to accommodate the variations in feature shapes. Due to these limitations, much current feature-recognition research has moved away from direct graph searching in favor of other approaches. Nevertheless, graph-based matching still appears in some current research, although generally only at a final stage in the recognition process when the problem has been sufficiently reduced in size.

4.1.3 *Rule-Based Approaches.* The rule-based method for feature recognition gained much momentum in the mid-'80s, after successful applications of expert systems to other domains. As in other expert systems, in which rules are used for knowledge representation, inference rules in feature recognition are used to capture knowledge about geometric and topological properties of features. A rule-based feature system consists of rules, an inference mechanism (or rule interpreter), and working memory. Rules attempt to specify a set of necessary and sufficient preconditions for the patterns found in a feature. The antecedents of the rules are statements describing the geometric and topological properties of a pattern such as adjacency relationships among entities, orientation relationships among entities, and the types of entities involved. The consequence of a rule is usually identification of a feature such as a slot, pocket, or hole. A generic rule looks like

$$(rule < rule\#> (if (< condition\ -1 >) (< condition\ -2 >) \ldots (< condition\ -N >)) (then (<action-1>) (<action-2>) \ldots (<action-N>)))$$

In the preceding, action could represent a recognized primitive feature given that all the required (geometric and topological) conditions are satisfied. The inference mechanism controls how these rules are applied to the input data and may employ forward chaining, backward chaining, or opportunistic rule firing [Henderson 1984]. The working memory contains the intermediate inference results from firing rules and the features that have been recognized at any time. These types of systems are usually implemented in a logic programming language or a production system that directly supports logical inference, such as Prolog or OPS5. Henderson [1984], Kung [1984], and Hummel [1989] are examples in this category.

Henderson [1984] uses logic programming rules to extract swept subtractive features such as cylindrical holes, pockets, and slots. With this method, a 3D model in boundary representation is converted into facts in PROLOG. These

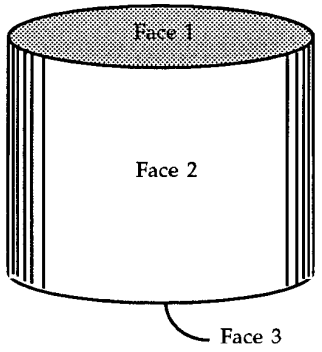


**Figure 12.** Example feature (cylindrical hole) and its associated rule.

facts are used by production rules that encode the necessary and sufficient conditions for a feature. An example rule describing a cylindrical hole feature shown in Figure 12 may look like

*IF* *A face of type planar exists which is a hole entrance face,*
*and the face adjacent to the entrance face is of cylindrical type,*
*and the face is convex,*
*and the next adjacent face is of planar type,*
*and this planar face is only adjacent to the cylindrical face,*
*THEN* *the entrance face, the cylindrical face, and the planar face comprise a cylindrical hole.*

To implement this in PROLOG, these rules are then formalized using predicate calculus (predicates and terms) as shown:

Cylindrical-hole (set-of-faces):-
entrance-face (face1),
Adjacent (face1, face2)
Cylindrical (face2)
Convex (face2)
Adjacent (face2, face3)
Not-equal (face3, face1),
Plane (face3)
Adjacent-faces (face3, face1)
Set-of-faces (face1, face2, face3).

When this rule is applied to the input B-Rep file, it returns a set of faces that comprise a feature if the execution of the rule is successful. Similarly, rules may be defined for other features, whether cylindrical or polyhedral. Henderson's method begins with subtracting the solid model of the desired part from a stock, generating the volume corresponding to the material to be removed. Feature recognition then is performed on this removed material volume by applying rules. When a feature is recognized, the machined volume for the feature is generated and subtracted from the original removal volume. The concept of an entrance face in the rules allows extra information about machining operations to be derived. Specifically, the accessibility (the orientation in which a feature can be machined, or "accessed") can be derived for each feature recognized using the entrance face for the feature. The recognized features are then bound into a feature graph in an order based on these accessibilities that is convenient for machining operations.

Kung [1984] developed a similar system that converts B-Reps into symbolic facts describing geometric and topological properties. The boundary faces of the objects are decomposed into edge loops and ultimately into items called F-faces, which are surfaces that can be formed by a single machining operation. Using rules, these surfaces are then classified as being cylindrical, noncylindrical, basic, or secondary. Further recognition can be performed using this classification. Two hierarchies of feature pattern rules are implemented, one for cylindrical features and one for polyhedral features. The rules for the cylindrical features and polyhedral features are applied sequentially and in the decreasing order of complexity in each case. This system does not consider feature interactions at all, but it does demonstrate that cylindrical features can be

recognized just as easily as polyhedral features.

Dong and Wozny [1988] propose a rule-based feature recognition system that utilizes the concept of frames. Frames are data structures containing multiple slot/value pairs. In the context of this work, frames are created for both the part representation and the feature representation. For example, the frame for a part consists of slots (and appropriate values) for the number of faces and edges in the part, the material of the part, and frames for each face in the part. The frames for the faces in turn consist of the number of edges in the face, as well as the type of face (planar, etc.), and a normal to the face. Feature frames consist of slots for width and height of the feature, as well as the geometry and topology that define the feature. The recognition is then performed by exhaustively searching the part frame for matches to all instances of feature frames.

PART (Planning of Activities, Resources and Technology) is a commercial process-planning system originally developed at the University of Twente [van Houten et al. 1989]. PART incorporates a rule-based feature recognition system. The patterns to be identified as features are specified in a feature description language. Feature recognition was two phases, feature pattern recognition and parameter extraction. The feature patterns are recognized and compared by using a set of predefined functions (rules). These functions operate on lists of geometric entities such as faces or edges. A check for subsuming and equal features eliminates redundant features. Parameter extraction obtains the position, orientation, and dimensions of the features. Adjacent features may be combined to form compound features for a hierarchy of features.

Although the rule-based method for feature representation is simple and successful for isolated features and simple interacting features, it has some limitations. First, it is verbose: to represent a simple feature like a slot, we may need up to 15 statements. Second, it is impossible to encode all properties about all the different occurrences of a feature in a rule set since feature characteristics are nonunique. It is possible that one rule representation may correspond to more than one feature. Rule-based systems look for exact matches with rule preconditions; to recognize all instances we may need rules for every conceivable pair of interactions between features and/or every configuration of an abstract feature. Due to this enumerative nature, with $n$ feature types, considering only pairwise interactions the recognition algorithm would at best be $O(n^2)$, since $n^2$ cases have to be considered.

Although syntactic pattern recognition is the formalization of pattern rules using pattern grammars, the rule-based method uses the rules themselves in place of the formalized language to prevent the limitations that grammar primitives impose. Since no grammar or primitives are required in a rule-based method, any definable feature concept, whether 2D or 3D, can be described by rules.

4.1.4 *Convex-Hull Techniques.* Unlike other feature-recognition methods introduced earlier, which rely on properties of surface entities for feature recognition, convex-hull techniques use volumetric properties to extract features from solid models. The technique is based on the idea of finding the materials that must be removed from a raw stock to produce a part. Instead of relying on pattern matching like the techniques previously mentioned, this approach exploits convex hull algorithms and Boolean operations for feature analysis. The convex hull is the smallest convex set that contains the polyhedral object. The recognition is attained by decomposing the object in stages as the (regularized) set difference from its convex hull.

To reflect the nature of this decomposition method, Woo [1982] called the convex decompositions alternating sum

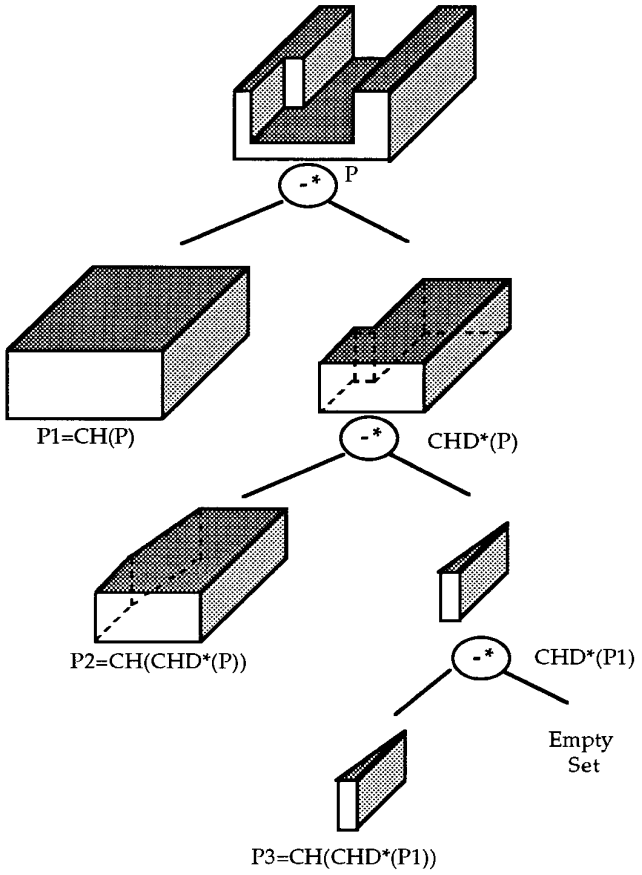


**Figure 13.** ASV decomposition of polyhedral object $P$.

of volumes (ASV). ASV decomposition represents an object by a series of convex volumes with alternating signs (for volume addition and subtraction). ASV decomposition works by first constructing a convex hull for the given object and then finding the regularized set difference between the object and its convex hull. If this set difference is empty, the ASV returns the object as a single convex object and terminates; otherwise, the object is partitioned into connected components and each connected component represents its *deficiency*. ASV decomposition is then reapplied recursively to each deficiency until the resulting set differences for all deficiencies are empty. Figure 13 shows the ASV decomposition for a polyhedral part with a through slot and a blind slot. Here $P$ stands for the object, $CH(P)$ is the convex hull of $P$, and $CHD^*(P)$ represents the regularized convex-hull difference (deficiency).

Using the ASV technique, a nonconvex polyhedron can be represented by a hierarchical tree (similar to a CSG tree) whose root node represents the object, intermediate nodes represent the union and set difference operators, and leaf nodes represent the primitive convex polyhedral sets or an empty set. There are certain major issues in convex-hull based techniques. The first is nonconvergence: ASV decomposition does not always terminate, which limits the domain of geometric objects that ASV decomposition can handle. The ASV de-

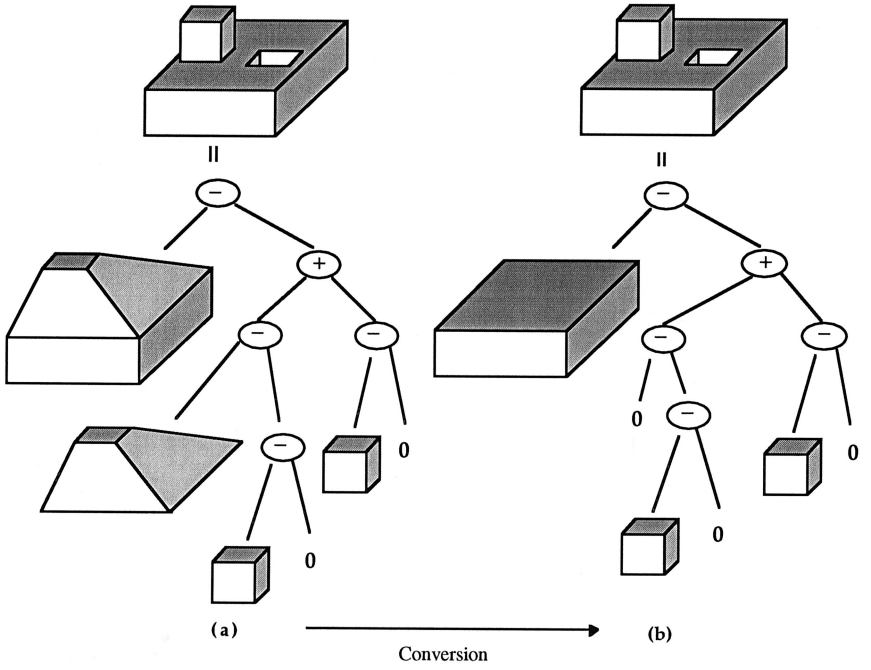


**Figure 14.** Conversion of ASV components (a) into features (b).

composition does not terminate when the convex hull of one volume in the decomposition is equal to the convex hull of that same volume's deficiency. Intuitively, such a situation creates a cycle in the ASV decomposition. That is, the same volume occurs over and over again within the ASV tree. Another issue is conversion from alternating sum-of-volumes to features. Although the components in the hierarchical ASV tree represent convex hulls and set-differences, these convex sets may not represent features in general. Conversion into features is therefore required, as illustrated in Figure 14. The research by Woo et al., discussed in the following, represents the major work in this field.

Woo's [1983] work is the earliest effort in this category. In this approach, the ASV of an object $\Omega$ can be formally defined as

$$\Omega = \sum (-1)^i H_i,$$

where $H_i$ represents each convex hull at different stages of decomposition.

No knowledge concerning the types of the volumes or their classification is produced. Human interaction is therefore required to guide the manufacturing of each convex hull component. The difficulties with this earliest effort include its limitation to convex parts because of lack of guaranteed convergence for general nonconvex parts, or of conversion of the generated volume components into features for subsequent machining. Tang and Woo [1991a, b] also propose an approach to reducing the issue of computational complexity involved in the ASV decomposition and describe an algorithm that checks if a given object will have a convergent ASV decomposition. For the computational complexity issue, they describe a linear space and an $O(n \log N)$-time algorithm and data structure for computing the difference between an object and its convex hull. An $O(N2 \log N)$ algorithm that determines if a part has a convergent ASV decomposition is also developed. This algorithm utilizes the algorithm for finding differences between

objects and their convex hulls to check whether any given stage of the ASV decomposition is nonconvergent. It then must check each such stage.

Kim [1990, 1992, 1993], Kim and Roe [1992], and Waco and Kim [1994a, b, 1993] are the latest efforts in convex hull techniques for feature recognition: they solve the nonconvergence and component volume conversion problems and therefore expand the application scope to concave parts. To remedy the nonconvergence problem, they propose a modified version of the convex decomposition algorithm called alternating sum of volumes with partitioning (ASVP). The ASVP decomposition begins with the steps for ASV decomposition until nonconvergence is detected. Once nonconvergence is detected, remedial partitioning is applied. Then, the ASV decomposition steps are reapplied to each partitioned component. Figure 14(a) illustrates the ASVP algorithm for a simple object.

Using the ASVP decomposition, they propose a novel approach to identifying and extracting volumetric features from polyhedral objects: (i) ASVP decomposition is applied to the boundary representation of the given object to obtain the alternating sum of volumes; (ii) the ASVP volumes are then converted into feature entities by various combinations of these volumes; (iii) recognized features are then classified into generic groups based on volume contribution and local accessibility information using the normal vectors of the original faces of each component. The various combination operations used to convert ASVP components into features include combination of volumes with opposite volume contributions and combining components with positive volume contributions. Since the resulting features may have both positive and negative components for machining applications, positive features are subsequently converted to negative features (which correspond to volumes to be removed from the part). This task is achieved by rewriting the Boolean expression of the positive components.

In summary, the convex-hull approach can handle most parts with interacting features by finding the alternating sum of volume decomposition. However, when features interact, certain common volumes may be shared by more than one feature, but the decomposition algorithm allocates the common volume to only one of the involved features. This problem may be addressed either by using feature-growing techniques or by recognizing features through hints. Also, it should be noted that it is difficult to construct convex hulls for curved objects. Menon and Kim [1994] suggest that local rounding operations such as fillets can be separately recognized and removed from the B-Rep, and the remaining features can be found by subsequently applying ASVP to the approximate part. Finally, after the gross features are identified, the local rounding operations can be restored. In terms of cost, the computational complexity of the approach is at least as great as that of computing convex hulls and performing set-difference operations.

4.1.5 *Cell-Based Techniques.* Cell-based techniques are similar to convex hull techniques in that both decompose the volume of the part. Moreover, cell-based techniques are also concerned with decomposing the volume of the part that is to be removed (called the delta volume, or -volume) into smaller volumes. These volumes (or cells) then either need to be directly recognized as features or, as in ASV decomposition, need to be mapped (usually through recombination) to features.

Tseng and Joshi [1994] give an example of some recent work with cell-based feature recognition. First the volume to be machined is identified and decomposed into basic removable blocks (cells) by extending all the bounding faces and considering all possible intersections. This intuitively "cuts" the -volume into pieces according to its own half-spaces.

The next step involves the reconstruction of these cells into feature volumes. Through 1D, 2D, and 3D cell connections a maximal connected feature block is created. Next, in order to classify the feature block, the nonsolid virtual faces are identified and deleted from the attributed adjacency graph (AAG) of the feature block. The classification of the feature block is then performed via matching of the resultant AAG through a simple rule-based algorithm. One advantage of this approach is that by recombining the cells in different orders it is possible to generate all the possible alternative interpretations of a part. The most prevalent drawback is that the faces of the delta volume (i.e., the faces used in the decomposition into cells) must sufficiently divide the delta volume itself. Therefore, the delta volume must be polyhedral, and if the delta volume of a part is convex, the cell decomposition will simply be one large cell.

Several researchers [Sakurai and Chin, 1993; Sakurai 1994, 1995; Dave and Sakurai, 1995] also utilize cell-based techniques for feature recognition, extending the work of Tseng and addressing some of its weaknesses. Their work also begins by decomposing the delta volume of a part. However, they allow for curved faces to be recognized by matching cylindrical faces to portions of the curved faces. After the decomposition, these methods must also recombine the cells into features. The approach taken is first to combine the cells into *maximal cells*, which are sets of the cells that obey some proximity and adjacency specifications. These maximal cells are then used to find the features of the part: they are subtracted from each other to produce features. It should be noted that these methods are also well suited to generating alternate interpretations of a part by subtracting the maximal cells in different orders.

4.1.6 *Evidence-Based Reasoning Approaches.* The feature recognition process can be realized by a fundamental AI paradigm: generate-and-test, where the generate task is realized through pattern matching and the test task is accomplished through feature validation. Generation is a process of finding hypotheses about possible features using pattern-recognition techniques. Validation, on the other hand, is a process in which hypothesized features are verified based on additional geometric or topological constraints. Features not passing the validation test are rejected. This verification is a necessary step since feature hypotheses may represent a feature that is combinatorially possible but cannot be realized physically on the original object due to physical and geometric constraints. A common method for verification is to use an expert system in which rules are used to represent features and constraint knowledge and each feature hypothesis is checked against the rules to see whether any rules have been violated.

Marefat and Kashyap [1990] and Vandenbrande and Requicha [1990, 1993] used this generate-and-test strategy. In Marefat and Kashyap [1990], knowledge is represented by graphs, the recognition process is guided by an hypothesis generate-and-test process, and verification is realized through a rule-based expert system, where rules are used for representation of features and constraint knowledge.

The method proposed in Vandenbrande [1990]; Vandenbrande and Requicha [1990, 1993]; and Requicha [1996] represents considerable progress in recognition of interacting features using a rule-based approach for generating hints. In this approach the algorithm searches for hints and incomplete features in the first stage. Production rules generate hints on the presence of a feature. An incompletely specified feature is associated with each hint. A hint about a feature may be generated by a characteristic combination of part faces, by a design feature from which a manufacturing feature can be inferred, or by certain manufacturing properties such as dimensions that are associated with

a particular feature type. For example, a hint for a slot may be produced from two parallel planar faces with opposing normals, and a hint for a hole may be produced from a thread attribute. Searching for feature hints rather than complete feature descriptions enables the approach to identify interacting features. The reason is that when features interact, their topological properties may change, and hence they may not be recognized as unique complete descriptions, but the existing patterns and attributes can still offer many hints about the feature's existence. The recognizer uses a rule-based approach for generating clues or hints about potential features from information gathered from several sources including nominal and tolerance geometry, attributes (e.g., thread), and functional features specified by a designer. Hints are subsequently validated using geometric testing based on criteria closely associated with machinability constraints.

Feature hints are passed to a classifier that categorizes them into three groups: promising, unpromising, and rejected. Rejected hints are discarded. Unpromising hints are stored in a *blackboard* and all activities on them are temporarily suspended. A blackboard model prescribes the organization of the domain knowledge and all the input, intermediate, and partial solutions needed to solve the problem [Nii 1986]. Pieces of knowledge are applied at the most opportune time, resulting in incremental generation of partial solutions. The unpromising hints may be reactivated later upon receiving further information. Promising feature hints are further processed by the feature completer, which searches for all relevant data about the feature. The feature completion is achieved by growing the feature volumetrically along feature-specific directions and classifying the feature with respect to the part and the raw material. Completion can be one-dimensional, involving linear, radial, or circular extensions, or two-dimensional, where a feature cross-section is also extended laterally in a plane normal to the translational sweep axis. The completed features are stored in the blackboard, where another set of rules attempts to combine them with other features that satisfy certain preconditions. For example, two coaxial and adjacent holes are combined to form a counterbored hole, which is a type of composite hole.

Figure 15 shows the proposed architecture of the system based on hints. The feature finder operates in several stages by selectively activating sets of rules, which are applied to the available data. In the verification phase, the rules for verifying clues and resolving conflicting hints are activated. A proof-of-concept implementation is developed in a Knowledge Craft/PADL-2 testbed. Vandenbrande's method can deal effectively with interactions, since the search process does not rely on complete information.

As with the generate-and-test method previously described, evidential reasoning, or uncertainty reasoning, has been used for feature recognition. Evidential reasoning offers a consistent means for handling uncertainties. Uncertainty may arise in design due to conflicting, redundant, or missing data. When features interact, uncertainty develops as a result of the nonuniqueness of the patterns associated with the topology and geometry of features in these interactions. When a feature interacts with another feature, certain kinds of topologic and geometric changes are possible [Wang 1992].

(1) The new feature modifies the topologic properties of the existing feature, but its own properties are not changed.
(2) The topologic properties of the new feature are modified, but the patterns used in recognition of the existing feature are not changed.
(3) Both the new and the existing features are modified.

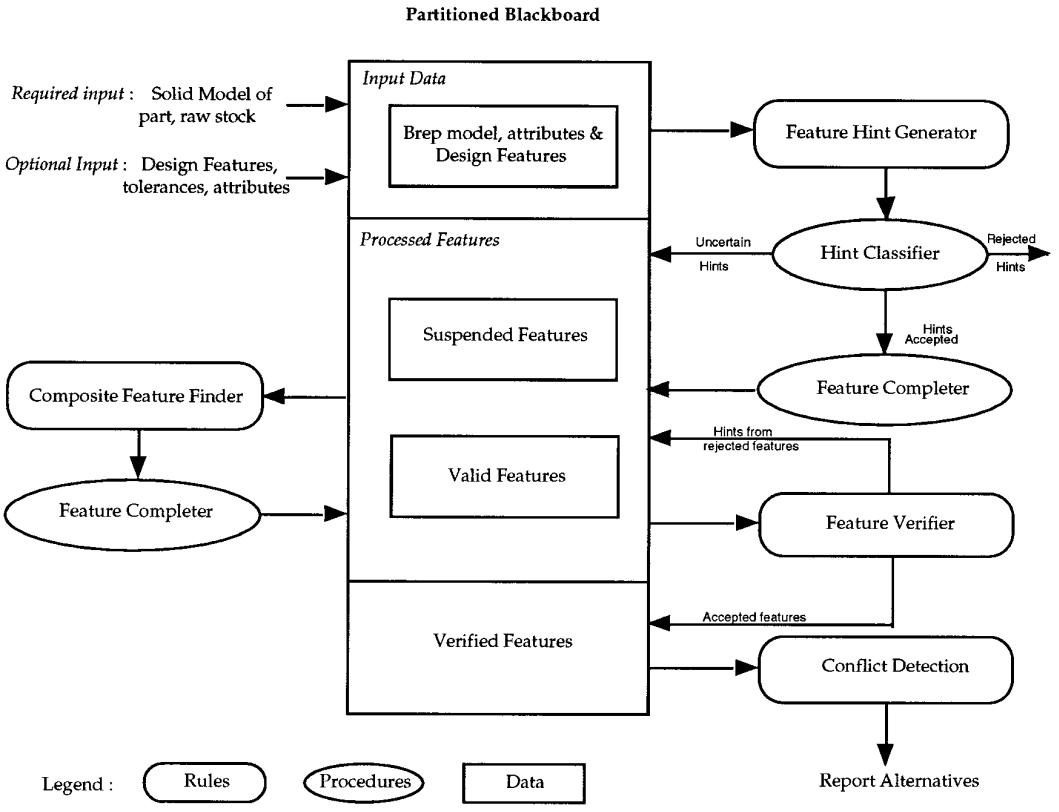


**Figure 15.** Architecture of feature finder using feature hints [Vandenbrande 1990a; Vandenbrande and Requicha 1993].

Regardless of the kind of interaction, the resulting representation of the features usually does not correspond to predefined feature patterns. Thus simple pattern matching will not be able to recognize these features. Specifically, the modified geometric or topological information may need to be recovered in order to complete the broken or changed patterns prior to recognition or matching. Uncertainty reasoning offers a formal and consistent mechanism to accomplish this goal through evidence aggregation. The idea is that even though there are many uncertainties in possible conclusions from data presented by two or more interacting features, there is also rich evidence that, if combined properly, can help in recovering the modified and/or necessary information.

With this approach, evidence supporting or rejecting the existence of instances of different features can be combined to describe depressions of a part in terms of manufacturing features. In such a framework, each piece of evidence may not by itself be sufficient for recognizing a feature, but generates a probability (or a weight), which is a measure of confidence that relates the evidence (a topological or geometric relationship) to a feature. It is the collection of evidence and consistent combination of its components that produces a description of the features involved.

The two most popular techniques for handling multiple sources of evidence are Bayesian belief accumulation [Pearl 1988] and the Dempster–Shafer theory [Shafer 1976]. In the former, the relationships between evidence and hypoth-

eses are described by three numbers: the prior probabilities, the likelihoods, and the posterior probabilities. Evidence (topological and geometric relationships) supports or disconfirms the hypotheses about the existence of different feature instances through a measure of confidence or a likelihood assignment. This evidence can then be combined using probability axioms and the Bayesian rule. In the Dempster–Shafer approach, relationships among the evidence and the hypotheses are described in terms of the belief intervals and basic probability assignments, which are the extent to which a piece of evidence contributes to the proof of an hypothesis. In this method, evidence (topological and geometric relationships supporting or not confirming the hypotheses about the existence of different feature instances) exerts its influence through a quantity called a basic probability assignment, which assigns a measure of belief to a set of supported hypotheses.

Both mechanisms offer consistent and coherent means to combine available topological and geometric evidence to overcome the difficulties associated with the nonuniqueness of features in interactions. There are several important advantages to such a scheme for extracting and identifying features. First, since features are recognized by the collection of evidence, the individual pieces of evidence need not carry equal weight, and in fact some of these may support conflicting conclusions. As long as the cumulative combination of the pieces of evidence is in favor of a correct interpretation, a correct description of the part will be obtained. Second, one need not know ahead of time which set of topological and geometric relationships, or which patterns, will be observed for each feature. In fact, at different times different topological and geometric relationships may indicate the same feature class. Third, the same topological or geometric relationship may simultaneously support two different features with different degrees of confidence.

In the field of feature recognition from CAD models, Marefat has proposed a method using uncertainty reasoning in analyzing interacting features [Marefat and Kashyap 1990]. The approach is tailored towards handling machining features and uses a new graph representation, referred to as a *cavity graph*, to represent the depressions of a part. Cavity graphs are an extension of the attributed adjacency graphs proposed in Joshi and Chang [1988] but differ from the AAGs in the following aspects. First, each node in the cavity graph is labeled with its principal normal direction, which determines the orientation of a face with respect to other faces. (Since this orientation is relative to the other faces, it establishes a relative spatial relationship between the face and other faces in the depression of the part.) Second, the cavity graph introduces the concept of unification, which uses the same node to represent two or more distinct object faces when two or more faces can be merged into a larger virtual face. Unification of faces is possible if the unified face does not interfere with the interior of the object [Marefat et al. 1990]. Finally, since depressions are usually associated with concavities, cavity graphs contain only concave links. A part may have several cavity graphs associated with its different depressions. Figure 16 shows an example part with a slot feature, the AAG and the cavity graph for this example part.

Primitive features are defined in a fashion similar to the AAG approach, but utilizing the new cavity graph. Recognition is carried out through an hypothesis generate-and-test method. Hypothesis generation is carried out by partitioning the cavity graph of the part into the subgraphs that represent the feature templates. The advantages of cavity graphs are that they represent a much smaller search space, consequently reducing the computation re-

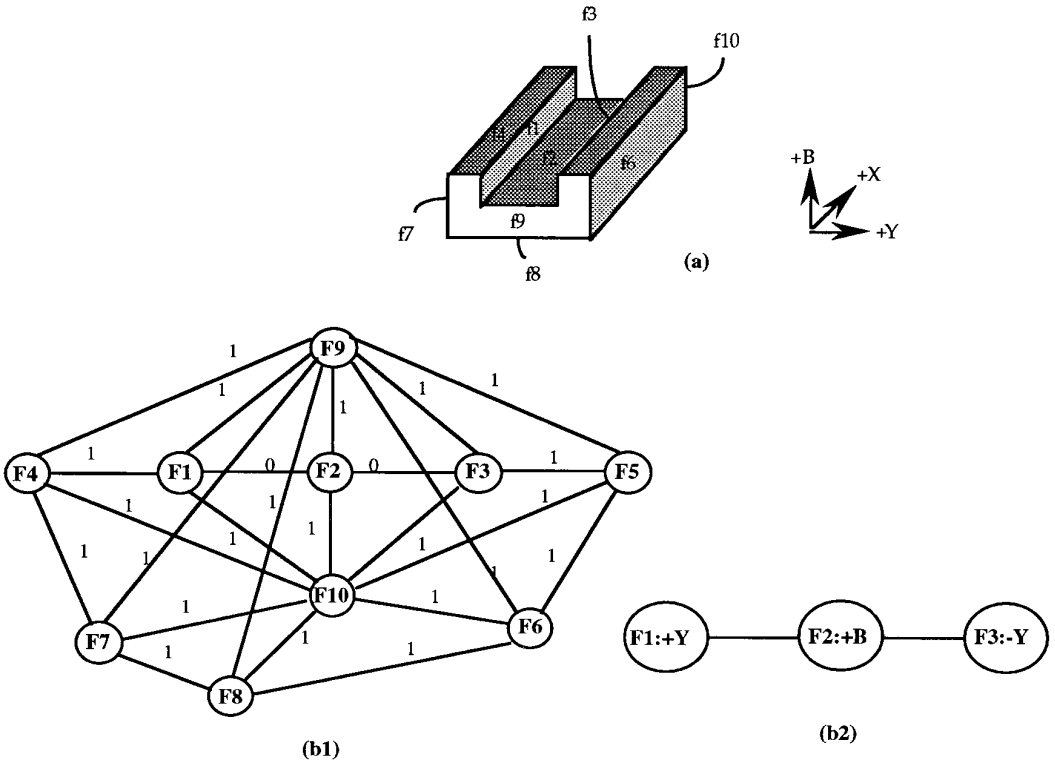


**Figure 16.** (a) Part with a slot feature; (b1) its global graph, and; (b2) its cavity graph.

quired for pattern matching, and that they carry local spatial information.

Although cavity graphs and the generate-and-test process improve the ability to extract interacting features by themselves, a better method is gained by the incorporation of an evidential reasoning technique in a graph-based context. Marefat and Kashyap [1990] formulate this problem as one of identifying the necessary and most probable virtual links (links that do not exist but need to be added to the cavity graph) for extraction of features from the partitions of the cavity graph. This approach works by combining available geometric and topological evidence about each candidate virtual link using Dempster's rule of combination. Dempster-Shafer theory is exploited to identify the most probable virtual links among all possible candidates. The results of this evidence aggregation identify virtual links that are subsequently augmented to the original cavity graph, resulting in a supergraph that can readily be partitioned to extract features.

The major contributions of this approach include its introduction of evidential reasoning for recognition of features. This mechanism uses a clustering technique to select the virtual links in its final phase. A similar approach has been developed based on Bayesian networks [Ji and Marefat 1995]. In addition, a new contribution of this method is the generation and incorporation of hierarchical feature-based evidence. No proof that the technique will always find the correct virtual link cluster was provided, but in the experiments performed, the proposed techniques always provided the correct set of virtual links.

Finally, Trika and Kashyap [1994] proposed a feature-recognition method similar to the work of Ji and Marefat

[1995] in which the notion of the cavity graph is again utilized to represent a part and features. Furthermore, virtual arcs that enable interactions of features to be handled are hypothesized and created. However, virtual arcs are created by extending faces of the part and computing intersections; that is, the faces of the part are extended and checked for pairwise intersections. When two extended faces intersect (at an edge) this edge becomes a virtual arc. Once the virtual arcs are restored, the feature recognition can occur through graph matching.

Due to the nonuniqueness in feature representations, evidential reasoning is an especially well-suited approach to feature recognition. A nonmonotonic reasoning approach such as constraint relaxation would also belong to this category. For example, in the case of a pocket not bounded on one side, a method relying strictly on pattern recognition would fail because of its inflexibility, since it expects and recognizes the pattern of a pocket as an entity with four walls. Evidential reasoning can also be advantageous in obtaining multiple interpretations of a given part in terms of different sets of features, because the same set of evidence can point to the existence of different sets of features, each set yielding an alternative interpretation.

4.1.7 *Other Approaches.* Many of the feature-recognition approaches discussed call upon different techniques and are in a sense hybrid, but they all share one underlying mechanism that categorizes them. Here we discuss several approaches that, for one reason or another, could not be placed in any of the preceding sections.

De Martino et al. [1994a] propose a feature-recognition mechanism that incorporates both syntactic pattern matching and graph-based ideas. The motivation for this combination is the need for a system that first recognizes the protrusions and depressions of the component so that features (suitable to some domain) then can be recognized by graph matching. The advantage of decomposing the problem into these steps is that the second recognition stage (the identification of specific features) can be modified for different domains of features.

In recent work, Gupta et al. [1994], Gupta and Nau [1995], Nau et al. [1993], and Regli et al. [1994] recognize a part as a collection of MRSEVs (material removal shape element volumes)—a STEP-based library of machining features. MRSEVs are volumetric features corresponding to machining operations on three-axis milling machines. From the CAD model, the algorithm constructs sets of MRSEVs occurring in alternative interpretations of the design. The different MRSEV models are then evaluated to yield the optimal interpretation (in terms of machining cost). The feature-recognition algorithm is based on the assumption that every primary valid MRSEV will contribute some face in the delta volume. That is, each face in the delta volume of a part is matched to all possible features that could create such a face. All the faces in the delta volume are considered and the set of all primary MRSEVs is constructed. The researchers argue that within a restricted domain this approach can find all of the possible interpretations of a part.

### 4.2 AFR Using CSG

The feature-recognition techniques introduced so far use B-Reps as their input information from the solid modeler. In this section, we investigate the use of another solid representation scheme, namely, constructive solid geometry (CSG). CSG is a volumetric representation in which a solid object is explicitly represented by an ordered binary tree. The leaves of the tree are instances of primitive solids. The intermediate nodes contain regularized Boolean operations including *union*, *intersection*, and *difference*. Figure 17 shows a CSG representation for a part.

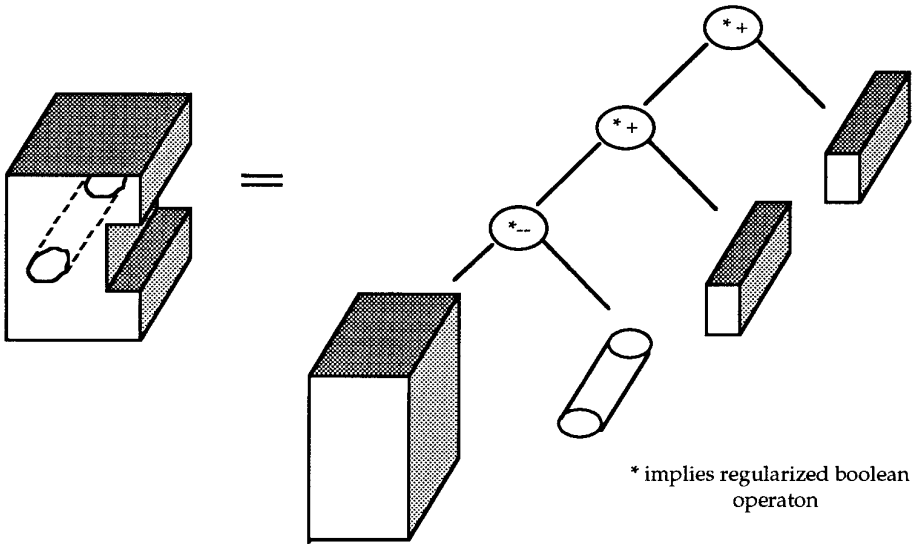


**Figure 17.** CSG representation for an example part.

A CSG representation is simple and unambiguous. It is a useful paradigm in interactive design, but its representation is not unique. This is the major problem encountered when using CSG for feature recognition and other applications. By nonuniqueness, it is meant that one solid object may admit several different valid CSG representations. Figure 18 shows the same part as in Figure 17, but with a distinct CSG representation. The nonuniqueness is caused by the fact that any Boolean expression may have an infinite number of equivalent expressions and means that all the CSG nodes involved in a specific feature could, in general, be scattered around within the tree. This aspect in turn may mean that an individual CSG primitive, such as a parallelepiped block, a sphere, or a triangular prism, is too general and ephemeral to be equated to a feature. In addition, the difference operation in the CSG tree may not always correspond to a manufacturing material-removal operation. On the other hand, certain removal operations may be implicit without using any difference operation. For example, the union of two partially overlapped coaxial cylinders of different radii would represent a cylindrical part after a lathe operation [Lee and Fu 1987]. Due to these difficulties, attempts to use CSG representation for automatic feature recognition begin by converting them to a simpler representation.

Lee proposed an approach that converts a CSG representation into a similar, but unique, tree representation by identifying the different nodes in the CSG belonging to the same feature and merging them into one node. New nodes are constructed such that each uniquely represents a machining feature [Lee and Fu 1987]. The proposed approach consists of two steps, namely, feature identification and unification. Feature identification is used to identify all primitive nodes in a CSG that may belong to one feature. Unification is used to aggregate these identified nodes into one node, transferring a specific feature into a desirable and isolated representation. Unification not only reduces the degree of nonuniqueness of the CSG tree, but also allows manufacturing attributes to be associated with meaningful geometric volumes. Figure 19 shows the schematic diagram for this method.

Feature identification is accomplished through pattern matching. Features are

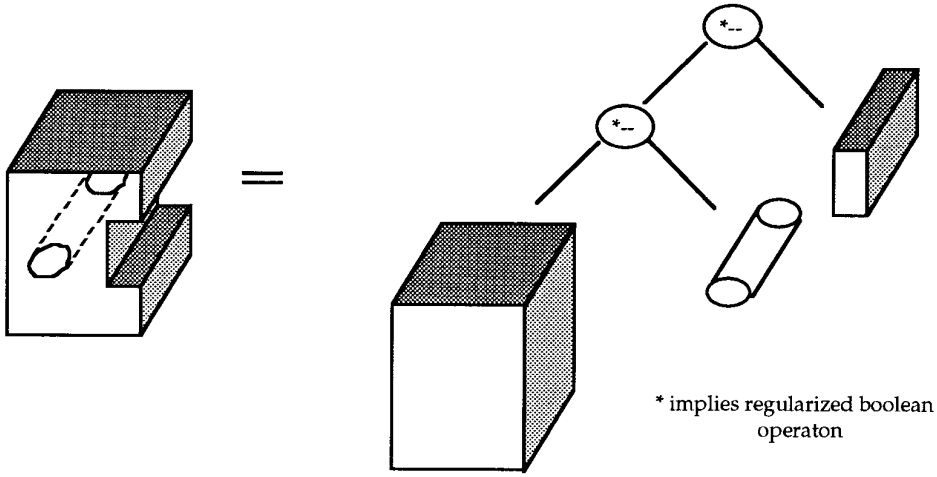


**Figure 18.** Nonuniqueness of the CSG representation.

modeled as patterns of volumetric primitives that satisfy certain geometric constraints, especially constraints for facial and axial alignment. The rationale behind classifying features through their principal axes is that each feature class has a unique type of axis. Here the type of axis (or principal axis) is defined by the coordinate frame it is within, its orientation, origin, and length. For example, features like cones, cylinders, and tori can all be characterized by a single axis, features like spheres have axes with arbitrary orientation, and features like cubes, due to their asymmetry, have 12 axes. During the feature recognition process, CSG primitives are first partitioned into equivalent classes according to the orientations of their axes, each consisting of members parallel to each other. Within each equivalence class, axes involved in a particular feature can be located according to the conditions defined by the feature. This leads to the recognition of specific features within each equivalence class.

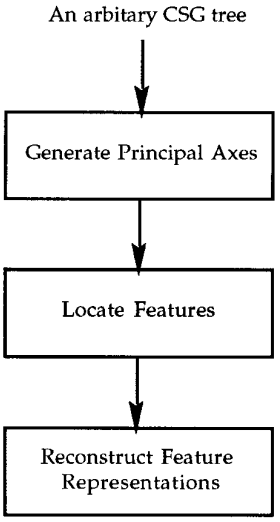


**Figure 19.** Block diagram for extracting features from CSG input.

Once all the features have been identified, unification can be initiated. During this phase, the CSG tree is rebuilt by merging or grouping equivalent CSG nodes into one node to represent a unique manufacturing feature. Unification consists of these steps:

(1) tree reconstruction in which all participating nodes of the feature are relocated and grouped to form a subtree, and
(2) tree transformation in which the subtree resulting from the first step is replaced by an equivalent subtree.

Once all feature nodes have been unified, the feature subtrees are processed

bottom-up and replaced with simplified subtrees or primitives that directly correspond to features.

Recently, Perng et al. [1990] also developed an algorithm for automatic feature recognition from the CSG tree representation. In this technique, the CSG representation is first transformed into a different hierarchical volumetric representation called destructive solid geometry (DSG). From the DSG representation, the required object surface information is derived in terms of a boundary representation. The purpose of this transformation is to overcome the problem that CSG tree primitives cannot be used directly by an application. The transformation is characterized as follows.

Input:

The CSG tree can be expressed in the "inorder" form as

$$P = P_0 \pm P_1 \pm \ \ldots \ \pm P_n,$$

where the symbols $\pm$ are defined as geometric binary union or difference operations, $P$ is the part, and $P_i$ are the CSG primitives.

Output:

$$P = S - E_1 - E_2 - E_3 \ldots - E_m,$$

where $S$ is the raw stock material and $E_i$ are the excess material volumes contained in $S$.

The constraints for the transformation are that every $E_i$ must belong to one of the defined feature primitives and all $E_i$ must be disjoint. The transformation converts the union operations in the CSG into difference-only operations and the CSG primitives into disjoint removable volumes. Figure 20 illustrates this transformation process, with Figure 20(a) showing the part, Figure 20(b) showing the CSG representation, and Figure 20(c) showing the DSG representation. The DSG is suitable for automatic feature recognition, since the primitives are disjoint removal volumes, which correspond to some manufacturing operation.

To recognize features from the DSG representation, features are first classified hierarchically for pattern-matching convenience. The recognized features are subsequently associated with feature attributes including dimensions and possible tooling entrance faces. A limitation of this approach is that the technique cannot handle the CSG constructs or primitives that result from intersection operation(s).

Shpitalni and Fischer [1994] propose a recognition scheme for machining "regions" from CSG input. In this work machining regions are defined as groups of features that all interact with one another, but not with other regions. The recognition approach is similar to the preceding approaches in that it first converts the CSG representation into a more consistent representation. In this work, the intermediate representation is a positive CSG tree in which only (regularized Boolean) union and intersection operations are allowed. Subtraction operations in the original CSG tree are represented by negation and intersection operations. Once the input is in positive CSG tree format, the recognition process recursively divides space into quadrants until each quadrant is either full or empty of material in the part. Then each quadrant is labeled as a pocket or island (depending on its being empty or full), and finally these labeled quadrants are recombined to give the machining regions of the part. The recombination strategy searches through the CSG nodes within the labeled quadrants to group related volumes into larger regions. Of course, this approach actually extracts machining features only if the machining regions of a part are identically the machining features, which occurs when there are no feature interactions.

Another approach to feature recognition using CSG is that of Parry-Barwick and Bower [1995], which provides a novel technique for feature recognition using set theory. The approach offers important advantages. First, it allows

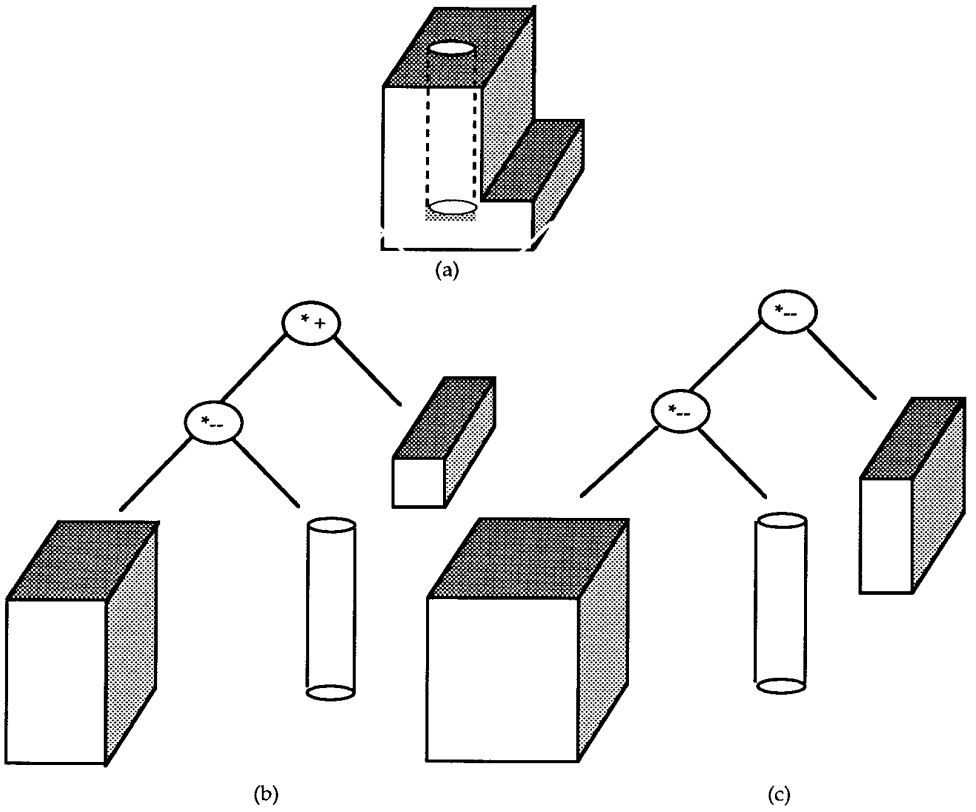


**Figure 20.** Example of converting CSG into DSG representation: (a) the part; (b) CSG representation; and (c) DSG representation.

for user-defined features. This approach essentially performs template matching with a variable template on a specific solid model. The advantage of such flexibility is the expanded domain to which the feature recognizer can be applied. Second, the approach allows for approximate matching of the templates; that is, it does not require an exact template match, but rather can return a feature that is a slight deviation from the input template. The mechanism of this approach is given the template to match (which is encoded as a set, or volume) and the solid model input (a set of volumes (CSG)), the template is "moved" across the model and checked for matches. The movement is accomplished by translating one (or more) variables in the space of the model. For example, to match a 3D template to a solid model, the method would have to translate the position of the template, the rotation of the template, and the scale of the template (in all a 10-dimensional problem). The method described in this work is applied only to 2D models although it is not theoretically limited to any dimension.

In summary, although an important motivation for CSG development and use has been that it could more closely represent machining operations, it has received less attention for feature identification and recognition than those utilizing B-Rep input. The major difficulties have been the nonuniqueness of the representation and the lack of a general relationship among the primitives involved in the construction of a CSG and the features of the resulting design.

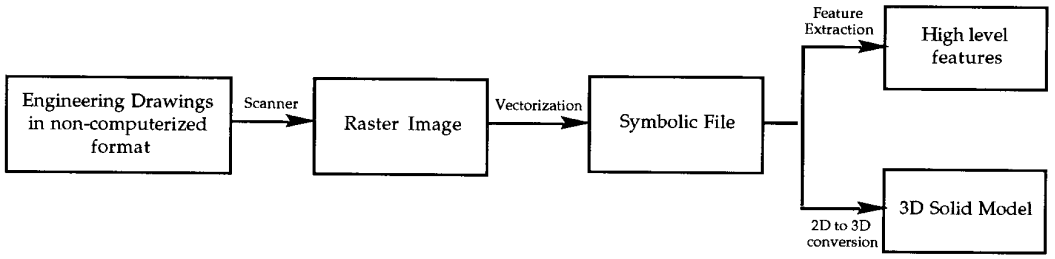


**Figure 21.** Schematic approach for identifying and extracting features from engineering drawings, or converting them into solid models.

### 4.3 AFR from 2D Engineering Drawings

Most research in feature recognition uses solid models as input, since it is believed that solid model representations contain more information. Another motivation for the use of solid models is that new generations of CAD systems provide this information directly. As a result, comparatively little attention has been paid to automatic feature recognition from 2D input (engineering drawings and sketches), even though engineering drawings are still frequently used to document design information. An engineering drawing here is a three-view representation in which front, side, and top views are given from an orthographic projection, whereas sketches are a 2D drawing of a 3D part.

CAD-related research on engineering drawing has focused on the following aspects. The first is the conversion of engineering drawings on paper into a proper computerized representation, such as a symbolic file [Vaxiviere 1992]. This conversion is usually accomplished through algorithms that convert raster images from a scanner into CAD-readable vector representations in terms of lines and curves. This research is a visual recognition process involving scanning and preprocessing, vectorization, and post-processing. Although engineering drawings in a wireframe representation file are suitable for tasks such as documentation, wireframe representations are not conducive to applications such as automatic process plan generation because all entities in these files are in terms of low-level entities such as lines, edges, or faces.

The second aspect of CAD-related research on engineering drawings has been to extract high-level features from computerized engineering drawings. Some researchers have also attempted to convert the symbolic 2D representations obtained into 3D solid model representations [Aldefeld 1983; Dori and Tombre 1995]. Figure 21 illustrates the schematic diagram of CAD-related research based on engineering drawings. Markowsky and Wesley [1986] is a good survey of this area. In the following, we focus on feature recognition from engineering drawings.

As in research on understanding solid model representations, the underlying philosophy for extracting features from engineering drawings has been to view a complex part as being composed of elementary primitives (features) belonging to a set of predefined classes and to recognize these elementary primitives by making use of the knowledge about the class-dependent patterns of the 2D representations. Research in this field usually makes the assumption that the three-orthogonal-view representation is unambiguous; that is, one drawing corresponds to one and only one unique object. This assumption may not always be true.

Recently, Meeran and Pratt [1993] introduced a method implemented in PROLOG for the extraction of features from engineering drawings. This method is based on the observation that

most simple manufacturing features, such as pockets and slots, are manifested in at least one view by closed loops of constructional entities, using line and circular arcs. For example, a cylindrical boss feature may contain a circle (a closed loop) in the top view and two rectangles in the front and side views, respectively. The identification of these loops through pattern matching and subsequent combination of the recognized entities from the three views are the basis for this feature recognition method. To recognize closed loops, the recognizer is provided with a library of patterns, represented in production rules, that correspond to a range of isolated features. The recognition process consists of the following steps. First, all simple isolated features are recognized through pattern matching. The corresponding entities for the recognized isolated features in the three views are subsequently removed from the original views. Next, the remaining composite entities are examined using logic-based production rules to see whether they belong to interacting features. Those entities belonging to interacting features are then isolated for further processing. In the third step, the remaining composite entities are gathered and grouped under the heading of generalized protrusion or depression features.

Recognizing an isolated feature is straightforward. An individual feature is recognized in terms of its pattern of edges in the drawings of the three orthogonal views. However, for the feature to be recognized, certain correspondences must be established among patterns in different views. Given the same machined entity in each view and the correspondences among the matched entities, an isolated feature is subsequently recognized. The isolated features that can be recognized through this algorithm include cylindrical holes, slots, pockets, grooves, and threads.

To recognize interacting features, the approach must be able to deal with incomplete patterns and infer the embedded features based on this incomplete information. The general logic for dealing with feature interactions is complex, especially in the cases in which more than two features interact with one another. The logic used in this research includes two steps: determining the lines and arcs that terminate at a previously identified object, and extending these lines to determine if their extension ends within the pattern. If so, we assume that the previously identified object disrupts the current object, whose complete outline can be recovered by extending all terminal points. This simple logic can only deal with simple two-feature interactions. Specifically, it is not appropriate for an interaction in which both features are disrupted.

Earlier research work in this field includes Burn [1991], which derives feature data from drawings and uses them in the construction of a CSG solid model of a part, and Perng [1988], which develops an algorithm for automatic 3D machining feature recognition. Other research has tackled the feature-recognition problem for rotational parts. This is an easier problem since in essence, given the axis of rotation, it can be reduced to a 2D problem.

Research on automatic reconstruction of a 3D solid model from engineering drawings is a bottom-up interpretation process. Objects are represented hierarchically in a network (such as a semantic net), with the top of the network representing 3D objects and the bottom representing 2D primitives. The inference process works up through the hierarchy by first grouping the 2D primitives to infer primitives at a higher level, which are in turn used to infer features at even higher levels, continuing up through to the top of the hierarchy for the entire object. This bottom-up inference process cannot always be successful in the absence of auxiliary information. It is also vulnerable to computational explosion due to the huge search space and the exhaustive nature of the search. To reduce the search space, heuristic information is employed to help formulate an evaluation function

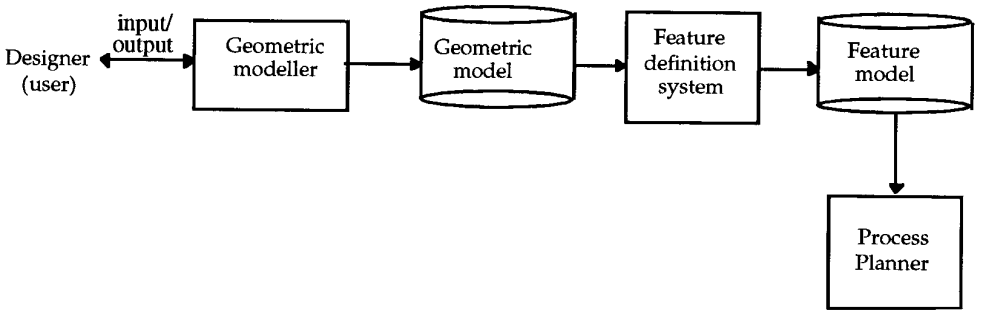


**Figure 22.** Human-assisted feature definition.

for a best first search algorithm. In addition, a model-guided search is also used by some researchers [Aldefeld 1983] to further reduce the search space and to validate the hypotheses.

To summarize, the major difficulty encountered using engineering drawings for feature recognition or 3D solid model construction is the ambiguous nature of the underlying representation. This aspect is reflected by the fact that 2D data may not contain sufficient semantic information to uniquely identify an object. The assumption that three orthographic views can unambiguously define a unique object may therefore not be realistic in real-world applications, and has limited most developed techniques to application to polyhedral parts because of inadequate facilities for dealing with general shapes.

Finally, engineering drawings are not the only kind of 2D input possible for feature recognition. Hwang and Ullman [1994] describe a system to recognize features from 2D freehand sketches. The input to this system is a 2D sketch of a 3D part. The overall system allows the user to interactively sketch a part, then have the feature recognized from this part. The preceding functionality is broken down into the three subsystems: a 2D freehand sketching subsystem, a feature-recognition subsystem, and a spatial reasoning subsystem. The sketching subsystem is responsible for the user input. The feature-recognition subsystem works through a rule-based system by matching the user's sketches (line segments, arcs, ellipses) to 3D features. The spatial reasoning subsystem incorporates new features being drawn to include the previously recognized features found in the part.

## 5. OTHER PARADIGMS FOR OBTAINING FEATURE INFORMATION

In addition to using automatic feature recognition to automate the link between CAD geometric models and process planning, two other paradigms for integrating design and manufacturing process planning have been studied by researchers. The idea has been either to incorporate features into the CAD data during the design process or to devise systems that recognize features with interactive support from a user. The paradigms for these approaches have been respectively called [Shah 1991]:

—design by features (or feature-based design), and
—human-assisted feature recognition from geometric models.

Human-assisted feature definition provides a way for a qualified manufacturing engineer to interactively pick and group topological entities (edges, faces) from an image of the part to define a feature. Figure 22 [Shah 1991] shows an overview of this approach. One disadvantage of this approach is that it does not provide for automatic transfer of information between design and pro-

duction. Its advantage is that it allows the designer to design a part in whatever way is most convenient. Prototype systems for this purpose have been built at General Motors Research Laboratories and at the National Bureau of Standards [Brown and Ray 1987].

In an alternative paradigm, feature-based design (or design by features), primitive features are incorporated during the design stage (see Rosen [1993] for a survey of recent work in feature-based design). The main component of feature-based design is that it allows users to design by adding, subtracting, and manipulating instances of generic features.

It may be recalled from the discussion at the beginning of this article that features are application-specific and domain-dependent. For this reason, features usually are suitable either for a design itself or for an application domain (such as manufacturing). If the features used during design are specific to design (design features), it will be necessary later to convert them to manufacturing (or application-specific) features so that a manufacturing application such as a process planner can perform its tasks.

Certain feature-based design systems support inheritance properties in feature families, thus enhancing the design environment. Systems for this purpose have been built for designing injection-molded parts [Vaghul and Dixon 1985], aluminum castings [Vaghul and Dixon 1985], and machined parts [Chang et al. 1988].

Although feature-based design can partially integrate part design and process planning, it has posed certain problems [Karinthi and Nau 1992]. First, it may require extra knowledge on the part of the designer since manufacturing and assembly requirements may need to be considered, especially if the features used during design are application-oriented. This is a task that most designers are not qualified to undertake. Second, the approach traditionally does not address multiple feature interpretations that may exist for the same component. This is an important issue related to alternative strategies to produce a given design. Finally, in feature-based design, as mentioned earlier, features are dependent upon a given application. This implies that even if features are incorporated into the CAD model it may become necessary to reinterpret the part with a different set of features.

Recent work has attempted to address the preceding difficulties of feature-based design in one of two ways: feature-mapping techniques have been developed to take a feature-based design and map its features to the features of other domains; and generic/flexible feature-definition strategies have been incorporated into feature-based design. The problem of handling multiple feature interpretations within feature-based design has been approached by integrating AFR techniques with feature-based design. In the following section, we discuss the techniques developed to solve the problems in feature-based design.

### 5.1 Feature-Based Design and Feature Recognition

Feature-based design has sometimes been regarded as an alternative to feature recognition. Instead of simply searching the CAD geometric models for features, feature-based design incorporates the features directly into the CAD description. However, several difficulties have been encountered. First, feature-based design does not provide a mechanism for alternative feature descriptions of a part. That is, when a designer draws a part using feature-based design, feature interactions can result in the existence of new, alternative features within the part. These features may or may not have been intentionally introduced by the designer, but in either case, they may not exist within the CAD model. Second, the features used for design may not correspond to the features used for other applications

(e.g., manufacturing). Moreover, if the designer is required to work with features specifically suited to an application domain, say manufacturing, this may require too much knowledge on the part of the designer or impose too many design limitations. Nevertheless, design for manufacture is quite popular now, in view of the lack of robust commercial feature-recognition systems.

There has been work that addresses these shortcomings but does not use feature recognition. For example, some recent work in feature-based design has allowed a user-definable set of features or has helped in the maintenance of manufacturing consistency.[3] This work is not surveyed in this section since it has not been concerned with feature recognition per se. This section addresses the difficulties in feature-based design that make use of feature recognition.

The general strategies for enhancing feature-based design utilizing feature recognition have been either to map the features used in the design to features in other applications or to incorporate feature-recognition strategies completely. The idea of feature mapping has already been seen in the convex-hull techniques for feature recognition. Here the goal is to start with a description of a part according to one set of features and then to map (or translate) this description into a new description using an entirely new set of features. The convex-hull techniques demonstrate feature mapping from ASV decomposition to machining feature decomposition.

Chamberlain et al. [1993] describe a system to map protrusion features to depression features within a feature-based design framework so that, during the design of a part, a designer can create both depression and protrusion features. However, only depression features roughly correspond to manufacturing features. Protrusion features, on the other hand, are not directly manufacturable (i.e., they do not directly correspond to manufacturing features) since they are produced by removing surrounding material, not material within the feature itself. The mapping process from protrusion features to depression features consists of two major elements: extending the stock of the part and recognizing depression features within this extension. That is, when a protrusion feature is called for, the stock of the part is extended to include this new protrusion. However, this extension may create material in addition to the protrusion feature, so this extra material needs to undergo a feature-recognition process.

In work integrating feature-based design and feature recognition, there has been some research in a modeling system based on incremental feature recognition [Laakko and Mäntylä 1993]. The approach combines design by features and feature recognition and gives the designer the choice of using either method. The designer's task is simplified by allowing both solid and feature modeling operations. The method also maintains consistency between the geometric model and the feature-based model. If a feature is added through feature-based design methods, not only is the geometric model updated, but new features created from the interactions of old features with this new feature are recognized. The feature recognition strategy is a graph-based approach.

Han and Requicha [1994] propose a system to incrementally recognize features in a part during the design process. This system is based on the earlier work in Vandenbrande [1990], Vandenbrande and Requicha [1990, 1993], and Requicha [1996], as described in Section 4.1.6. After each step of the feature-based design process, the method computes the new features within the part, based on the old features found in the part; the new feature is added, along

[3] Please see Duan et al. [1993], Feng and Kusiak [1995], Chen and Hoffman [1995], Ovtcharova et al. [1994], Ovtcharova and Jasnoch [1993, 1995], and Yu et al. [1992].

with the spatial interactions between these features.

De Martino et al. [1994b] provide an approach to integrating feature-based design and feature recognition. As with Laakko and Mäntylä [1993], this approach also maintains geometric and feature-based design data, but this work maintains consistency between the two by also storing an intermediate feature model. This intermediate model, which represents features in a generic format, is mapped to both the geometric model and the feature-based design model. Also as with Laakko and Mäntylä's system, when features are added through feature-based design or by changing the solid model directly, the intermediate model is altered and this information is propagated through the system. The feature-recognition strategy is rule-based in this method, which offers another advantage in that feature mapping can occur from the intermediate model to any domain-specific feature model desired. This allows the features in the feature-based design model to differ from those of the application domain.

Finally, work presented in Ko et al. [1993] and Ko and Park [1994] provides another integration of feature-based design and feature recognition. In it a new model is created, in addition to the solid model, to represent the interactions among the features already recognized (or designed) within the part. When a new feature is added (or deleted), the model checks what other features are affected (through interaction) by this new feature. Only the affected features are updated with the new change. That is, the model in this approach maintains a graph whose nodes are features and whose arcs represent dependencies (interactions) of the features. When a new feature is removed from the part, only the other features dependent upon this feature need be updated. The updating occurs through feature recognition, which is accomplished by searching for closed loops of edges within the solid model.

## 6. SUMMARY AND CONCLUSION

This article has explored the issue of automatic understanding of shapes from CAD databases. This problem is particularly important for applications such as manufacturing and automatic inspection. The gap in information between CAD and CAPP or automated inspection is reflected by the fact that these applications require as input high-level semantic information in the form of features, whereas traditional CAD representations can provide only low-level geometric information such as faces, edges, and vertices. This information mismatch has greatly hindered the progress of manufacturing automation. Three approaches to bridging the gap are interactive feature recognition (human-assisted), feature-based design, and automatic feature recognition. The authors have focused on reviewing, classifying, and discussing in moderate detail each of the automatic feature identification and recognition (AFR) techniques including graph-based methods, logical inference methods, syntactic methods, volumetric methods, and convex-hull techniques.

### 6.1 Information-Processing Methods

Features have been defined by some researchers as collections of faces, and by others as volumes. The information-processing methods for AFR corresponding to these feature definitions can be informally categorized into pattern-matching or volume-extraction techniques. Pattern-matching methods accomplish their task based on the recognition of features as patterns made up of topological entities such as faces, edges, and vertices. The volumetric methods are based on generation and classification of volumetric entities.

The pattern-matching methods can relatively easily and quickly recognize isolated features and simple interacting features. The graph-based techniques, syntactic methods, and logical-inference methods can be classified as different

approaches within this category. These methods, however, exhibit fragile behavior in interpreting more complicated interacting features, because feature interactions may produce unexpected changes in the topological patterns associated with different features. For example, feature interactions may cause the deletion, fragmentation, addition, or merger of topological entities. In addition, feature-recognition techniques based on pattern matching have high computational complexity due to the need to search (potentially exhaustively) huge search spaces. These aspects strongly suggest that pattern matching alone is not appropriate for solving real-world feature-recognition problems. However, with the aid of other techniques such as evidential reasoning, pattern matching can recognize rather complex interacting features. As a result, pattern matching coupled with these techniques seems to offer an attractive approach. An advantage of such combinations is that they can still fully exploit the benefits of the various pattern-matching techniques. The recognized features must still be verified to insure feature validity.

Convex-hull techniques such as ASV and ASVP are not as sensitive to the patterns of topological and geometric entities. However, these methods may not be mature enough to develop complete interpretations of the interacting features from the ASV components. The reason is that in interacting situations, certain common volumes may be shared by more than one feature, but decomposition algorithms normally allocate these common volumes to only one of the interacting features. This problem may be addressed by developing feature-growing techniques, repeatedly selecting primitives and combining them with their adjacent volumes if they satisfy specific matching conditions.

### 6.2 Information Requirements

In terms of input information requirements, B-Reps are currently the most popular geometric representation scheme in mechanisms for automatic recognition of features from design models. B-Rep provides a description of an object in terms of its surface and edge entities. These surface and edge entities encourage the use of pattern-matching techniques for feature recognition. However, these entities are also sensitive to feature interactions since interactions can significantly change the observed entities or their properties. The main advantages of this scheme are that it is unambiguous and unique (ignoring changes in tessellation) and that both the volumetric and pattern-matching techniques can easily use it.

The CSG representation is simple, concise, unambiguous, and a useful paradigm in interactive design. It encourages techniques that are based on the manipulation of volumes. Interactions among features are easier to deduce and resolve at the volumetric level. However, these operations must eventually access the boundaries of the interactions between the volumes, and calculating these boundaries using the CSG representation scheme is expensive. Another major problem with the CSG scheme is its nonuniqueness (the same design may have an unlimited number of CSG representations), which has greatly hindered efforts in feature recognition using this scheme. As a result, CSG has received little attention and, more important, efforts to use CSG for automatic recognition of features invariably begin by converting it into a simpler representation.

Although engineering drawings are still the format by which a majority of designs are represented and/or transferred, in comparison with solid model representation schemes little attention has been paid to these for feature recognition. The major problems encountered in employing engineering drawings for feature recognition or 3D solid model construction include the ambiguity in the representation of engineering components and the difficulty in converting engineering drawings on paper into an

effective computer format. The ambiguity is reflected by the fact that the 2D data may not contain sufficient semantic information to uniquely identify a design object. The assumption that three orthographic views can unambiguously define a unique object is therefore not realistic, and such an assumption significantly limits the application scope of the involved methods. In addition, the conversion of engineering drawings from paper to a computer format such as a symbolic edge-face representation is a difficult problem that involves complex image-processing techniques.

An ideal representation would be a hybrid representation scheme that would inherit the advantages of both B-Rep and CSG. A representation of this nature could enable us to deal with interacting features more easily.

### 6.3 Shortcomings of Existing Systems

Most of the existing approaches to AFR have been limited to a particular application. This is understandable since the definition of features is application-specific. For example, the required feature components might be quite different for, say, process planning and finite element analysis, both of which may differ from the designer's perception of features. Application-independent features may be useful only as an intermediate stage and have to be processed further to suit the application. Furthermore, the features obtained must be validated for the given application. For example, a slot feature is a valid machining feature only if it can be accessed by the cutter tool. However, in the existing literature there has been little focus on the validation of the recognized features. In fact, one may argue the validity is specific to the application and should be left, for example, to the process planner. Such arguments separate the issues of validity of features for application from the geometric interpretation issues.

For design by features we need to decide upon a set of features and base the model on this finite set. However, the "same" feature may have different connotations in the different approaches [Shah 1990]. For example, Figure 7(a), which is defined to be a slot, can be a slot, a pocket, or a profile [Vandenbrande and Requicha 1990, 1993; Requicha 1996]. It is a pocket in the MRSEV library used in Gupta et al. [1994] and is considered as a depression in the approach used in De Floriani [1989]. In the convex-hull techniques [Woo 1982], the model is rendered in terms of alternating sum of volumes rather than features such as slots or pockets. This is a problem, since feature-based modelers need to interface well with existing modelers and application programs [Shah and Mathew 1991]. When a model is transferred from one system to another the features must have the same meaning in both systems. This should also be true when interfacing a feature modeler with an application. Standardization is necessary for this to be possible. Shah and Mathew discuss the conflict between transferability of shape information and versatility versus convenience of any feature data exchange standard.

Existing approaches primarily deal with machining features. Recently, Lentz and Sowerby [1993] developed an algorithm for feature extraction and recognition for a limited domain of sheet metal parts. In addition, information such as tolerances and functional features has not been effectively utilized for feature recognition. A good example of a system taking advantage of such information would be Vandenbrande and Requicha [1990; 1993] and Requicha [1996]. In this approach tolerances and attributes trigger hints on the existence of features.

Furthermore, many of the existing systems deal with an abstraction of real-world components. As a result, these methods, though they provide a framework, cannot be applied directly to an engineering problem. The feature library chosen may have patterns that can never be machined. For example,

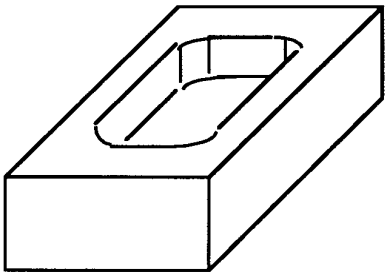

**Figure 23.** Actual machinable pocket.

slots and pockets are simplified into rectangular blocks and their corner radii ignored. A pocket is usually machined by a milling cutter, which has a finite radius. Figure 7(b) shows a pocket that cannot be machined in a milling machine—a milled pocket would realistically look like Figure 23.

## REFERENCES


ALDEFELD, B. 1983. On automatic recognition of 3D structure from 2D representations. *Comput. Aided Des. 15,* 2, 59–64.

BAUMGART, B. G. 1972. Winged edge polyhedron representation. Stanford AI Lab Tech. Memo, AIM-179, STAN-CS-320.

BROWN, P. AND RAY, S. 1987. Research issues on process planning at the National Bureau of Standards. In *Proceedings of the 19th CIRP International Seminar on Manufacturing Systems*.

BURN, J. M. 1991. *B-Rep to CSG Conversion Tool-Oriented Process Planning*. Springer-Verlag.

CHAMBERLAIN, M. A., JONEJA, A., AND CHANG, T.-C. 1993. Protrusion-features handling in design and manufacturing planning. *Comput. Aided Des. 25,* 1, 19–28.

CHANG, T. C., ANDERSON, D. C., AND MITCHELL, O. R. 1988. QTC—an integrated design/manufacturing/inspection system for prismatic parts. In *Proceedings of ASME Conference on Computers in Engineering* (San Francisco, CA), Vol. 1, 417–426.

CHEN, X. AND HOFFMAN, C. M. 1995. On editability of feature-based design. *Comput. Aided Des. 27,* 12, 905–914.

CHIREHDAST, M. AND PAPALAMBROS, P. Y. 1994. Conversion of spatial-enumeration scheme into constructive solid geometry. *Comput. Aided Des. 26,* 4, 302–314.

CHOI, B. K. 1982. CAD/CAM compatible tool-oriented process planning for machining centers. Ph. D. Thesis, Purdue University.

CHOI, B. K., BARASH, M. M., AND ANDERSON, D. C. 1984. Automatic recognition of machined surfaces from 3-D solid model. *Comput. Aided Des. 16,* 2, 81–86.

CHUANG, S. H. AND HENDERSON, M. R. 1990. Three-dimensional shape pattern recognition using vertex classification and vertex-edge graphs. *Comput. Aided Des. 22,* 6, 377–387.

CLARK, D. E. R. AND CORNEY, J. 1994. Identification of general protrusion and depression features. In *Proceedings of ASME Conference on Computers in Engineering*, Vol. 1, 55–65.

CORNEY, J. AND CLARK, D. E. R. 1991. Method for finding holes and pockets that connect multiple faces in 2 1/2D objects. *Comput. Aided Des. 23,* 10, 658–668.

DAVE, P. AND SAKURAI, H. 1995. Maximal volume decomposition and its application to feature recognition. In *Proceedings of ASME Conference on Computers in Engineering*, 553–568.

DE FLORIANI, L. 1989. Feature extraction from boundary models of 3D objects. *IEEE Trans. Pattern Anal. Mach. Intell. 11,* 8, 785–798.

DE MARTINO, T., FALCIDIENO, B., AND GIANNINI, F. 1994. An adaptive feature recognition process for machining contexts. *Adv. Eng. Softw. 20,* 91–105.

DE MARTINO, T., FALCIDIENO, B., GIANNINI, F., HASSINGER, S., AND OVTCHAROVA, J. 1994b. Feature-based modeling by integrating design and recognition approaches. *Comput. Aided Des. 26,* 8, 646–653.

DONG, X. AND WOZNY, M. 1988. FRAFES, a frame-based feature extraction system, In *International Conference on Computer Integrated Manufacturing* (IEEE, Troy, NY), 296–305.

DORI, D. AND TOMBRE, K. 1995. From engineering drawings to 3D CAD models: Are we ready now? *Comput. Aided Des. 27,* 4, 243–254.

DUAN, W., ZHOU, J., AND LAI, K. 1993. FSMT: A feature solid-modeling tool for feature-based design and manufacture. *Comput. Aided Des. 25,* 11, 29–38.

FALCIDIENO, G. AND GIANNINI, F. 1989. Automatic recognition and representation of shape-based features in a geometric modeling system. *Comput. Vision Graph. Image Process. 48,* 10, 93–123.

FENG, C.-X. AND KUSIAK, A. 1995. Constraint-based design of parts. *Comput. Aided Des. 27,* 5, 343–352.

FIELDS, M. C. AND ANDERSON, D. C. 1994. Fast feature extraction for machining applications. *Comput. Aided Des. 26,* 11, 803–813.

FINGER, S. AND SAFIER, S. 1990. Representing and recognizing features in mechanical designs. In *Proceedings of Design Theory & Methodology Conference* (Chicago, IL), 19–25.

FU, K. S. 1982. *Syntactic Pattern Recognition and Applications.* Prentice-Hall, Englewood Cliffs, NJ.

GADH, R. AND PRINZ, F. B. 1992. Recognition of geometric forms using the differential depth filter. *Comput. Aided Des. 24,* 11, 583–599.

GUPTA, S. K. AND NAU, D. S. 1995. Systematic approach to analyzing the manufacturability of machined parts. *Comput. Aided Des. 27,* 5, 323–342.

GUPTA, S. K., KRAMER, T. R., NAU, D. S., REGLI, W. C., AND ZHANG, G. 1994. Building MRSEV models for CAM applications. *Adv. Eng. Softw. 20,* 121–139. Also available as ISR-TR93-84.

HAN, J. H. AND REQUICHA, A. A. G. 1994. Incremental recognition of machining features. In *Proceedings of ASME Conference on Computers in Engineering,* Vol. 1, 143–149.

HENDERSON, M. R. 1984. Extraction of feature information from three dimensional CAD data. Ph. D. Thesis, Purdue University.

HENDERSON, M. R., CHUANG, S. H., AND GAVANKAR, G. P. 1990. Graph-based feature extraction. In *Proceedings of NSF Design and Manufacturing Systems Conference* (Tempe, AZ), 183–189.

HUMMEL, K. E. 1989. Coupling rule-based and object oriented programming for the classification of machined features. In *Proceedings of ASME Conference on Computers in Engineering,* Vol. 1, 409–418.

HWANG, T.-S. AND ULLMAN, D. G. 1994. Recognize features from freehand sketches. In *Proceedings of ASME Conference on Computers in Engineering,* Vol. 1, 67–78.

JAKUBOWSKI, R. 1982. Syntactic characterization of machine-parts shapes. In *Cybern. Syst. Int. J. 13,* 1–24.

JI, Q. AND MAREFAT, M. 1995. Extracting and identifying form features: A Bayesian approach. *Comput. Aided Des. 27,* 6, 435–454.

JOSHI, S. AND CHANG, T. C. 1988. Graph-based heuristic for recognition of machined features from a 3D solid model. *Comput. Aided Des. 20,* 2, 58–66.

JUAN-ARINYO, R. AND SOLE, J. 1995. Constructing face-octrees from voxel-based volume representations. *Comput. Aided Des. 27,* 10, 783–791.

KARINTHI, R. R. AND NAU, D. 1992. An algebraic approach to feature interactions. *IEEE Trans. Pattern Anal. Mach. Intell. 14,* 2, 469–484.

KIM, Y. S. 1990. Convex decomposition and solid geometric modeling. Ph. D. Thesis, Stanford University.

KIM, Y. S. 1992. Recognition of form features using convex decomposition. *Comput. Aided Des. 24,* 9, 461–477.

KIM, Y. S. 1993. Form feature recognition using convex decomposition. In *Proceedings of NSF Design and Manufacturing Systems Conference* (Charlotte, NC).

KIM, Y. S. AND ROE, K. D. 1992. Conversions in form feature recognition using convex decomposition. In *Proceedings of ASME Conference on Computers in Engineering,* Vol. 1, 233–244.

KO, H. AND PARK, M.-W. 1994. Integration methodology for feature-based modeling and recognition. *Adv. Eng. Softw. 20,* 75–89.

KO, H., PARK, M.-W., KANG, H., SOHN, Y., AND KIM, H. S. 1993. Integration methodology for feature-based modeling and recognition. In *Proceedings of ASME Conference on Computers in Engineering,* 23–34.

KUNG, H. 1984. An investigation into development of process plans from solid geometric modeling representation. Ph. D. Thesis, Oklahoma State University.

KYPRIANOU, L. K. 1980. Shape classification in computer-aided design. Ph. D. Thesis, Cambridge University, U.K.

LAKKO, T. AND MANTYLA, M. 1993. Feature modeling by incremental feature recognition. *Comput. Aided Des. 25,* 8, 479–492.

LEE, Y. C. AND FU, K. S. 1987. Machine understanding of CSG: Extraction and unification of manufacturing features. *IEEE Comput. Graph. Appl. 7,* 1, 20–23.

LENTZ, D. H. AND SOWERBY, R. 1993. Feature extraction of concave and convex regions and their intersections. *Comput. Aided Des. 25,* 7, 421–437.

LENTZ, D. H. AND SOWERBY, R. 1994. Hole extraction for sheet metal components. *Comput. Aided Des. 26,* 10, 771–783.

LI, R. K. 1986. A conceptual framework for the interaction of computer-aided design and computer-aided manufacturing. Ph. D. Thesis, Arizona State University.

MAREFAT, M. AND KASHYAP, R. L. 1992. Automatic construction of process plan from solid model representations. *IEEE Trans. Syst. Man Cyber. 22,* 5, 1097–1115.

MAREFAT, M. M. AND KASHYAP, R. L. 1990. Geometric reasoning for recognition of three-dimensional object features. *IEEE Trans. Pattern Anal. Mach. Intell. 12,* 10, 949–965.

MAREFAT, M., FEGHHI, S. J., AND KASHYAP, R. L. 1990. IDP: Automating the CAD/CAM link by reasoning about shape. In *Proceedings of Sixth IEEE Conference on Artificial Intelligence Applications (CAIA)* (Santa Barbara, CA), Vol. 2, 172–177.

MANTYLA, M., NAU, D., AND SHAH, J. 1996. Challenges in feature-based manufacturing research. *Commun. ACM 39,* 2, 77–85.

MARKOWSKY, G. AND WESLEY, M. A. 1986. Generation of solid models from two-dimensional and three-dimensional data. In *Solid Modeling by Computer: From Theory to Application.* Plenum, New York, 23–51.

MEERAN, S. AND PRATT, M. J. 1993. Automated feature recognition from 2D drawings. *Comput. Aided Des. 25,* 1, 7–17.

MENON, S. AND KIM, Y. S. 1994. Handling blending features in form feature recognition using convex decomposition. In *Proceedings of ASME Conference on Computers in Engineering,* Vol. 1, 79–92.

NAU, D. S., GUPTA, S. K., KRAMER, T. R., REGLI, W. C., AND ZHANG, G. 1993. Development of machining alternatives based on MRSEVs. In *Proceedings of ASME Conference on Computers in Engineering,* 47–57.

NII, H. P. 1986. Blackboard systems: The blackboard model of problem solving and the evolution of blackboard architectures, part one. *AI Mag. 7,* 2, 38–53.

OVTCHAROVA, J. AND JASNOCH, U. 1993. Towards a consistency management in a feature-based design. In *Proceedings of ASME Conference on Computers in Engineering,* 129–143.

OVTCHAROVA, J. AND JASNOCH, U. 1995. Feature-based design and consistency management in CAD applications: A unified approach. *Adv. Eng. Softw. 20,* 65–73.

OVTCHAROVA, J., HABINGER, S., VIEIRA, A. S., JASNOCH, U., AND RIX, J. 1994. SINFONIA: An open feature-based design module. In *Proceedings of ASME Conference on Computers in Engineering,* Vol. 1, 29–43.

PARRY-BARWICK, S. AND BOWYER, A. 1995. Multidimensional set-theoretic feature recognition. *Comput. Aided Des. 27,* 10, 731–740.

PEARL, J. 1988. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, Palo Alto, CA.

PERNG, D. B. 1988. Automatic 3D machining feature extraction from 2D CAD data. Ph. D. Thesis, National Chiao Tung University, Taiwan.

PERNG, D. B., CHEN, Z., AND LI, R. K. 1990. Automatic 3D machining feature extraction from 3D CSG solid input. *Comput. Aided Des. 22,* 5, 285–295.

PETERS, T. J. 1993. Mechanical design heuristics to reduce the combinatorial complexity for feature recognition. *Res. Eng. Des. 4,* 4, 195–201.

PHAM, D. T. 1991. *Artificial Intelligence in Design.* Springer-Verlag, New York.

PRABHAKAR, S. AND HENDERSON, M. R. 1992. Automatic form-feature recognition using neural-network-based techniques on boundary representations of solid models. *Comput. Aided Des. 24,* 7, 381–393.

REGLI, W. C., GUPTA, S. K., AND NAU, D. S. 1994. Feature recognition for manufacturing analysis. In *Proceedings of ASME Conference on Computers in Engineering,* Vol. 1, 93–104.

REQUICHA, A. A. G. 1980. Representation for rigid solids: Theory, methods, and systems. *Comput. Surv. 12,* 4, 105–131.

REQUICHA, A. A. G. 1996. Geometric reasoning for intelligent manufacturing. *Commun. ACM 39,* 2, 71–76.

REQUICHA, A. A. G. AND VOELCKER, H. B. 1985. Boolean operations in solid modeling: Boundary evaluation and merging algorithms. *Proc. IEEE 73,* 1, 30–44.

ROSEN, D. W. 1993. Feature-based design: 4 hypotheses for future CAD systems. In *Proceedings of ASME Conference on Computers in Engineering,* 119–128.

ROSSIGNAC, J. R. AND O'CONNOR, M. 1990. Selective geometric complex: A dimension-independent model for pointsets with internal structures and incomplete boundaries. In *Geometric Modeling for Product Engineering,* M. Wozny, J. U. Turner, and K. Preiss, eds., North-Holland.

SAKURAI, H. 1994. Decomposing a delta volume into maximal convex volumes and sequencing them for machining. In *Proceedings of ASME Conference on Computers in Engineering,* Vol. 1, 135–142.

SAKURAI, H. 1995. Volume decomposition and feature recognition: Part 1—polyhedral objects. *Comput. Aided Des. 27,* 11, 833–843.

SAKURAI, H. AND CHIN, C.-W. 1993. Defining and recognizing cavity and protrusion by volumes. In *Proceedings of ASME Conference on Computers in Engineering,* 59–65.

SAKURAI, H. AND GOSSARD, D. C. 1990. Recognizing shape features in solid models. *IEEE Comput. Graph. Appl. 10,* 5, 22–32.

SEDGEWICK, R. 1984. *Algorithms.* Addison-Wesley, Reading, MA.

SHAFER, G. 1976. *A Mathematical Theory of Evidence.* Princeton University Press, Princeton, NJ.

SHAH, J. J. 1991. Assessment of features. *Comput. Aided Des. 23,* 5, 331–343.

SHAH, J. J. AND MANTYLA, M. 1995. *Parametric and Feature-Based CAD/CAM: Concepts, Techniques, and Applications,* John Wiley, New York.

SHAH, J. J., NAU, D. S., AND MANTYLA, M., eds. 1994. *Advances in Feature Based Manufacturing,* Elsevier, North Holland.

SHAPIRO, V. AND VOSSLER, D. L. 1993. Separation for boundary to CSG conversion. *ACM Trans. Graph. 12,* 1, 35–55.

SHPITALNI, M. AND FISCHER, A. 1994. Separation of disconnected machining regions on the basis of a CSG model. *Comput. Aided Des. 26,* 1, 46–58.

STALEY, S., HENDERSON, M. R., AND ANDERSON, D. C. 1983. Using syntactic pattern recognition to extract feature information from a solid geometric data. *Comput. Mech. Eng. 2,* 2, 61–66.

TANG, K. AND WOO, T. 1991a. Algorithmic aspects of alternating sum of volumes part 1: Data structure and difference operation. *Comput. Aided Des. 23,* 5, 357–365.

TANG, K. AND WOO, T. 1991b. Algorithmic aspects of alternating sum of volumes part 2: Nonconvergence and its remedy. *Comput. Aided Des. 23,* 6, 435–422.

TRIKA, S. N. AND KASHYAP, R. L. 1994. Geometric reasoning for extraction of manufacturing features in iso-oriented polyhedrons. *16,* 11, 1087–1100.

TSENG, Y.-J. AND JOSHI, S. B. 1994. Recognizing multiple interpretations of interacting machining features. *Comput. Aided Des. 26,* 9, 667–688.

VAGHUL, M. AND DIXON, J. R. 1985. Expert systems in a CAD environment: Injection molding part design as an example. In *Proceedings of ASME Conference on Computers in Engineering,* Vol. 2, 77–82.

VAN HOUTEN, F. J. A. M., VAN'T ERVE, A. H., AND JONKERS, F. J. C. M. 1989. PART, a CAPP system with a flexible architecture. In *Second CIRP Workshop on CAPP* (Hannover).

VANDENBRANDE, J. 1990. Automatic recognition of machinable features in solid models. Ph. D. Thesis, University of Rochester.

VANDENBRANDE, J. AND REQUICHA, A. A. 1990. Spatial reasoning for automatic recognition of interacting features. In *Proceedings of ASME Computers in Engineering Conference,* Vol. 1, 251–256.

VANDENBRANDE, J. H. AND REQUICHA, A. A. G. 1993. Spatial reasoning for the automatic recognition of machinable features in solid models. *IEEE Trans. Pattern Anal. Mach. Intell. 15,* 12, 1269–1285.

VAXIVIERE, P. 1992. Celesstin: CAD conversion of mechanical drawings. *IEEE Comput. 25,* 7, 46–54.

WACO, D. L. AND KIM, Y. S. 1993. Considerations in positive to negative conversion for machining features using convex decomposition. In *Proceedings of ASME Conference on Computers in Engineering,* 35–46.

WACO, D. L. AND KIM, Y. S. 1994a. Geometric reasoning for machining features using convex decomposition. *Comput. Aided Des. 26,* 6, 477–489.

WACO, D. L. AND KIM, Y. S. 1994b. Handling interacting positive components in machining feature reasoning using convex decomposition. *Adv. Eng. Softw. 20,* 107–119.

WANG, E. 1992. Using automatic feature recognition to interface CAD to CAPP. In *Proceedings of ASME Conference on Computers in Engineering,* Vol. 1, 215–231.

WOO, T. C. 1982. Feature extraction by volume decomposition. In *Proceedings of Conference on CAD/CAM Technology in Mechanical Engineering* (MIT, Cambridge, MA), 76–94.

WOODWARK, J. R. 1988. Some speculations on feature recognition. *Comput. Aided Des. 20,* 4, 189–196.

YU, T.-T., CHEN, Y.-M., AND MILLER, R. A. 1992. A framework for sheet metal part design and manufacturing assessment. In *Proceedings of ASME Conference on Computers in Engineering,* Vol. 1, 53–60.

Received October 1993; revised February 1997; accepted May 1997