Extracting Alternative Machining Features: Al Algorithmic Approach

by W.C. Regli, S.K. Gupta, D.S. Nau

T.R. 94-55

*Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry*

Extracting Alternative Machining Features: An Algorithmic Approach

William C. Regli
Department of Computer Science and
Institute for Systems Research
University of Maryland
College Park, MD 20742

Satyandra K. Gupta
Mechanical Engineering Department and
Institute for Systems Research
University of Maryland
College Park, MD 20742

Dana S. Nau
Department of Computer Science,
Institute for Advanced Computer Studies and
Institute for Systems Research
University of Maryland
College Park, MD 20742

Corresponding Author: William C. Regli
Department of Computer Science and
Institute for Systems Research
University of Maryland
College Park, MD 20742

Extracting Alternative Machining Features: An Algorithmic Approach

William C. Regli*
Department of Computer Science and
Institute for Systems Research
University of Maryland

Satyandra K. Gupta
Mechanical Engineering Department and
Institute for Systems Research
University of Maryland

Dana S. Nau
Department of Computer Science,
Institute for Advanced Computer Studies and
Institute for Systems Research
University of Maryland

Abstract

Automated recognition of features from CAD models has been attempted for a wide range of application domains. In this paper we address the problem of representing and recognizing the complete class of features in alternative interpretations for a given design.

We present a methodology for recognizing a class of machinable features and addressing the computational problems posed by the existence of feature-based alternatives. Our approach addresses a class of volumetric features that describe material removal volumes made by operations on 3-axis vertical machining centers including: drilling, pocket-milling, slot-milling, face-milling, chamfering, filleting, and blended surfaces.

This approach recognizes intersecting features and is complete over all features in our class; i.e., for any given part, the algorithm produces a set containing all features in our class that correspond to possible operations for machining that part. This property is of particular significance in applications where consideration of different manufacturing alternatives is crucial.

This approach employs a class of machinable features expressible as MRSEVs (a STEP-based library of machining features). An instance of this methodology has been implemented using the ACIS solid modeler and the National Institutes of Health C++ class library (NIHCL).

1 Introduction

In general, there may be many different ways to manufacture a given design. It is becoming increasingly evident that consideration of these manufacturing alternatives is crucial for tasks such as manufacturability analysis and process planning. How easy it is to manufacture a design, or whether it will be possible to meet the design specifications at all, may depend on which manufacturing alternative is chosen.

In the specific case of machining, different ways to manufacture a design for a machinable part correspond to different interpretations of the design as sets of machining features. Ideally, we would like a feature recognition system to find the machining features corresponding to all of the different ways in which the design could be machined. However, for complex parts, it usually is not feasible simply to enumerate *all* of the feature instances, because the number of them can be very large, or even infinite—and in most cases, very few of the potential feature instances for a part will make practical manufacturing sense.

To address this need, this paper presents a methodology for recognizing a class of "well-behaved" machinable features that are useful for manufacturing. These features could then be used as input to a variety of possible applications, such as manufacturability analysis, automated redesign, process planning, or part-code generation for group technology. Our approach has the following characteristics:

1. It is complete over all features in our class; i.e., for any given part, the set of features produced by the algorithm contains all features in our class that correspond to machining operations on that part, regardless of how complicated the intersections are among those features.

2. It is capable of identifying and eliminating some machining features as inaccessible. The features eliminated are guaranteed to be inaccessible in all potential operation plans for the given design.

3. Applications such as manufacturability analysis and automated redesign require features that correspond directly to manufacturing operations. The class of machinable features we employ are expressible as MRSEVs (a PDES/STEP-based library of machining features) [23]. MRSEVs are definable using EXPRESS (the PDES information modeling language) and as PDES form features. By employing a set of features based on a standard interchange format such as STEP, we are attempting to use an independently defined feature class and address a domain of machinable parts of interest to a large community.

4. The algorithms have been implemented and are being incorporated as part of the *IMACS*[1] design critiquing system under development at the University of Maryland's Institute for Systems Research.

Section 2 presents a survey of related work in the areas of feature recognition and handling alternative interpretations. Section 3 defines the class of machining features that we will consider in this paper and introduces the essential terminology for describing the recognition problem and alternative feature interpretations. Section 4 describes a methodology for recognizing instances from our feature class from a solid model of a part, provides an analysis of the completeness and complexity of the approach, and gives a brief overview of how the output of the feature recognition system can be employed to generate and evaluate alternative feature interpretations of the part. Section 5 discusses our implementation of this system and illustrates three example parts. Section 6 outlines how this methodology is used to generate and evaluate alternative feature-based models for the part. Lastly, Section 7 gives conclusions and future directions for work in this area.

2 Related Work

Feature-based CAD/CAM techniques have been an important research area over the past decade. Feature recognition has been successfully employed for a variety of applications including process planning, design analysis, and part code generation for group technology. Significant effort has been directed towards defining sets of form features to fit the requirements for individual applications and exploit the strengths of the pattern searching or knowledge-based techniques used to recognize

[1]Interactive Manufacturability Analysis and Critiquing System.

them. For a more comprehensive overview of feature-based manufacturing techniques, the reader is referred to [41].

2.1 Feature Recognition

In one of the early efforts on feature extraction, Woo [44] proposed a method for finding general depression and protrusion features on a part by decomposing the convex hull of the solid model. The approach had several problems, including the existence of pathological cases in which the procedure would not converge. The non-convergence of Woo's approach has been solved in recent work by Kim [20, 21, 43], whose system produces a decomposition of the convex hull of a part as general form features. Extension of this method from polyhedra to the more general surfaces required for realistic parts is currently under investigation.

The seminal work of Henderson [16] applied rule-based systems on the feature recognition problem and has served as a foundation for more recent AI-based approaches. Henderson has also made extensive use of graph-based methodologies: in [10] making use of graph-based algorithms to find protrusion and depression features. In Chuang and Henderson [2], graph-based pattern matching is used to find feature patterns from part geometry and topology. Chuang and Henderson [3] were the first to note the need to explicitly address both computational complexity and decidability when defining the feature recognition problem. Their paper formalized the problem of recognition of features (including compound features) by parsing a graph-based representation of a part using a web grammar. More recently, Gavankar and Henderson [33] adapted neural networks to recognize features from polyhedral objects. Peters [30] describes techniques for training neural networks to recognize feature classes that can be customized by the end user.

De Floriani [7] employed graph-based algorithms for finding bi-connected and tri-connected components to partition a polyhedral part into several varieties of protrusion and depression features. Joshi's [18] approach used subgraph isomorphism algorithms to match feature patterns to patterns in the topology of polyhedral parts. Sakurai [40] developed a graph-based system capable of handling limited types of user-defined features, providing for a degree of application-specific customizability. In many of these approaches, the graph-based representation schemes have proven difficult to extend to the more complex surfaces and features found in realistic manufacturing problems. Corney and Clark [4, 5] have had success extending the capabilities of graph-based algorithms to more general $2\frac{1}{2}$-dimensional parts.

Kyprianou [24] presented the first effort to use a grammatical approach to parse solid models of parts for group coding. Methods based on graph-grammars have been used both to recognize features [32, 38] and to translate between differing feature representations [37]. Peters [31] analyzes the combinatorial complexity of graph and grammatical approaches to feature recognition and presents heuristics to reduce it. To address such combinatorial problems, recent work by Gadh and Prinz [9] describes techniques by abstracting an approximation of the geometric and topological information in a solid model and finding features in the approximation.

The work of Dong [8] included formalization of a feature description language and was the first to employ a frame-based reasoning system to extract machining features for computer-aided process planning. Dong's approach included the ability to construct volumetric features from surface features and perform an analysis of tool accessibility.

The ability to recognize interacting features has been a goal of a number of numerous research efforts, among them [8, 9, 18]. The approach of Marefat [27] built on the representation scheme of Joshi [18] and used a combination of expert system and hypothesis testing techniques to extract surface features from polyhedral objects and handle a variety of their geometric interactions. Marefat argues that his approach is complete over a class of polyhedral features, i.e., that it generates all features in his class that can be found from the geometry of a part.

The most comprehensive approach to date for recognizing features and handling their interactions has been the OOFF system (Object-Oriented Feature Finder) of Vandenbrande [42]. Vandenbrande's work, using a knowledge-based approach like Dong's, provides a framework for recognizing a significant class of realistic machining features of interest for process planning via artificial intelligence techniques in combination with queries to a solid modeler. He formalized a class of machining features and presented recognition "hints" for each class. The hints are extracted from the solid model and classified as to their potential for building a feature instance; unpromising hints are discarded. A frame-based reasoning system then acts on the hints and attempts to complete a feature frame with information needed to construct geometrically maximal feature instances. One of the fundamental contributions of Vandenbrande's work was a formal method for representing interactions among the features by calculating the "required" and "optional" volumes for each promising feature instance.

The recent work Laakko and Mäntylä [25] couples feature-based design and feature recognition to provide for incremental feature recognition. This type of approach identifies changes in the geometric model as new or modified features while preserving the existing feature information. They also provide for some form of customizability with use of a feature-definition language to add new features into the system.

2.2 Finding Alternative Feature Interpretations

The AMPS process planning system [1] uses heuristics for feature refinement to combine a set of features into a more complex feature, or split a feature into two or more features.

Karinthi [19] completed the first systematic work on the generation of alternative interpretations of the same object as different collections of volumetric features. They present an algebra for computing alternate interpretations of parts resulting from algebraic operations on the features.

The OOFF system of Vandenbrande [42] produces some alternative feature interpretations. However, the generation of alternatives is not well controlled nor is the class of alternatives produced by OOFF specified.

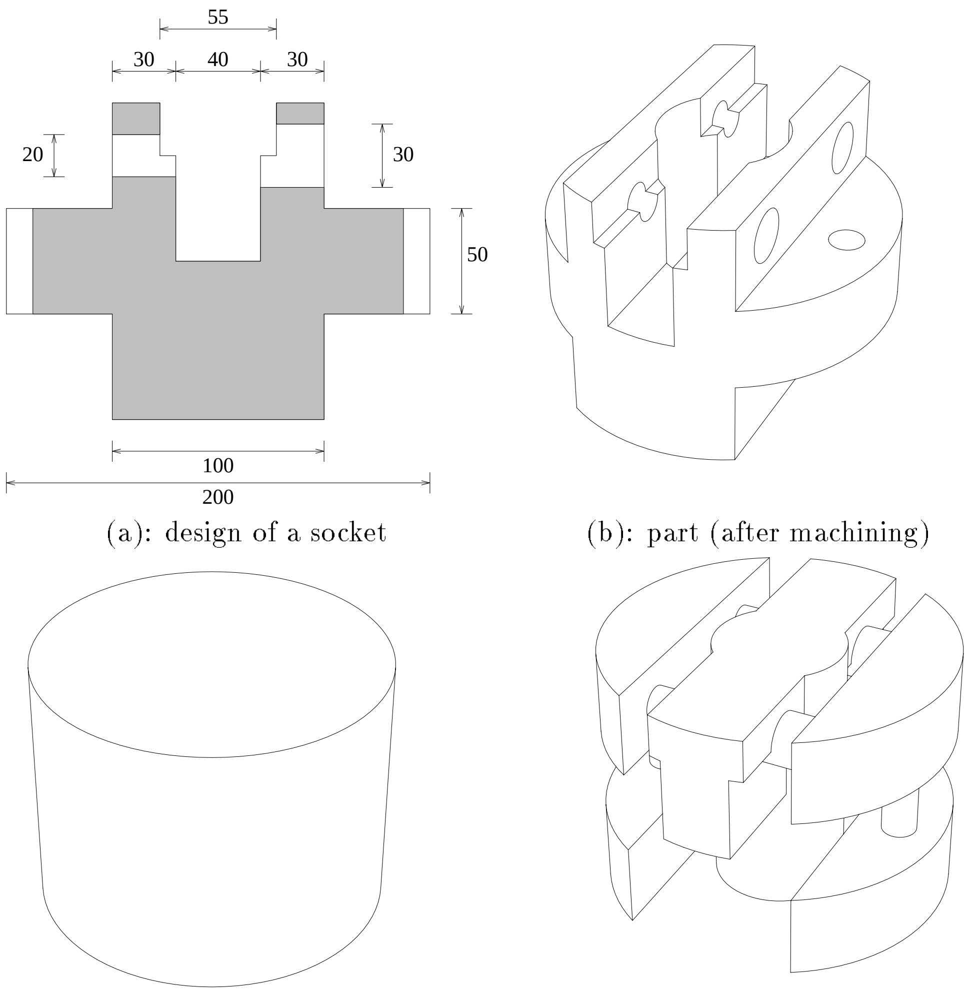
In recent work by Sakurai [39], the volume to be machined is decomposed into cells. Exhaustively, each combination of cells is then matched against user-defined feature templates. While the method is capable of generating all alternative feature interpretations composed of the primitive cells, it does so at a large combinatorial cost.

Waco and Kim [43] have extended convex decomposition techniques to produce alternative decompositions of the removal volume through aggregating and growing form feature primitives.

3 Preliminaries

A common class of solids are those described by r-sets with manifold boundaries [26, 36]. In the context of this paper, a *solid* is an r-set whose boundary is a manifold consisting of planar, elliptical, toroidal, conical and spherical surfaces. A design is specified in terms of a solid model and associated design attributes (such as tolerances, design features, and geometric and topological specifications) that are to be realized through machining operations. In this paper, we will focus on the geometric and topological design attributes.

A machined part, P, is a solid object represented by a CAD model of the part design to be produced by a finite set of machining operations. For example, Figure 1(a) shows a design for



(a): design of a socket (b): part (after machining)

(c): stock (before machining) (d): the delta volume

Figure 1: An example part and stock.

a socket and Figure 1(a) as CAD model of the design. The initial workpiece, S, is the solid object of raw stock material to be acted upon by a set of machining operations generating features (Figure 1(c)). Machining operations remove material from the initial workpiece in order to create the design attributes of the part (i.e., $P \subseteq S$). As illustrated in Figure 1(d), the total removal volume is referred to as the *delta volume* (Δ), and it is the regularized difference [17] of the initial workpiece and the design: $\Delta = S -^* P$.

3.1 Feature Class

In a machining operation, a cutting tool is swept along a trajectory, and material is removed by the motion of the tool relative to the current workpiece. The volume resulting from a machining operation is called a machining feature. A machining feature corresponds to a single machining operation made on one machine setup. Each machining feature has a single approach direction (or orientation) for the tool—this is represented by a unit vector, $\vec{v}$.

In this paper, we will consider features to be instances of *feature types*, parameterized solids

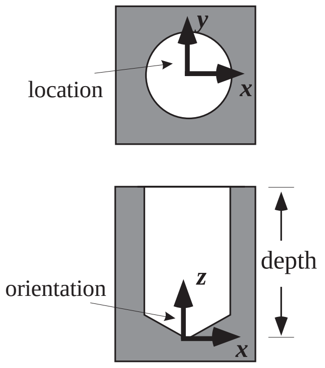
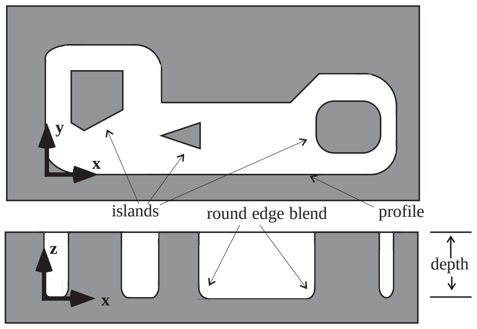
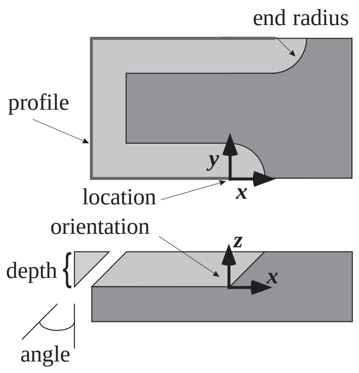
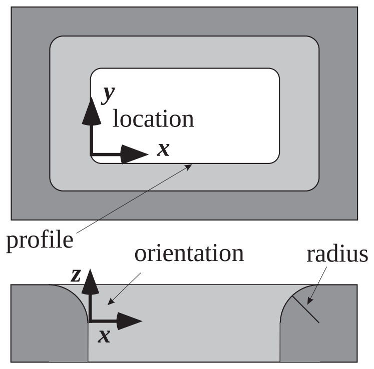
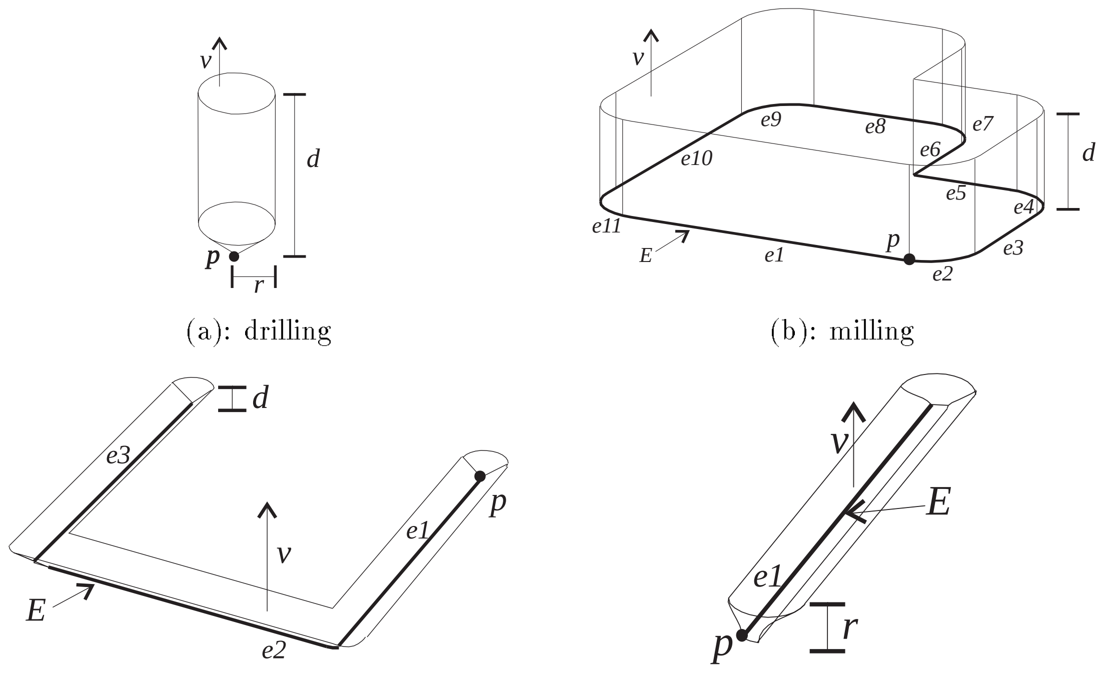
Attributes	Description
Drilling Feature **location** p **radius** r **orientation** $\vec{v}$ **depth** d	
Milling Feature **location** p **orientation** $\vec{v}$ **depth** d **edge profile** E **bottom blend** b_{type}, b_d **island edge profiles** I	 y x islands round edge blend profile z x depth
Chamfering Feature **location** p **orientation** $\vec{v}$ **edge profile** E **depth** d **angle** a **end radius** r_e	 end radius profile y location x orientation z depth x angle
Filleting Feature **location** p **orientation** $\vec{v}$ **radius** r **edge profile** E **end radius** r_e	 y location x profile orientation radius z x

Figure 2: Classes of machining features.



(a): drilling (b): milling

(c): chamfering (d): filleting

Figure 3: Parameterized feature instances.

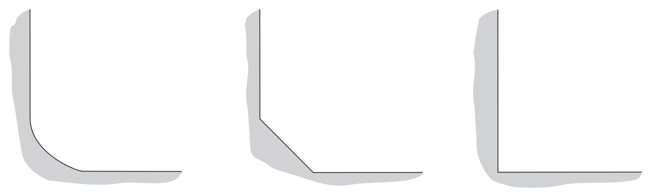

(a): round blend (b): flat blend (c): no blend

Figure 4: Bottom blend types.

that correspond to various types of machining operations on a 3-axis machining center. We will use the following feature types, which are adapted from Kramer's library of Material Removal Shape Element Volumes (MRSEVs) [23]:[2]

1. In order to create a *drilling feature*, we will sweep a drilling tool of radius r for a distance d along the trajectory represented by its *orientation vector* $\vec{v}$, ending at a *datum point* p, as shown in Figure 2. Thus, the volume describing the drilling feature can be modeled as a parameterized volume, as shown in Figure 3(a), consisting of a cylinder of radius r boolean unioned with a cone that represents the conical tip of the drilling tool. The conical tip has a tip angle that describes the shape of the surface of the drilling tool end. We assume there exist a finite number of possible tip angles based on the drilling tools that are available; for simplicity all of the examples of drilling features in this paper have tip angles of 120 degrees.

[2]Kramer's MRSEVs are volumetric features that categorize the shapes of volumes that can be removed by operations on a 3-axis machining center. They can be defined using the EXPRESS modeling language and as STEP form features. The features described in this paper correspond to MRSEV hole, pocket, and edge-cut feature types.

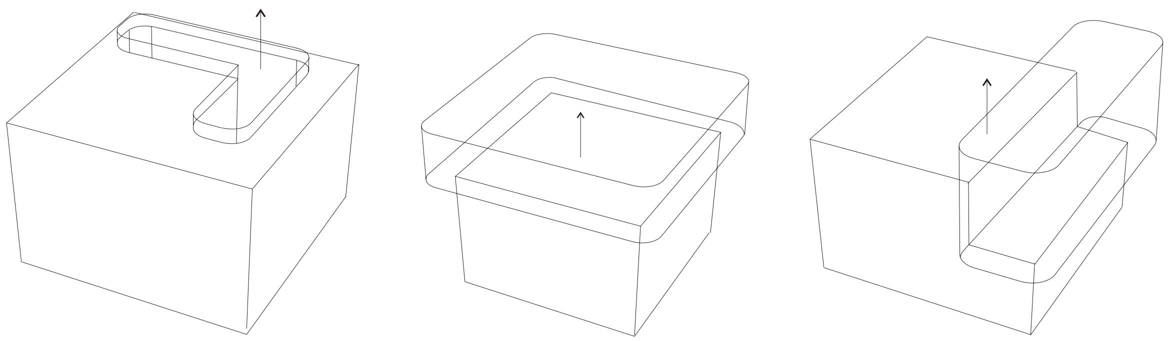

(a):pocket-milling feature (b):face-milling feature (c):step-milling feature

Figure 5: Subclasses of milling features. Arrows denote feature orientation.

2. To create a *milling feature*, we will sweep a milling tool whose orientation vector is $\vec{v}$, starting at a datum point p, and moving through a swept volume of depth d whose cross-sectional area is bounded by an *edge profile* [23] $E = \{e_1, e_2, \ldots e_n\}$, as shown in Figure2. Thus, the volume describing the milling feature can be modeled as a parameterized volume, as shown in Figure3(b).[3] The parameters for this volume also include the following:

   - Within the area bounded by the edge profile E there may be a finite set I of zero or more *islands*. Each island i in I is bounded by its own edge profile E_i.
   - Depending on the shape of the milling tool, there may be a transition surface between the bottom face of the milling feature and its side and island faces. There are three possibilities for the shape of this transition surface, as shown in Figure 4. To represent which of these shapes a milling feature has, we will associate an optional bottom blend type b_{type} and dimension b_d with the feature.

   Suppose f is a milling feature with an edge profile E. Then each edge e of E bounds some side face s of f. If it is possible to sweep s for some nonzero distance away from f without intersecting the part, then we will say that e is *open*; otherwise e is *closed*. Depending on which edges of E are open and which edges of E are closed, milling features can be partitioned into the following three subclasses:

   (a) Figure 5(a) shows an example of a *pocket-milling feature*. For this subclass of milling feature, each edge in the edge profile E is closed.

   (b) Figure 5(b) shows an example of a *face-milling feature*. For this subclass of milling feature, there are no islands or bottom blends and all of the edges in the edge profile E are open.

   (c) Figure 5(c) shows an example of a *step-milling feature*. For this subclass of milling feature, at least one, but not all, of the edges in E are open.

[3]The parameters of the volume describing milling feature do not require that all of the corners be round. Thus, the volume they define will not always correspond *directly* to a milling operation; for example, all three of the features in Figure5(c) qualify, geometrically, as milling features according to our definition. Profile offsetting (described in Section3.4) is performed in a later step to modify the volumes so they do correspond to milling operations. Once altered, the volumes initially identified without corner radii produce alternative features useful during the redesign process, where one might need to minimize tool changes or machine setups.

3. To create a *chamfering feature*, we will sweep a chamfering tool whose orientation vector is $\vec{v}$, starting at a datum point p and moving at a depth d along a trajectory described by an edge profile $E = \{e_1, e_2, \ldots e_n\}$, as shown in Figure 2. Thus, the volume describing the chamfering feature can be modeled as a parameterized volume, as shown in Figure 3(c). The parameters for this volume also include the cutting tool's tip angle a (we assume that there is tooling available with a 45 degree tip angle); and an optional end radius, r_e, to denote the conical surface that might be left at the start and end of a chamfer.

4. To create a *filleting feature*, we will sweep a filleting tool with orientation vector $\vec{v}$ and radius r, starting at a datum point p, and moving at a depth d along a trajectory described by an edge profile $E = \{e_1, e_2, \ldots e_n\}$, as shown in Figure 2. Thus, the volume describing the filleting feature can be modeled as a parameterized volume as shown in Figure 3(d). Note that each edge e in the edge profile E bounds a curved surface of radius r that is tangent to the orientation vector $\vec{v}$ at the edge e.

   The parameters for this volume also include an optional end radius, r_e, to denote the toroidal surface that might be left at the start and end of a fillet. We will assume that there is a finite set of available tools, each with fixed radii.

$\mathcal{M}$ is the set of all instances of the above feature types. For each feature f in $\mathcal{M}$, $type(f)$ is f's feature type.

3.2 Primary Features

Removal volumes for these features are bounded by different types of surfaces, each of which (planar, conical, etc.) may be considered the subset of the boundary of one or more instances of machining features. For realistic machined artifacts only a few of these features yield reasonable machining operations.

We will be interested in features that correspond to the maximal realistic machinable volume made by a single machining operation in a single machining setup. Such features can be easily truncated to produce the machining volumes used in actual operation plans [15, 13].

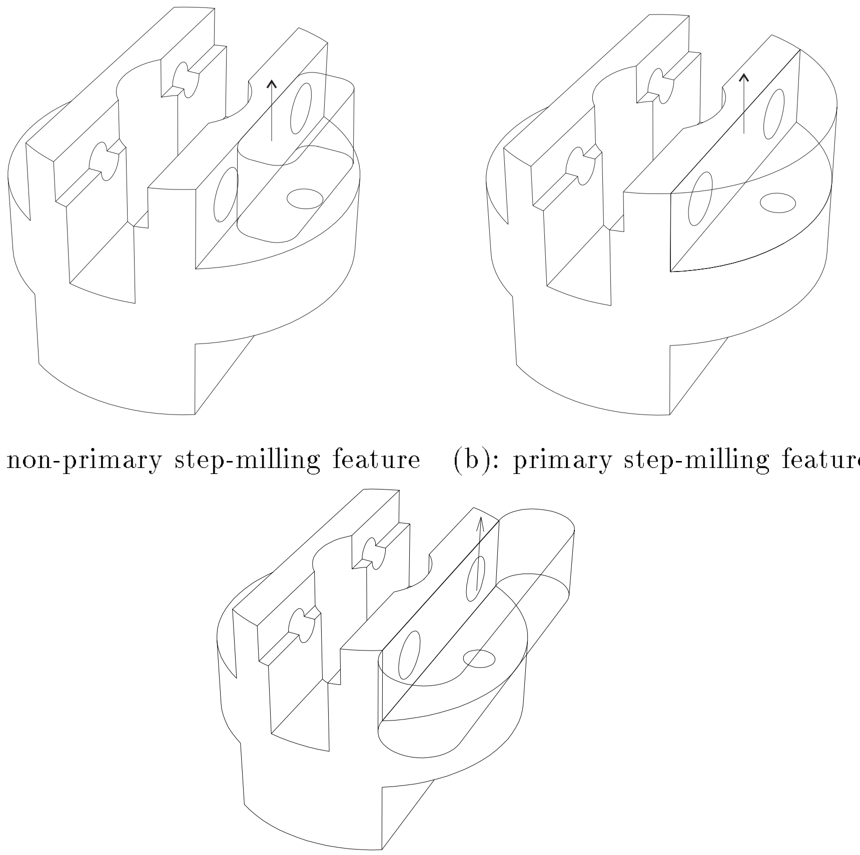
Let M be a set of features, and f be any feature in M. Then f is *M-primary* if f satisfies the following properties:

- f removes as much stock material as possible without intersecting P. In other words, f does not intersect P, and there is no feature instance g in M that has the same machining operation and orientation as f, does not intersect P, and removes more material than f. More formally, there is no g in M with the same orientation vector as f such that $f \cap^* P = \emptyset$, $type(f) = type(g)$, $g \cap^* P = \emptyset$, and $(f \cap^* S) \subset (g \cap^* S)$.

- f is the smallest feature in M that satisfies the above property. In other words, for every feature instance f' in M that satisfies the above property, $f \subseteq f'$.

A feature is *primary* if it is $\mathcal{M}$-primary.

As an example, the feature shown in Figure 6(a) is not primary because it describes only a portion of the maximum possible removal volume, and the feature shown in Figure 6(b) is primary because it describes the maximum possible removal volume.


(a): non-primary step-milling feature (b): primary step-milling feature

(c): offset primary step-milling feature

Figure 6: Examples of non-primary, primary, and an offset primary feature instances.

3.3 Well-Behaved Features

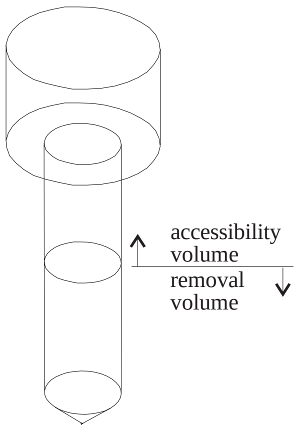
Given solids representing the part P and the stock S as input, ideally, one would like an algorithm that returns the set of all features that can be used to generate an operation plan for producing P from S. This is an unrealistic expectation, because for even simple P and S there may be, theoretically speaking, infinitely many possible feature instances, even when restricted to primary features [14]. This raises the following question: of potentially infinite many features, which should be recognized?

To eliminate many unrealistic feature possibilities, we define a feature f to be *well-behaved* if it satisfies any of the following properties:

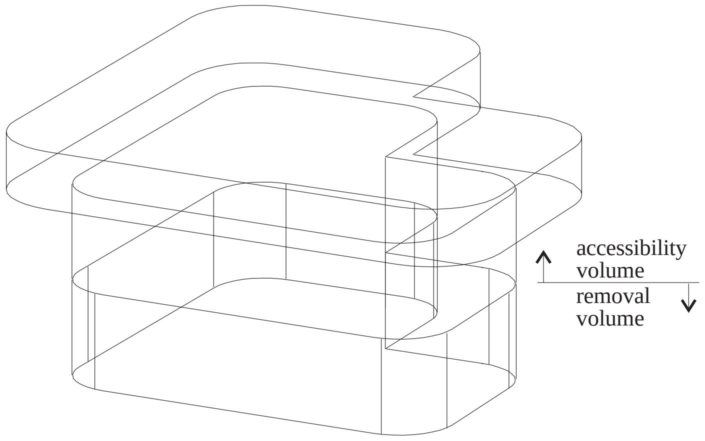
1. f is an M-primary drilling feature, where M is the set of all drilling features f in $\mathcal{M}$ such that a subface of one of f's side or ending surfaces is part of $b(\Delta)$, the boundary of Δ.

2. f is a primary milling feature, and f subsumes an M-primary milling feature, where M is the set of all milling features f in $\mathcal{M}$ such that f's bottom is coplanar with two or more non-collinear edges of Δ. For an example, see Figure 9(c).

3. f is an M-primary filleting or chamfering feature, where M is the set of all filleting and



(a): drilling feature from Figure 3(a).



(b): milling feature from Figure 3(b).

Figure 7: Accessibility volumes associated with a drilling and a milling feature.

chamfering features f in $\mathcal{M}$ such that every edge in f's edge profile is an edge of Δ and $b(f) \cap^* b(\Delta) \neq \emptyset$.

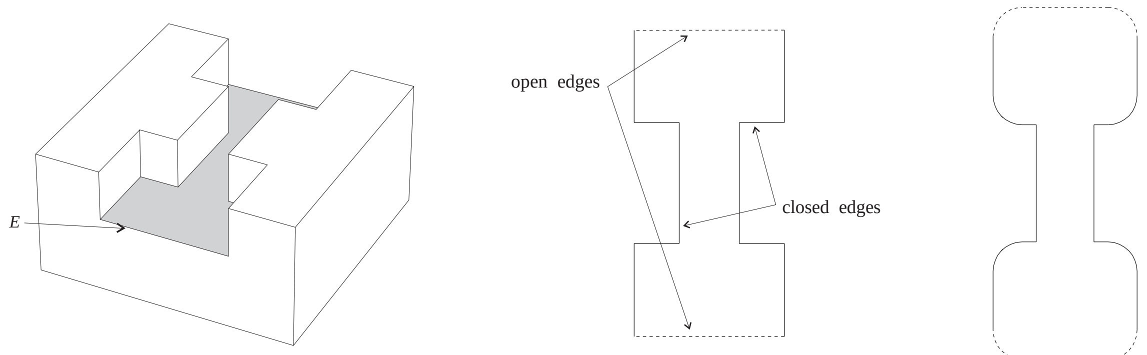
3.4 Offsetting and Accessibility

Although the feature types defined earlier are intended to correspond to machining operations, specific feature instances may sometimes present unrealistic machining requirements. Our approach addresses the machining issues described below.

Tooling Constraints. Some features may violate constraints on the physical dimensions of tooling (e.g., limits on the maximum radius for drilling features and limits on dimension for surfaces identified as blends, fillets, or chamfers). We will want to disregard such features.

Accessibility. Some feature instances might not be machinable due to a variety of machining considerations, and we want to avoid generating these features. In general, accessibility is a very complicated property to verify. It depends on the shape and dimensions of the machine tool and cutting tool, and the order in which the features are machined—all of which are decided when an operation plan is generated. Development of a general methodology for determining whether a feature f is accessible would require generating all of the alternative operation sequences employing the feature f, to see if f is accessible in any one of them. Such algorithms have been developed in the context of generating and evaluating alternative operation plans [11, 15, 12], but are beyond the scope of feature recognition *per se*.

For the purposes of this paper, we use the following criteria to eliminate obviously inaccessible features. For each feature f we calculate an accessibility volume, $acc(f)$, that corresponds to the volume swept out by the non-cutting portion of the tool that machines f. If $acc(f)$ has a non-empty intersection with the part P, then feature f is inaccessible in every operation plan for P, and thus can be discarded. In Figures 7(a) and (b), the accessibility and removal portions of feature volumes are illustrated.



(a): profile E (b): before offsetting (c): after offsetting

Figure 8: An example of edge profile offsetting.

Offsetting. A feature may contain sharp corners that cannot be machined by a milling tool, or the most cost-effective way to mill a volume may be to perform the machining operation using the largest possible tool. Such situations may require that the tool move outside the boundary of the stock material.

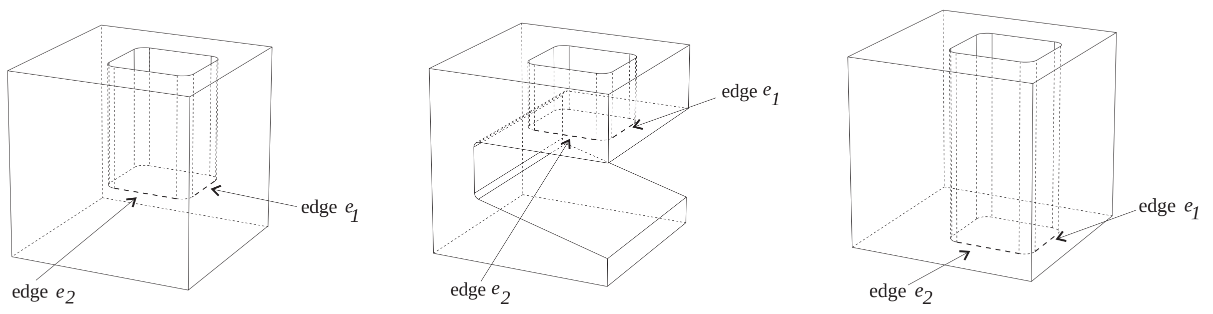
After a possible milling profile has been identified, we will want to adjust its profile to provide an *offset feature* such as shown in Figure 8, that takes these machinability considerations into account. In the figure, the edges of profile E have been offset to take into account the radius of a cutting tool. An example of an offset step-milling feature is given in Figure 6(c). Our method for feature offsetting is summarized in Section 4.6.

4 Recognizing Features

The feature recognition problem can be defined as follows: given a part P and a piece of stock S, return the set of feature instances $\mathcal{F}$ found from P and S. Our algorithm for this problem has two components. The first builds the set of well-behaved feature instances, as outlined below:

```
BUILD_WELL-BEHAVED_FEATURES($P, S$)
INPUT: solid models of a part $P$ and stock $S$
OUTPUT: the set of well-behaved feature instances, $\mathcal{F}$.

  1. Initialize $\mathcal{F} = \emptyset$.
  2. For all geometric attributes (such as edges, faces, and vertices) $g$ of $P$ and $S -^* P$ do
      (a) For each feature type $t$ in $\mathcal{M}$
            i. Construct all well-behaved feature instances of type $t$ capable of creating $g$, as
               described in Sections 4.1 through 4.3.
           ii. Add these feature instances to $\mathcal{F}$
  3. Return($\mathcal{F}$)
```



(a): bottomed milling feature (b): bottomless milling feature (c): through milling feature

Figure 9: Cases of possible milling features.

A second procedure adjusts the set of well-behaved features, $\mathcal{F}$, discarding inaccessible ones and, where possible, offsetting edge profiles:

```
BUILD_FEATURES($P, S$)
INPUT: solid models of a part $P$ and stock $S$
OUTPUT: a set of feature instances, $\mathcal{F}$.

  1. $\mathcal{F}$ = BUILD_WELL-BEHAVED_FEATURES($P, S$)
  2. For each feature $f$ in $\mathcal{F}$ do
      (a) Perform an inaccessibility check of $f$
      (b) If $f$ is inaccessible, remove it from $\mathcal{F}$
  3. For each milling feature $f$ in $\mathcal{F}$ do
      (a) Offset the edge profile of $f$
      (b) Perform an inaccessibility check of $f$
      (c) If $f$ is inaccessible, remove it from $\mathcal{F}$
      (d) Examine the edge profile of $f$ to identify subclassification for milling feature (i.e., pocket-
          milling, face-milling, or step-milling).
  4. Return($\mathcal{F}$)
```

As space limitations preclude giving an implementation-level description, the remainder of this section will outline the geometric procedures for constructing individual well-behaved primary feature instances.

4.1 Constructing Drilling Features.

Drilling features are perhaps the most straightforward to recognize. An instance of a drilling feature can be found from any subface of a conical end surface or the cylindrical side face. From a portion of the conical ending surface, one can determine both the location and the orientation of the drilling feature. The radius r for the primary feature is calculated based on two observations: first, it must be less than or equal to the maximum radius in the available set of drilling tools. Second, it is the largest value such that the removal volume of the feature does not interfere with the part.

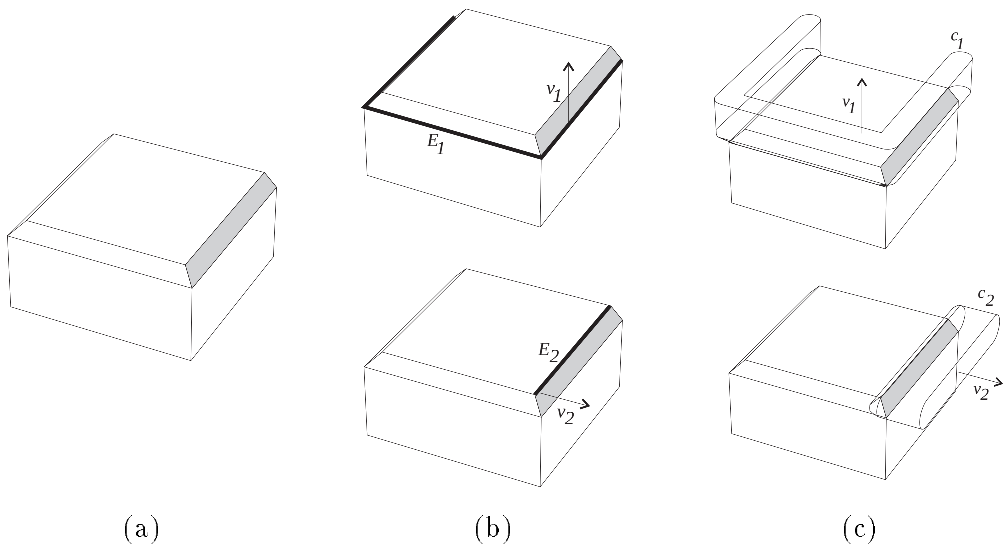


Figure 10: Finding instances of chamfering features.

The depth of the drilling feature is the minimal distance from its location to a point outside the workpiece along its orientation.

From a portion of a cylindrical side face of a drilling feature, one can determine the radius and orientation of a primary drilling feature. In the event that the surface was produced by a hole extending through the part, there are two possible primary feature instances: one in each direction along the axis of the cylindrical surface. For non-through features (those accessible in only one direction) the location for the primary feature instance can be found from the end surface, if one exists, or by calculating the deepest point at which the conical tip of the drilling tool may be placed without intersecting the part.

4.2 Constructing Face-Milling, Step-Milling and Pocket-Milling Features.

Construction of a milling feature starts at an edge of the part, e_1. Each edge e_1 in the part has the potential of belonging to three different types of feature instance:

1. As pictured in Figure 9(a), edge e_1 is an edge of one of the bottom surfaces of a milling feature (i.e., e_1 could have been created as part of the planar bottom face or a blend surface of the feature).

2. As pictured in Figure 9(b), edge e_1 could be an edge of a side surface of a milling feature having no bottom surface present in the part.

3. As pictured in Figure 9(c), edge e_1 could be a subset of an edge of a side surface of a milling feature which extends through the part. This type of feature is often called a through pocket.

The orientation of the milling feature is determined from the edges e_1 and e_2, where e_2 is another distinct coplanar edge in the boundary of P. For a given e_1, there are in the worst case $O(n)$ possible orientations for a primary milling feature, where n is the number of edges in the part P. Through pockets, because they can be milled from either of two setup directions, are modeled as two features with opposite orientations.

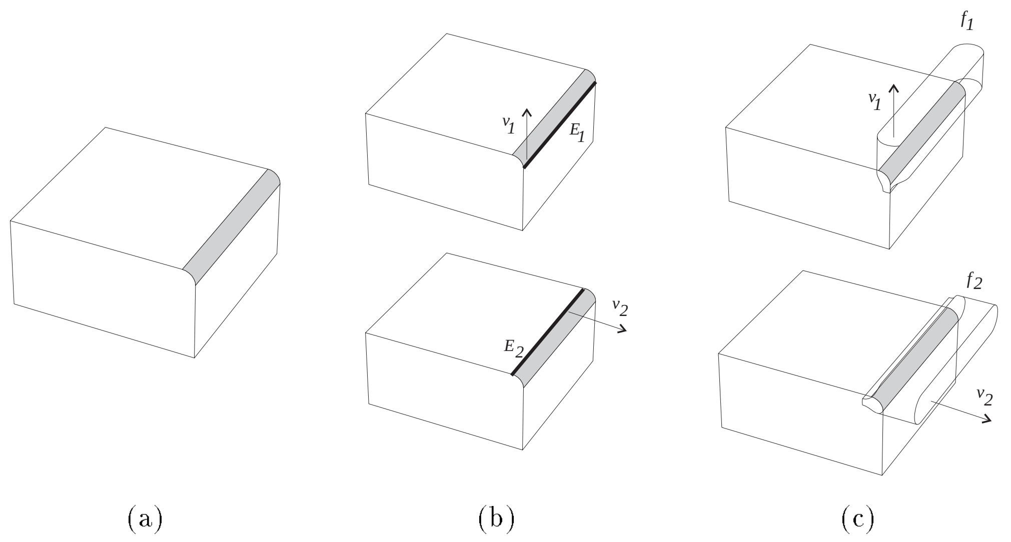


Figure 11: Finding instances of filleting features.

The profile of the milling feature can be computed as a normal projection of the part faces that lie in the half-space above (with respect to the orientation) the plane containing the bottom surface of the feature. The projection is computed onto the plane containing the edges e_1 and e_2 (shown in Figure 14(a)). In the first and second cases, this plane is the bottom surface of a milling feature that creates the edges and their adjacent surface. Note that this still applies when the bottom surface of the feature has been eliminated through interactions with other features and no longer exists in the model of the part P. In the event that e_1 or e_2 belong to a surface that is a bottom blend for the milling feature, then the edge profile is adjusted to take into account the blend radius of the milling tool.

In the third case, the feature extends through the part and thus has no bottom surface present in the delta volume. For this situation, the plane containing e_1 and e_2 provides the orientation vector $\vec{v}$ for the through feature. The part faces are mapped onto a projection plane perpendicular to $\vec{v}$, arriving at a cross-section of the through feature capable of creating these edges and surfaces.

Islands are found from the intersection of the face covering E with the part. Lastly, bottom blends are incorporated by examining the surfaces of P adjacent to E for candidate surfaces. If any blend surfaces are found, the profile is adjusted and the feature volume modified. Based on the edge profile and feature orientation, E and $\vec{v}$, the milling feature can be classified as pocket-milling, face-milling, or step-milling.

4.3 Constructing Chamfering Features and Filleting Features.

Given a surface s that is a portion of a face of a chamfering or filleting feature as shown in Figures 10(a) and 11(a), we construct a feature instance as follows:

1. Determine the set of possible orientations for feature instances that may have made the surface. For chamfering features, this set contains each vector $\vec{v}$ such that $\vec{v}$ is perpendicular to an edge of the delta volume and the angle between $\vec{v}$ and s is one of the available tip angles for chamfering tools. For filleting features, this set contains each vector $\vec{v}$ that is tangent to s and is perpendicular to an edge of the delta volume.

2. For each possible orientation, find the edge profile E for f and find the adjoining surfaces made by the operation. Figures 10(b) and 11(b) indicate the orientations for a chamfering tool and filleting tool.

3. Construct the feature instances, as shown in Figures 10(c) and 11(c).

4.4 Computational Complexity

Some approaches to feature recognition are based on data structures abstracted from a solid model of a part, such as graph-based methods [7, 18, 4, 5]. For such approaches, the computational cost can be calculated by counting the basic operations that need to be performed by the procedure. Peters [31] uses a similar approach to compute abstract complexity bounds on instances of the feature recognition problem itself.

Other approaches to feature recognition, including the one described in this paper, employ extensive queries to the solid modeling system to extract feature instances. In this case, the complexity of feature recognition algorithms depends on the cost incurred by executing queries to the solid modeler—and this in turn depends on many implementation-specific details such as data representation and overhead costs of the solid modeler. Therefore, we describe the complexity of our feature recognition algorithms simply in terms of the number of solid modeling operations required.

Let n be the number of edges in the boundary representation of the delta volume. Then, in general, the number of entities in the boundary representation is $O(n)$.[4] The algorithm Build_Well-Behaved_Features considers each of these n entities. At each entity, individual well-behaved feature instances are constructed. Each aspect of the delta volume's boundary representation can belong to at most $O(n)$ well-behaved features. Constructing each of these feature instances involves a constant number of calls to solid modeling operations. Thus the overall complexity of the algorithm to Build_Well-Behaved_Features is $(O(n^2 g(n)))$, where $g(n)$ is an upper bound on the complexity of individual solid modeling operations.

While there is no authoritative reference on the general complexity of solid modeling operations such as booleans, sweeps, and the like, consensus appears to be that the complexity of boolean operations lie between $O(n^2)$ and $O(n^4)$ or $O(n^5)$ time, depending on many implementation-specific details. Thus the complexity of Build_Well-Behaved_Features is between $O(n^4)$ and $O(n^6)$ or $O(n^7)$.

4.5 Alternatives and Completeness

It has been pointed out by Marefat [27, 28] that existing feature recognition methodologies have had only limited success in identifying and describing alternative feature interpretations. There are a variety of reasons for this shortcoming. For example, since features can intersect with each other, the introduction of a new feature into a design can divide other features into spatially disjoint component—components which may be computationally expensive to identify and recombine. This poses difficulty for traditional approaches: rule-based methods must capture all geometric situations that arise from the choice of feature hints and the ambiguities inherent in manipulating multiple

[4] In the worst case for this class of manifold parts and for these boundary data structures, we can say the size is $O(|E|+|V|+|F|)$ where E, S, and F are the sets of edges, vertices, and faces of Δ respectively. By Euler's equation $2=|V|-|E|+|F|$, we can simplify this to be $|V|+|E|+|F|=2+2|E|$ or $O(|E|)$—where $|E|$ is the number of edges of Δ.

interpretations with many separate rules; graph-based algorithms must syntactically or structurally capture these complexities.

In the existing literature, there have been several efforts at generating the complete class of alternative interpretations for a part. The feature algebra of Karinthi [19], starting from a single initial feature interpretation, exhaustively generates alternative interpretations of the part by manipulating the features with algebraic operators, but does not include a methodology for recognizing the features. Sakurai [39] presents a system that decomposes the volume to be machined into disjoint cells and then recombines the cells to form compound feature instances. Both of these methods are prone to combinatorial obstacles, are limited to polyhedral models, and address only rudimentary machining features and machining constraints.

In order to address this problem, we will define a feature recognition algorithm to be *complete* over some class of features $\mathcal{T}$ if for any given part P it produces the set of all features in $\mathcal{T}$ that appear in P.[5] This has the following immediate consequences:

- It is impossible for a feature recognition algorithm to be complete over the set of machining features $\mathcal{M}$, because there are parts for which there are an infinite number of possible instances of machining features.

- Even if we restrict ourselves to primary features, completeness is still impossible: there are simple machinable parts that can have infinitely many primary features.

- There are only finitely (in fact, polynomially) many well-behaved features for any given part. Thus completeness over the set of well-behaved features is an attainable goal.

An argument that the procedure BUILD_WELL-BEHAVED_FEATURES is complete over the class of well-behaved features can be made as follows. Given a part P, a piece of stock S, and a well-behaved feature instance f from P, f contributes some unique geometric and topologic characteristics to the boundary of P, and BUILD_WELL-BEHAVED_FEATURES makes use of them to reconstruct f. For each geometric and topological attribute of P, all features capable of producing that entity are eventually produced. More specifically, there are three possible cases:

1. *f is a drilling feature.* Then the boundary of the delta volume, $b(\Delta)$, must contain a portion of f's cylindrical side face or conical end face. BUILD_WELL-BEHAVED_FEATURES uses this face portion to determine the proper orientation and other parameters for f (as described in Section 4).

2. *f is a milling feature.* Then since f is well-behaved, $b(f) \cap^* b(\Delta)$ must contain two edges, e_1 and e_2, that are perpendicular to f's orientation. BUILD_WELL-BEHAVED_FEATURES checks all of the possibilities for e_1 and e_2, thus finding both the proper orientation for f and a location from which to determine the edge profile for f. The remainder of the parameters for feature f are calculated using the orientation and the edge profile.

3. *f is a chamfering or filleting feature.* Then the edges of f's edge profile are edges in $b(\Delta)$ and $b(f) \cap^* b(\Delta)$ is non-empty. By examining the edges and faces in $b(\Delta)$ as described in Section 4, an orientation for f is calculated. Using edge profile E, the other surfaces made by the machining operation are found by examining the part topology, and the volumetric feature instance is built.

[5]A formal proof of completeness of a feature recognition procedure would be quite complicated. It would require mathematically rigorous definitions of features and the feature recognition problem, and a proof that for every instance of the problem, the algorithm halts and returns the correct answer. This is discussed in more detail in [35].

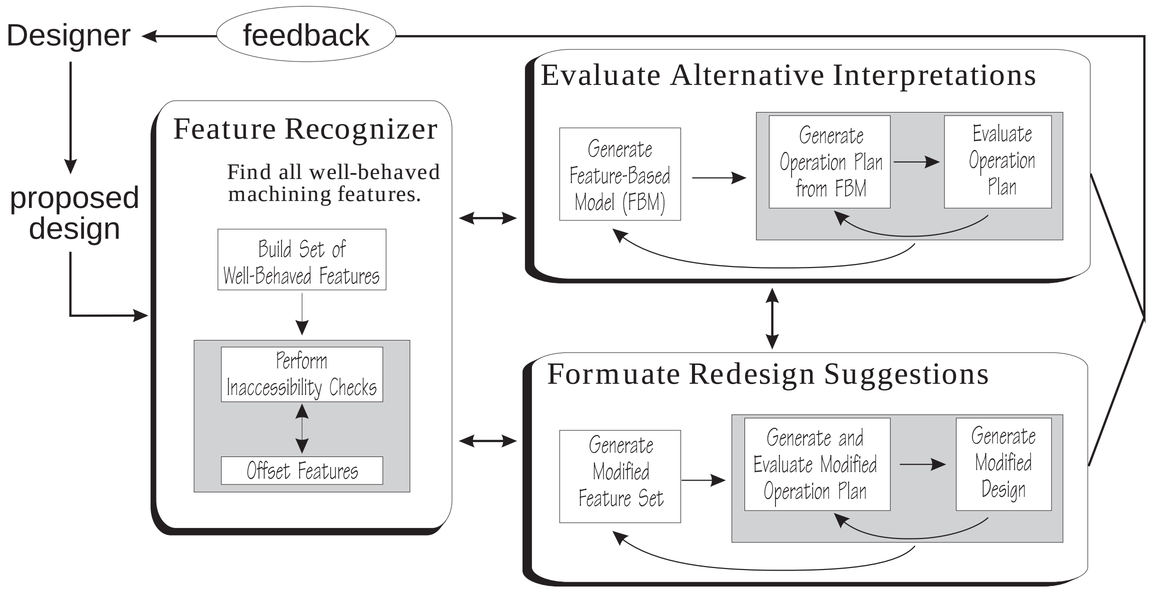

Designer
feedback
proposed design
Feature Recognizer
Find all well-behaved machining features.
Build Set of Well-Behaved Features
Perform Inaccessibility Checks
Offset Features
Evaluate Alternative Interpretations
Generate Feature-Based Model (FBM)
Generate Operation Plan from FBM
Evaluate Operation Plan
Formuate Redesign Suggestions
Generate Modified Feature Set
Generate and Evaluate Modified Operation Plan
Generate Modified Design

Figure 12: Feature recognition within the IMACS system.

4.6 Feature Offsetting

Offsetting the edge profile of a potential milling feature involves the following steps:

1. **Estimation of an optimal tool size.** In a typical milling operation, a larger tool diameter implies a shorter cutting trajectory and less operation cost and time. However, a variety of constraints resulting from the geometric configuration of the profile will restrict the maximum tool size that can be employed. In this step, the geometry of the profile is used to calculate an upper bound on the tool size.

2. **Alteration of the profile.** In some profiles, the estimation of tool size may reveal machinability problems. For example, two adjacent closed profile edges meeting at a convex corner results in a tool radius estimate of zero; a narrow distance between closed edges in the profile may return an estimate smaller than the smallest available tool. This step modifies profiles by offsetting convex corners inward to account for the corner radius left by a tool (shown for the closed edges in Figures 8(b) and (c)) or by dividing an otherwise unmachinable profile into a set of multiple profiles that can be machined with the available milling tools.

3. **Offsetting the profile.** After finding a usable bound on the tool size, the open edges of a milling feature are offset to account for the radius of the milling tool, shown for the open edges in Figures 8(b) and (c). The tool can move on or outside these edges during machining.

A more detailed presentation of feature offsetting appears in [35, 12].

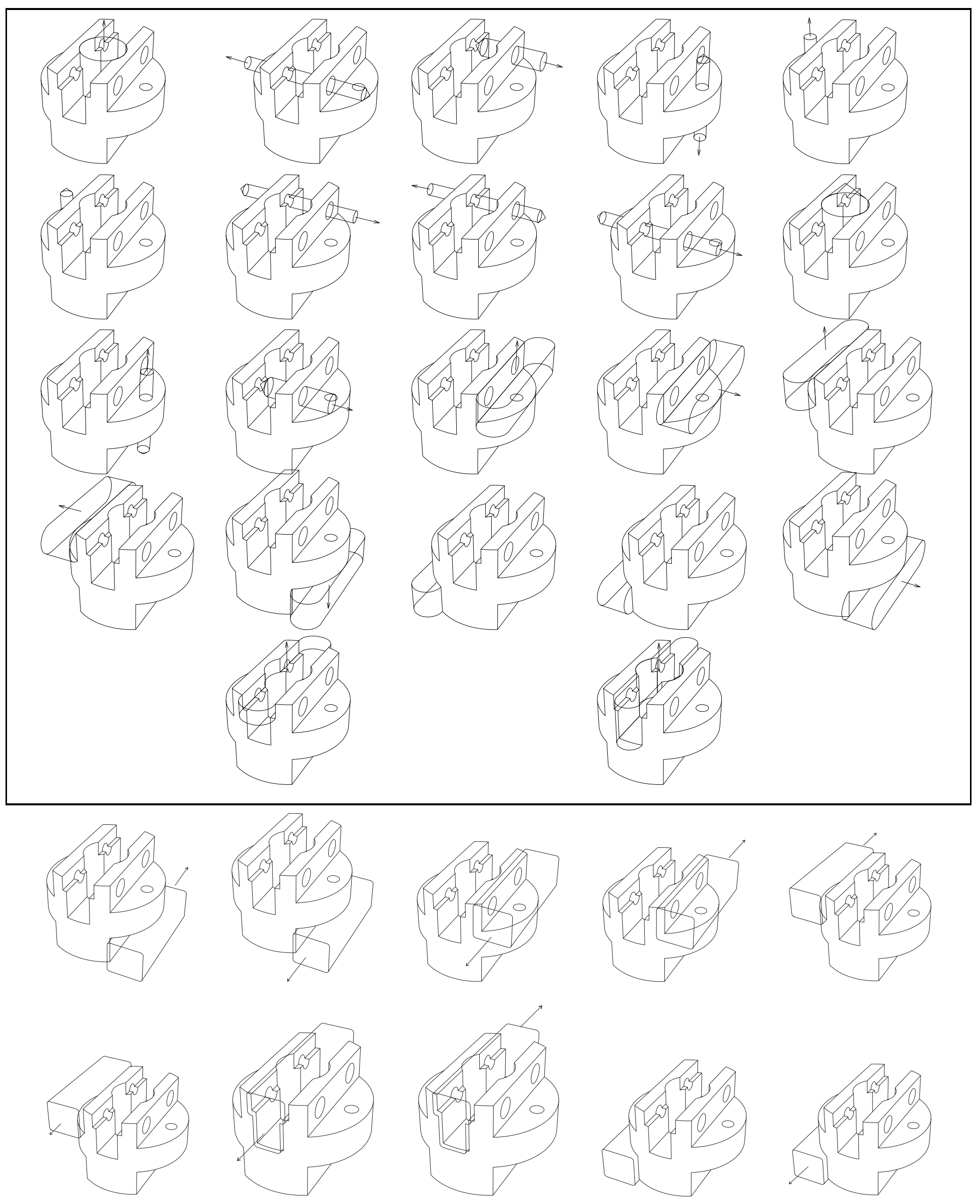
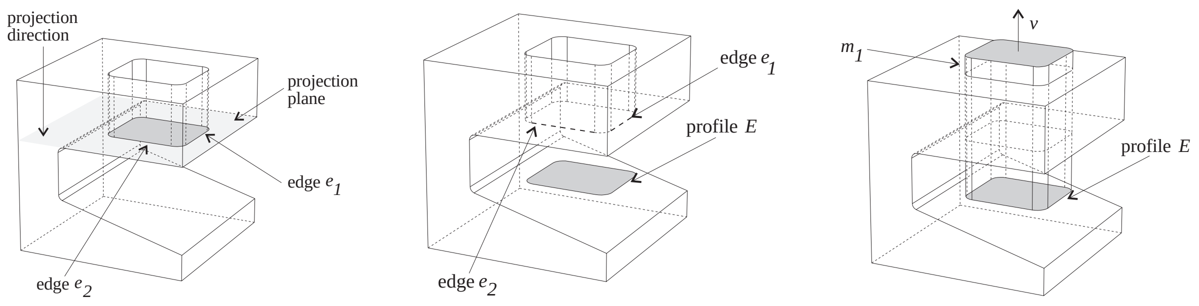


Figure 13: The features identified by BUILD_FEATURES for the part shown in Figure 1(b). Those features occuring in feature-based models are denoted within the box.



(a): Find edge profile (b): Locate edge profile (c): Build feature instance

Figure 14: Steps in recognizing a bottomless pocket-milled feature.

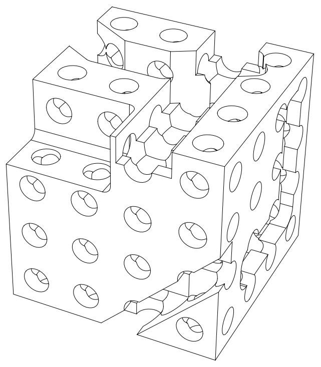


Figure 15: A part with a variety of feature interactions.

5 Implementation and Examples

As an instance of this feature recognition methodology, we have built a proof-of-concept implementation in C++ using version 3.0.1 of the AT&T C++ compiler from SUN Microsystems running on a SPARCStation model 10-30 workstation; version 1.5.1 of Spatial Technologies' ACIS© Solid Modeling Kernel; and version 3.14 of the NIH C++ Class Library from the National Institutes of Health. Also being employed in our development efforts are Ithaca Software's HOOPS© Graphics System and the Tcl/Tk embeddable command language and user interface toolkit developed at the University of California at Berkeley.

This recognition algorithm is being incorporated into the Interactive Manufacturability Analysis and Critiquing System (IMACS) under development at the University of Maryland's Institute for Systems Research. Within IMACS, as illustrated in Figure 12, the role of the feature recognition module is to produce the set of well-behaved features from the part's geometry and topology. In the current version of the implementation, we have omitted bottom blended surfaces and some instances of chamfer and fillet features as they were non-essential cases for current application requirements.

The second IMACS module uses this set of features to generate and evaluate alternative operation plans for the part by incorporating precedence constraints and information about machining parameters. The cost and time for operation plans satisfying the design requirements are used to estimate a rating of the part's manufacturability [15].

A third module formulates redesign suggestions based on plan information and the set of well-

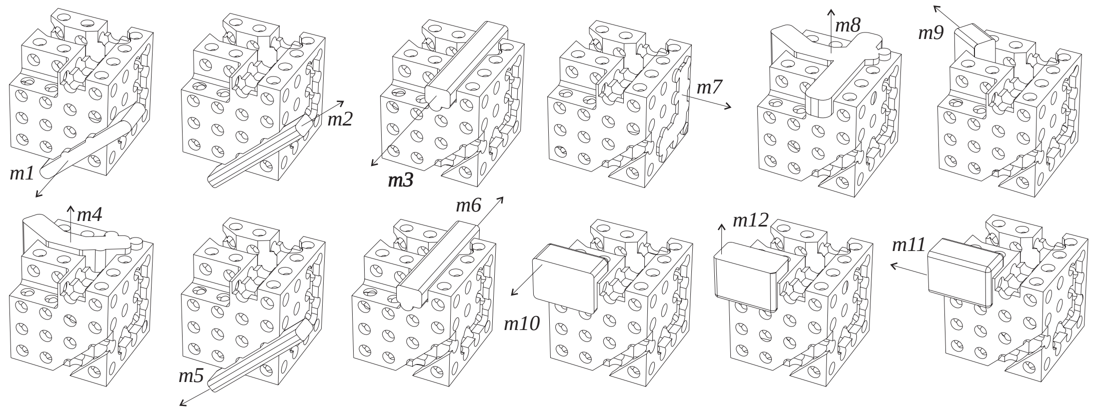


Figure 16: Twelve milling features from Figure 15.

behaved features. By making modifications to the operations in the previously generated plans, it creates modified versions of the design that, in addition to satisfying design requirements, have improved manufacturability [6]. The new designs are presented to the designer as alternative possibilities to be considered.

The following examples illustrate some of the functionality of our approach:

Example 1 In the case of the part in Figure 1, the 32 offset, accessible well-behaved features are pictured in Figure 13.

Example 2. One of the more difficult feature instances to recognize is that of a pocket-milled feature whose bottom surface has been removed through interaction with other features. Figure 9(b), presents a part with interacting milling features. In this case, the bottom surface of the pocket-milled feature has been eliminated through interaction with other features.

For this feature, the orientation is determined from the edges e_1 and e_2. A profile for the feature is found through the projection of the part faces along the normal of the plane passing through e_1 and e_2, as shown in Figure 14(a). One projection direction along the normal yields a non-empty edge profile E; the location for the bottom surface of the pocket-milling feature can then be calculated (Figure 14(b)). The edge profile E can then be swept to produce an instance of a primary pocket-milling feature m_1, shown in Figure 14(c).

Example 3. Figure 15 illustrates a complex solid with a variety of volumetric feature interactions. While itself not a particularly useful component, it does present some aspects of the geometric and topological complexities to be found in realistic components. In particular, this example has 346 faces in P and 413 faces in Δ. This part, considered with a rectangular block of stock material, yields 106 well-behaved drilling and twelve well-behaved milling features. The twelve milling features are shown in Figure 16.

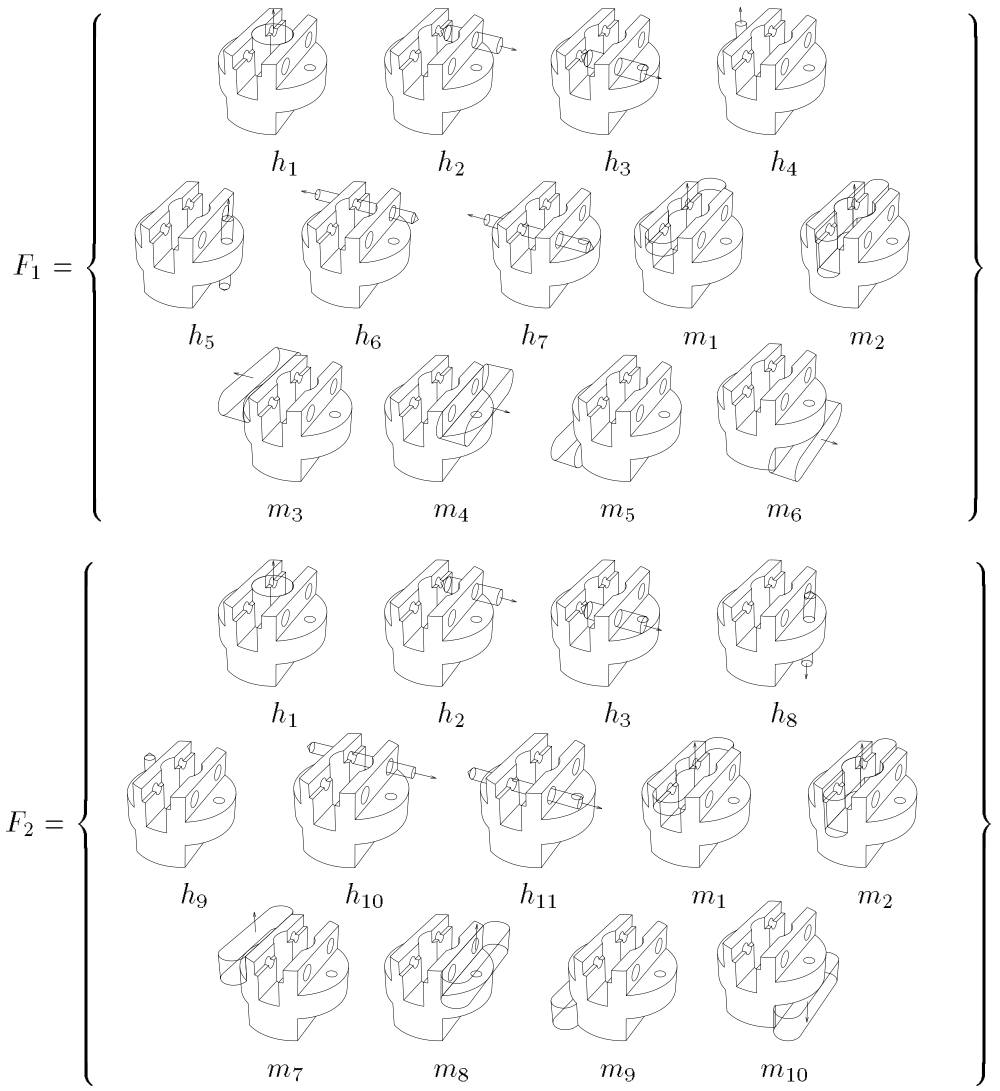


Figure 17: Two alternative feature-based models for the part in Figure 1(b).

6 Feature-Based Models

As mentioned in Section 5, the features produced by the BUILD_FEATURES algorithm given in Section 4 are used as input to a module that uses these features to generate and evaluate alternative operation plans for the machining of the part. A key step in generating alternative operation plans is to generate alternative feature-based models, which are sets of features that correspond to possible ways to machine the part.

A *feature-based model* (FBM) is a finite set of feature instances $F = \{f_1, f_2, f_3, \ldots, f_n\}$ (in this case from the class of machining features $\mathcal{M}$), that describe the volume of material to be removed by a set of machining features to create a part P from a piece of stock S. More specifically, an FBM is any finite set of machining features with the following properties:

1. *Sufficiency*: the features in F describe one possible way to interpret Δ: $S -^* P \subseteq \bigcup_{f \in F} f$.
2. *Necessity*: no proper subset of F contains Δ: for all $f_i \in F, S -^* P \not\subseteq \bigcup_{f \in (F - \{f_i\})} f$. In this way, an FBM does not contain redundant features and each feature of F contributes to the interpretation of the part.
3. *Validity*: in general, validity would require that f meet machinability requirements such as those for accessibility, fixturability, and tolerances. However, development of a general methodology that simultaneously considers all of these issues is beyond the scope of this work. Thus, in this paper, we will consider a feature to be valid if it is not inaccessible as discussed in Section 3.4, and if the volume described by f does not intersect the part P; i.e., for all f_i in F, $f_i \cap^* P = \emptyset$.

There may be many FBMs of P and S, each corresponding to a different interpretation of the part P as a subset of the set of well-behaved features $\mathcal{F}$ found by BUILD_FEATURES. A particular F need not model the optimal way of creating the design as there may exist many alternatives, each corresponding to a different collection of machining operations that could be used to produce the design from a given piece of stock material.

As an example, for the part P pictured in Fig. 1(b), $\mathcal{F}$ contains 32 feature instances, as shown in Figure 13. These features can be used to generate 512 different FBMs for the part; Figure 17 shows the two of the possible FBMs:

$$F_1 = \{h_1, h_2, h_3, h_4, h_5, h_6, h_7, m_1, m_2, m_3, m_4, m_5, m_6\}$$

and

$$F_2 = \{h_1, h_2, h_3, h_8, h_9, h_{10}, h_{11}, m_1, m_2, m_7, m_8, m_9, m_{10}\}.$$

Generation and evaluation of operation plans is performed by generating FBMs and determining possible orders in which the features in the FBM might be machined. Since each FBM is basically an irredundant set cover of the delta volume produced from the feature alternatives, models can be generated using variations on irredundant-set-covering techniques [34, 29], using pruning heuristics to discard unpromising FBMs. The generation and evaluation of machining alternatives is discussed in greater detail in [15, 13].

7 Discussion and Conclusions

In this paper, we have presented an algorithmic approach for recognition of a class of machinable features from solid models. Potential applications for our approach include process planning, NC part programming, fixture design and selection, and manufacturability evaluation. We are incorporating this approach into a system for automatically analyzing designs and providing feedback to the designer about their manufacturability: the *IMACS* design analysis and critiquing system under development at the University of Maryland's Institute for Systems Research.

Some of the distinguishing characteristics of our approach include:

1. The feature recognition algorithm is complete over our class of features, regardless of whether the features intersect with each other in complex ways. Knowing the limits on completeness is useful for applications such as manufacturability analysis, in which determining the manufacturability of a design may require trying many alternative feature-based models to calculate which provide the best operation plans.

2. Our approach incorporates characteristics to aid in downstream applications such as manufacturability evaluation and process planning: it handles features created by a variety of machining operations, including drilling, milling, chamfering, blending, and filleting; when possible, the features recognized are offset to account for the dimensions of the cutting tool to be used; and inaccessible features are discarded.

3. The algorithm's worst-case time complexity is polynomial in the number of edges in the delta volume and requires only a quadratic number of solid modeling operations. This represents an improvement over approaches that employ techniques that include exponential-time algorithms, and those that attempt to produce all of the often exponential number of feature-based models during the recognition phase. It should be noted that some of these existing approaches include the complex computations related to downstream applications as part of the recognition phase.

Some of the issues not yet addressed by this work include the following:

1. It would be valuable to investigate whether or not the completeness and complexity results can be extended to more complex feature types (such as composite features or feature groups) or to more diverse manufacturing domains and processes.

2. In recognizing features, our approach currently uses only geometric and topological information. It will be important to eventually incorporate the use of information such as tolerances, design features, functional requirements, and data relating to design history and intent.

Future goals include extending our results to include other, more complex, features; adapting the implementation to operate incrementally (i.e., making the necessary changes to the set of feature alternatives as the designer interactively makes design modifications); and exploring how to reduce the time complexity through simplification of the CAD model while still maintaining completeness.

Acknowledgements

This work was supported in part by NSF Grants NSFD CDR-88003012 and DDM-9201779.

References

[1] Tien-Chien Chang. *Expert Process Planning for Manufacturing.* Addison-Wesley Publishing Co., 1990.

[2] S. H. Chuang and M. R. Henderson. Three-dimensional shape pattern recognition using vertex classification and the vertex-edge graph. *Computer Aided Design*, 22(6):377–387, June 1990.

[3] S. H. Chuang and M. R. Henderson. Compound feature recognition by web grammar parsing. *Research in Engineering Design*, 2(3):147–158, 1991.

[4] J. Corney and D. E. R. Clark. Method for finding holes and pockets that connect multiple faces in $2\frac{1}{2}$d objects. *Computer Aided Design*, 23(10):658–668, December 1991.

[5] J. Corney and D. E. R. Clark. Face based feature recognition: Generalizing special cases. *International Journal of Computer Integrated Manufacturing*, 6(1 & 2):39–50, 1993.

[6] Diginta Das, Satyandra K. Gupta, and Dana S. Nau. Reducing setup cost by automated generation of redesign suggestions. In Kosuke Ishii, editor, *ASME Computers in Engineering Conference*, pages 159–170. ASME, September 1994.

[7] Leila De Floriani. Feature extraction from boundary models of three-dimensional objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(8), August 1989.

[8] Xin Dong. *Geometric Feature Extraction for Computer-Aided Process Planning.* PhD thesis, Rensselaer Polytechnic Institute, Troy, NY, USA, 1988.

[9] R. Gadh and F. B. Prinz. Recognition of geometric forms using the differential depth filter. *Computer Aided Design*, 24(11):583–598, November 1992.

[10] P. Gavankar and M. R. Henderson. Graph-based extraction of protrusions and depressions from boundary representations. *Computer Aided Design*, 22(7):442–450, September 1990.

[11] S. K. Gupta, D. S. Nau, W. C. Regli, and G. Zhang. A methodology for systematic generation and evaluation of alternative operation plans. In Jami Shah, Martti Mäntylä, and Dana Nau, editors, *Advances in Feature Based Manufacturing.* Elsevier/North Holland, 1993.

[12] Satyandra K. Gupta. *Automated Manufacturability Analysis of Machined Parts.* PhD thesis, The University of Maryland, College Park, MD, 1994.

[13] Satyandra K. Gupta, Thomas R. Kramer, Dana S. Nau, William C. Regli, and Guangming Zhang. Building MRSEV models for CAM applications. *Advances in Engineering Software*, 1994. To appear.

[14] Satyandra K. Gupta, William C. Regli, and Dana S. Nau. Manufacturing feature instances: Which ones to recognize? Technical Report CS-TR-3376, UMIACS-TR-94-127, ISR-TR-94-81, The University of Maryland, College, Park, November 1994.

[15] S.K. Gupta and D.S. Nau. A systematic approach for analyzing the manufacturability of machined parts. *Computer Aided Design*, 1994. To appear.

[16] Mark R. Henderson. *Extraction of Feature Information from Three-Dimensional CAD Data.* PhD thesis, Purdue University, West Lafayette, IN, USA, 1984.

[17] Christoph M. Hoffman. *Geometric and Solid Modeling: An Introduction.* Morgan Kaufmann Publishers Incorporated, CA, 1989.

[18] S. Joshi and T. C. Chang. Graph-based heuristics for recognition of machined features from a 3D solid model. *Computer-Aided Design*, 20(2):58–66, March 1988.

[19] R. Karinthi and D. Nau. An algebraic approach to feature interactions. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 14(4):469–484, April 1992.

[20] Y. S. Kim. Recognition of form features using convex decomposition. *Computer Aided Design*, 24(9):461–476, September 1992.

[21] Yong Se Kim and D. J. Wilde. A convergent convex decomposition of polyhedral objects. *Transactions of the ASME*, 114:468–476, September 1992.

[22] Thomas R. Kramer. A parser that converts a boundary representation into a features representation. *International Journal of Computer Integrated Manufacturing*, 2(3):154–163, 1989.

[23] Thomas R. Kramer. A library of material removal shape element volumes (MRSEVs). Technical Report NISTIR 4809, The National Institute of Standards and Technology, Gaithersburg, MD 20899, March 1992.

[24] Lycourgos K. Kyprianou. *Shape Classification in Computer Aided Design*. PhD thesis, Christ College, University of Cambridge, Cambridge, United Kingdom, 1980.

[25] Timo Laakko and Martti Mäntylä. Feature modelling by incremental feature recognition. *Computer Aided Design*, 25(8):479–492, August 1993.

[26] Martti Mäntylä. *An Introduction to Solid Modeling*. Computer Science Press, College Park, MD, 1988.

[27] M. Marefat and R. L. Kashyap. Geometric reasoning for recognition of three-dimensional object features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):949–965, October 1990.

[28] Michael Marefat and R. L. Kashyap. Automatic construction of process plans from solid model representations. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(5):1097–1115, September/October 1992.

[29] Yun Peng and James A. Reggia. Diagnostic problem-solving with causal chaining. *International Journal of Intelligent Systems*, 2:265–302, 1987.

[30] Thomas J. Peters. Encoding mechanical design features for recognition via neural nets. *Research in Engineering Design*, 4(2):67–74, 1992.

[31] Thomas J. Peters. Mechanical design heuristics to reduce the combinatorial complexity for feature recognition. *Research In Engineering Design*, 4:195–201, 1993.

[32] J. Miguel Pinilla, Susan Finger, and Friedrich B. Prinz. Shape feature description using an augmented topology graph grammar. In *Proceedings NSF Engineering Design Research Conference*, pages 285–300. National Science Foundation, June 1989.

[33] S. Prabhakar and M. R. Henderson. Automatic form-feature recognition using neural-network-based techniques on boundary representations of solid models. *Computer Aided Design*, 24(7):381–393, July 1992.

[34] J. A. Reggia, D. S. Nau, and P. Y. Wang. A formal model of diagnostic inference. II. algorithmic solution and applications. *Information Sciences*, 37:257–285, 1985.

[35] William C. Regli. *Geometric Algorithms for Recognition of Features from Solid Models.* PhD thesis, The University of Maryland, College Park, MD, 1995. In preparation.

[36] Aristides A. G. Requicha. Representation for rigid solids: Theory, methods, and systems. *Computing Surveys*, 12(4):437–464, December 1980.

[37] David W. Rosen, John R. Dixon, and Susan Finger. Conversions of feature-based representations via graph grammar parsing. In *AMSE Design Theory Methodology Conference*, 1992.

[38] Scott A. Safier and Susan Finger. Parsing features in solid geometric models. In *European Conference on Artificial Intelligence*, 1990.

[39] Hiroshi Sakurai and Chia-Wei Chin. Definition and recognition of volume features for process planning. In Jami Shah, Martti Mäntylä, and Dana Nau, editors, *Advances in Feature Based Manufacturing*, chapter 4, pages 65–80. Elsevier/North Holland, 1994.

[40] Hiroshi Sakurai and David C. Gossard. Recognizing shape features in solid models. *IEEE Computer Graphics & Applications*, September 1990.

[41] Jami Shah, Martti Mäntylä, and Dana Nau, editors. *Advances in Feature Based Manufacturing.* Elsevier/North Holland, 1994.

[42] J. H. Vandenbrande and A. A. G. Requicha. Spatial reasoning for the automatic recognition of machinable features in solid models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(12):1269, December 1993.

[43] Douglas L. Waco and Yong Se Kim. Geometric reasoning for machining features using convex decomposition. *Computer Aided Design*, 26(6):477–489, June 1994.

[44] Tony C. Woo. Feature extraction by volume decomposition. In *Conference on CAD/CAM Technology in Mechanical Engineering*, pages 76–94, March 1982.