한국 CAD/CAM 학회 논문집
제 3 권 제 1 호 1998년 3월 pp. 68-85

학술논문

Feature Recognition: the State of the Art

JungHyun Han*

ABSTRACT

Solid modeling refers to techniques for unambiguous representations of three-dimensional objects. Feature recognition is a sub-discipline focusing on the design and implementation of algorithms for detecting manufacturing information such as holes, slots, etc. in a solid model. Automated feature recognition has been an active research area in solid modeling for many years, and is considered to be a critical component for CAD/CAM integration. This paper gives a technical overview of the state of the art in feature recognition research. Rather than giving an exhaustive survey, I focus on the three currently dominant feature recognition technologies: graph-based algorithms, volumetric decomposition techniques, and hint-based geometric reasoning. For each approach, I present a detailed description of the algorithms being employed along with some assessments of the technology. I conclude by outlining important open research and development issues.

Key words : Solid Modeling, Feature Recognition, Computer-Aided Design (CAD), Manufacturing Process Planning

1. Introduction

Solid modeling refers to techniques for unambiguous representations of three-dimensional objects[1-3]. Its data structures and algorithms emerged in earnest in the early 1970s[4,5] and have since been used in a broad range of applications: computer-aided design and manufacturing (CAD/CAM), computer graphics and visualization, virtual reality, robotics, computer vision, etc. This paper presents the state of the art in *automated feature recognition* - arguably the most active research area among solid modeling applications.

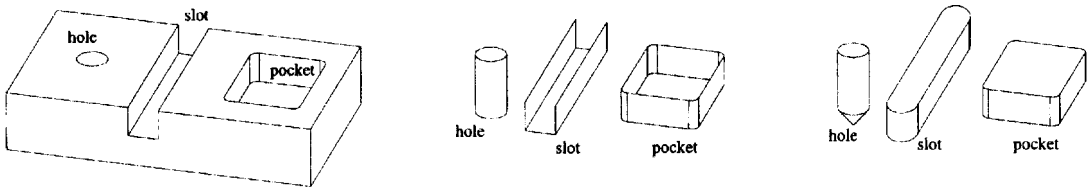
In the early 1960s Ivan Sutherland developed the SKETCHPAD system[6], the starting point for nearly all research in computer graphics. One of the first applications of this technology was in engineering design, and early CAD systems were essentially for two-dimensional drawing and drafting. However, the rise of solid modeling techniques has created a proliferation of sophisticated three-dimensional CAD systems in recent years.

On the other hand, numerically controlled (NC) machining was first introduced in the early 1950s, sparking research and development of algorithms for CAM. In industry, CAD and CAM are extensively used to assist in design and manufacture of products, respectively. However, effective CAD/CAM integration has been elusive, and extensive human intervention is still necessary to move ideas and designs between CAD and CAM in most manufacturing domains[7].

Computer Aided Process Planning (CAPP) is seen as a communication agent between CAD and CAM. Given CAD data of a part (a component of a product to be manufactured), the goal of CAPP is to generate a sequenced set of instructions used to manufacture the specified part. In order to do that, CAPP has to *interpret* the part in terms of *features*.

Informally, *features* are generic shapes or other characteristics of a part with which engineers can associate knowledge useful for reasoning about the part[8]. Fig. 1(a) shows feature examples: the part is interpreted in terms of a hole, a slot and a pocket. CAPP will use these features to generate manufac-

*종신회원, School of Electrical and Computer Engineering, Sung Kyun Kwan University, Suwon, 440-746 Korea



(a) part and features (b) surface features (c) (volumetric) machining features

Fig. 1. Feature examples.

turing instructions to produce the part. For example, CAPP typically generates a drilling operation for the hole.

Feature recognition is a front-end to CAPP and plays a key role in CAD/CAM integration. It is the process of converting CAD data of a part into a model of the manufacturing activities required to create the part. At the core of a part's CAD data is usually its solid model. Algorithms for feature recognition typically involve extensive geometric computations and reasoning about the solid model of the part.

This paper is organized as follows: Section 2 discusses feature representations, *feature models* and how feature recognition is used in feature model generation. There have been two decades of research on feature recognition since the seminal work of Kyprianou in 1980[9] and the literature on feature recognition is voluminous. Rather than attempting to exhaustively cover the history of the field, this paper discusses in a great detail the three currently most active approaches: graph-based algorithms, volumetric decomposition, and hint-based reasoning. Sections 3, 4 and 5 present the state of the art of these three approaches, respectively. (For wide but superficial surveys, see[10-12].) Section 6 raises important open research issues in feature recognition, and finally Section 7 draws conclusions.

2. Features

2.1 Machining features

Three dominant solid representations in use today are Constructive Solid Geometry (CSG), Boundary Representation (BRep) and Spatial Subdivision[1-3]. As the solid representation of the input part for feature recognition systems, BRep has been dominant. It is because, unlike other representations such as CSG, a BRep uniquely defines the entities, e.g. faces/edges/vertices, of a solid[13], and so searching for BRep entity patterns is more promising than searching for CSG patterns, etc.[14]. A feature can be represented as a collection of BRep faces of the part, called a **surface feature**. Fig. 1(b) shows examples of surface features.

The application domain that has received most of the attention of feature recognition researchers is machining. This paper focuses on machining feature recognition. In machining, a feature attempts to capture the effect of a cutting operation, and can be defined as a surface feature: a collection of BRep faces that are to be created by a machining operation. In contrast, a machining feature can also be defined as a **volumetric feature** representing volume swept by the cutting surfaces of the cutting tool during machining. In early days of feature recognition research, machining features were mostly represented as surface features. In recent years, however, it has become increasingly evident that volumetric features (often augmented with surface features) provide more comprehensive representations of the actual machining operations than surface features. Section 3 will demonstrate an example of deficiencies in surface representation of a feature. Volumetric features are becoming the norm in the current generation of feature-based systems.

Fig. 2 shows cutter abstractions and volumetric machining features defined in[15]. A hole is typically generated by a vertical sweep of a drilling cutter with a conical end. A hole feature is then the volume swept by such a drilling cutter, as shown in Fig. 2(a). Unlike holes, slot and pocket

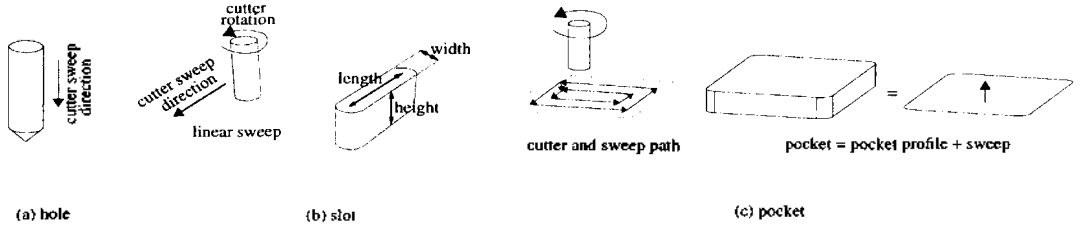


Fig. 2. Feature definition examples.

features are made by milling cutters. A slot is usually machined by a single linear sweep of a cylindrical end-milling cutter. A slot feature is the volume swept by the cutter motion, i.e. an elongated parallelepiped with rounded ends, as shown in Fig. 2(b). A pocket is machined by a series of cuts with an end milling cutter, as depicted in Fig. 2(c). The pocket feature is then represented by a swept volume of an arbitrarily-shaped planar *profile* (*floor*) along a vector. According to these definitions, the example part shown in Fig. 1(a) may have three volumetric machining features shown in Fig. 1(c). Fig. 3 shows an industrial part and its decomposition into 9 features, obtained from[15]. (In this figure, recognized pockets have convex corners unlike the pocket defined in Fig. 2(c). Section 6.3 explains why.)

2.2 Feature model generation

A unique representation of a part in terms of features is often called a *feature model* or an *interpretation* of the part[16,17]. There are essentially two ways of creating a feature model: *feature recognition* and *feature based design*, as depicted in Fig. 4. When a part is designed through the customary solid modeling operations, feature recognition is required to generate *manufacturing features*. In contrast, feature based design allows the designer to use features, called *design features*, as building blocks to create a part.

In some feature based design systems, design features correspond directly to specific manufacturing operations. For example, there are several prototype systems based on the *design-by-machining-features* approach, including Quick Turnaround Cell (QTC)[18], NEXT-Cut[19] and the University of California at Berkeley's Cybercut[20]. In the commercial world, Parametric Technologies' Pro/ENGINEER© and Bentley Systems' MicroStation Modeler© CAD packages are heavily dependent on the idea of designing with parametric machining features such as holes, swept profiles, etc.

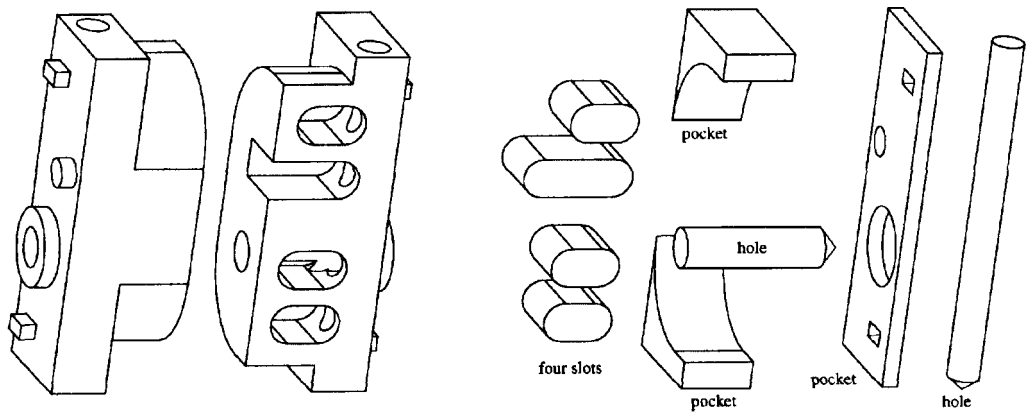


Fig. 3. A part and its decomposition into features.

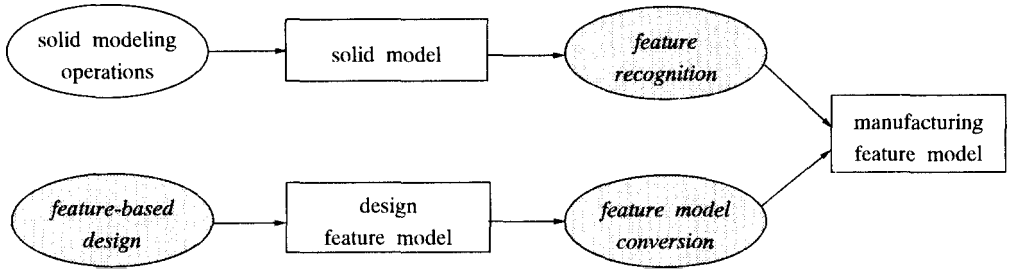


Fig. 4. Feature model generation.

When designers model the design directly in terms of machining features, the need to perform feature recognition might be eliminated - the final design includes a machining feature model. This benefit aside, there are several well-known limitations. First of all, in the majority of cases, the features that are most natural for use during the design phase are not machining features. An example of discrepancy between a design feature model and a machining feature model is shown in Fig. 5. This part may be designed by adding a rib to the base block as illustrated in the design feature model of (b). However, the machining feature model for the part may be defined by subtracting two steps (pockets) from the stock, which is a block sufficiently large to enclose the part, as shown in (c). We should not force designers to create the machining feature model shown in (c). Instead, we should convert whatever is created by the designers into a machining feature model. The process of converting a feature model in a domain (e.g. a design feature model) into a feature model in another domain (e.g. a machining feature model)

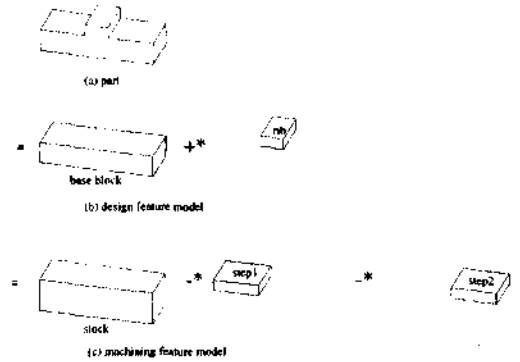


Fig. 5. A part with different feature models.

is called *feature model conversion*, as depicted in Fig. 4. (A comprehensive survey of feature model conversion approaches is given in[21].)

The second problem in the design-by-machining-features approach is related to the existence of multiple feature models. As described in Section 6.2, there are often multiple ways of interpreting a part in terms of machining features. The design-by-machining-features approach heavily assumes that the designers are going to specify a part using a set of features which is best for machining. This is often an unrealistic possibility in domains such as conventional machining in which simple parts may have many feature models. The feature model created by a designer is not necessarily the best for machining.

The most flexible design approach is to allow the designer to use whatever techniques are convenient for describing a part. A feature based design system may provide a rich library of feature primitives, a powerful ability to modify and combine these primitives, and some capability for user-defined features[22]. However, designers may not want to design a part in terms of features only. Most commercial CAD systems with feature based design capability provide an environment where both feature operations and solid modeling operations can be used in parallel during the design of a part. In this sense, Fig. 4 does not cover all possible design scenarios.

The evolving consensus is that (i) design and machining features are often distinct, (ii) design should be done in terms of design features or solid modeling operations, and (iii) the part model (which may be a design feature model, a solid model, or a

combination of both) should be converted into a feature model useful for the manufacturing or analysis task at hand. When the designer creates a part through both feature modeling and solid modeling, feature recognition is indispensable for generating a machining feature model. Even when a part is designed exclusively in terms of features and therefore feature model conversion is to be performed, geometric reasoning is required when direct mapping from design features to machining features is not possible. Such a geometric reasoning largely coincides with feature recognition. The boundary between feature recognition and feature model conversion is vague[23], and the algorithms for feature recognition play a key role in feature model conversion.

The following sections critically survey three dominant approaches in feature recognition for machining applications. For each approach, both descriptions of the geometric algorithms and discussions of their capabilities and limitations are presented in detail. Every section is organized not in the same format, but in a distinct format appropriate for discussing each approach.

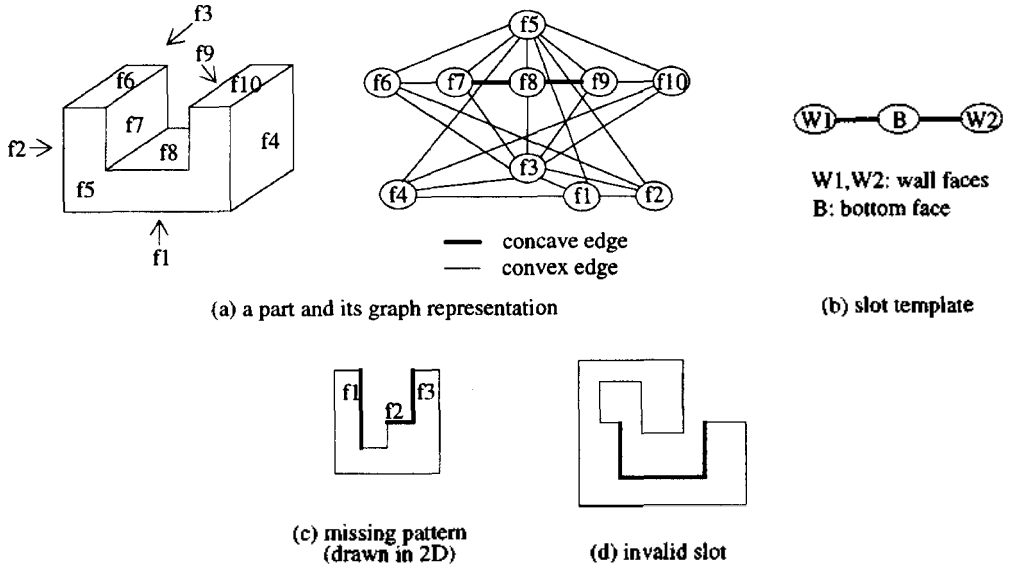
3. Graph-based Approaches

3.1 Description of technique

This approach was first formalized by Joshi and Chang[24]. Techniques based on the graph matching algorithms have been used in many subsequent research efforts and recently incorporated into commercial process planning software, such as Tecnomatix's PART[25,26]. In this approach, the BRep of the part is translated into a graph where, for example, its nodes represent faces and its arcs represent edges. An example is shown in Fig. 6(a). Additional information may be incorporated into the graph, e.g. edge-convexity, face-orientation, etc. Primitive features or feature templates are also represented by graphs (Fig. 6(b)). With this representation, the part graph is searched for the subgraphs that match the feature templates. In Fig. 6(a), (f7, f8, f9) will be matched with the slot template in Fig. 6(b).

3.2 Complexity analysis

Graph matching procedure is generally based on the *subgraph isomorphism technique*, which is a well-known NP-hard problem of exponential time complexity[27]. Graph pattern matching approaches have often been criticized for this computational shortcoming. In reality, however, this criticism may



(a) a part and its graph representation

(b) slot template

(c) missing pattern (drawn in 2D)

(d) invalid slot

Fig. 6. Graph pattern matching.

be unwarranted. In practical situations, the graphs representing feature templates are of limited size. In other words, the input size is so small that asymptotic worst-case complexity analysis is not appropriate. For example, the slot template in Fig. 6(b) has only three nodes and two arcs. The features defined in Trika and Kashyap's graph pattern matching algorithm have at most six face nodes[28].

Algorithms for computing subgraph isomorphism are of polynomial time complexity when the size of the graph to be matched is bounded by a constant. Suppose that G_1 is the part graph and G_2 is the feature template graph where the number of nodes in G_1 is n and the number of nodes in G_2 is k, where k is a constant. We can enumerate all of the subgraphs of G_1 with size k. Brute-force enumeration of the subgraphs requires ${}_nC_k$ operations. Its complexity is $O(n^k)$, which is polynomial because k is a constant. Now, we can apply the (sub) graph isomorphism algorithm between G_2 and a subgraph extracted from G_1. Its complexity is $O(2^k)$ $=O(1)$ because k is a constant. Therefore, the combined complexity is $O(n^k)$, which is polynomial. Note that this is a naive analysis for a brute-force algorithm. In implementing graph pattern matching for feature recognition, we can achieve much higher efficiency.

3.3 Feature intersections

A main problem with the graph pattern matching approach is that it makes it very difficult to recognize *intersecting* features. While quite successful in recognizing *isolated* features, this approach reveals many difficulties when the face patterns of the part are altered due to feature intersections. For example, from the part in Fig. 6(c), we can expect a slot with walls f1 and f3 and floor f2. However, the arc between f1 and f2 does not exist in the graph representation of the part, and therefore pattern matching will fail. The possible types of feature intersections that may arise in a complex part are unlimited. As we cannot enumerate all possibilities, naive pattern matching must be weak in recognizing intersecting features.

The ability to handle intersecting features has been an informal benchmark for feature recognition systems and therefore numerous research efforts have been made focusing on this issue. A novel solution to this problem was proposed by Marefat and Kashyap[29]. They observed that the arcs between (a feature's) face nodes in the part graph may be missing when features intersect. They proposed to *restore* the missing arcs into the part graph. They collected all possible candidates for missing arcs, and ranked the candidates based on part geometry information using the Dempster-Shafer theory[30]. The arcs with "notably different (higher)" ranks were restored. For example, in Fig. 6(c), the missing arc connecting f1 and f2 is restored, and therefore a slot with walls f1 and f3 and floor f2 can now be recognized. However, the problem with their method is that the *exact* set of missing arcs is not guaranteed to be identified. When we add fewer arcs than necessary, there are unrecognized features. When we add extraneous arcs, we may introduce bogus features. Marefat[31-33] proposed a similar approach based on Bayesian networks[34], but could not overcome the problem.

3.4 Issues in machining feature recognition

An additional problem with the graph pattern matching approaches is that it is difficult to ensure the machinability of recognized features. This is especially problematic when features are defined exclusively as surface features, i.e. collections of faces. As shown in Fig. 6(d), even though the three faces in bold match the face template of a slot, the recognized feature is not machinable as a slot because it is not accessible from the top. The non-volumetric notion of feature may cause fatal problems in machining applications. The graph pattern matching approach has also difficulties in handling variational feature instances of a class. For example, the pocket defined in Marefat and Kashyap's approach[29] has four wall faces and a bottom face. However, a pocket, in general, may have an arbitrary number of wall faces - making it difficult for graph pattern matching methods to recognize wide varieties of pocket instances.

Along the line of Marefat and Kashyap's work[29],

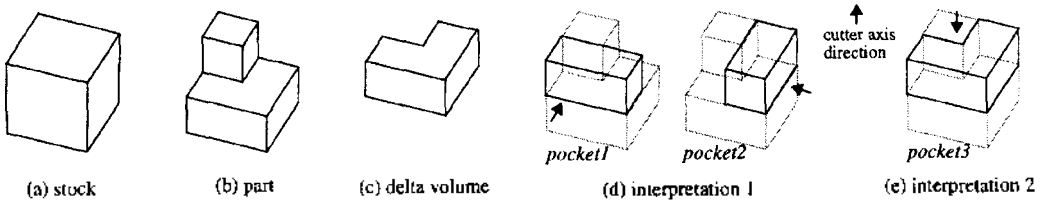


Fig. 7. Delta volume and multiple interpretations.

Trika and Kashyap[28] devised algorithms that can compute the *exact* set of missing arcs. However, their algorithms place very strong restrictions on input parts and feature intersections: the part must be polyhedral (only with planar faces) and iso-oriented (with no inclined faces). As a consequence, every recognized feature is cuboidal, generated by associating a volume with the recognized surface feature. This work is interesting from the viewpoint of pure pattern recognition. However, it does not consider manufacturing information that accounts for feature accessibility, selection of cutting tools, etc. Consequently, the recognized features are merely form features with cuboidal shapes, which might be good for shape analysis.

An important contribution of Trika and Kashyap[28] is related to the issue of *completeness*. The input for feature recognizers is typically a solid model for the desired part, plus a solid model of the stock (raw material). The material to be removed by machining, called the *delta volume*, is computed by subtracting the part from the stock, as depicted in Fig. 7. Trika and Kashyap called a feature recognizer complete if, for every part, the delta volume is contained in the union of all volumetric features generated by the feature recognizer. For example, if a feature recognizer generates two features shown in Fig. 7(d), it is complete. If a feature recognizer is not complete, there may exist unrecognized regions of the delta volume and therefore the specified part may not be obtained even after all feature removal operations are done. Trika and Kashyap[28] proved that their algorithm is complete.

4. Volumetric Decomposition Approaches

In the previous section, I explained the graph based algorithms and discussed several important issues in feature recognition. The most critical issue is how to recognize intersecting features. In this section, I discuss a different approach which is often called a volumetric decomposition and aims at handling intersecting features. Two major camps fall into this category: *cell-based decomposition* and *convex hull decomposition*. They decompose the input object into a set of intermediate volumes and then manipulate the volumes to produce features. In this section, I focus on the cell-based decomposition.

The cell decomposition approach for feature recognition was originally explored in 1983 by a research group from Allied Signal Aerospace (at that time Bendix) in Kansas City[35]. The objective of this approach, called the Phoenix Method, was to use volume decomposition to facilitate BRep to CSG conversion and to generate machining feature information[1]. However, they abandoned this approach due to a number of computational and representational limitations. Recently, however, several research groups have taken up the study of cell decomposition approaches once again. Sakurai [38,39] has been a leading advocate for the revival of this type of technique, and several other research groups have adopted similar methodologies[40-44].

4.1 Description of technique

The cell-based decomposition approach for feature recognition essentially consists of (i) delta

[1] Feature recognition can be considered as a conversion process from a BRep into a special CSG, called Destructive Solid Geometry (DSG)[36] where all Boolean operations at the internal nodes are subtractions only. General BRep-to-CSG conversion has been largely an open research issue despite some promising reports such as[37].

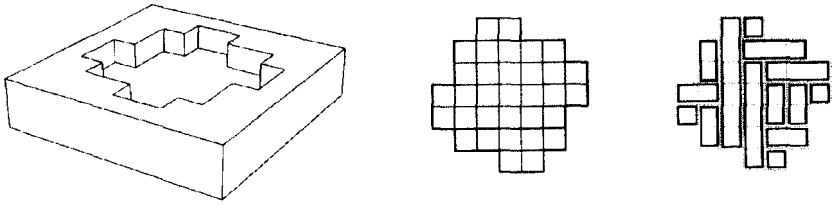


Fig. 8. Cell decomposition/composition.

volume decomposition into cells, (ii) cell composition and (iii) feature classification. In the first step, the delta volume is decomposed into minimal cells by extending and intersecting all the surfaces or halfspaces of the delta volume. If we are given the part shown in Fig. 8(a) and its stock corresponds to its convex hull, the delta volume is decomposed into the cells shown in (b). The union of all cells is equal to the delta volume, and the regularized intersection of any pair of cells is null. In the second step, a subset of the cells are combined (composed) to generate a volume to be removed by a machining operation, and in the last step the volume is classified as a machining feature. In the cell-based decomposition approach, the differences of the proposed algorithms mostly lie in the methods for combining cells into features.

4.2 Analysis of technique

I call the main problem of this approach *the global effect of local geometry*. A machining feature usually leaves its traces (faces) in a *localized* area of the part. However, the cell decomposition step extends *globally* the surfaces or halfspaces associated with the faces of the delta volume and quite often generates a huge number of cells as illustrated in Fig. 8. The difficulty in the cell-based decomposition approach is how to combine such cells and produce suitable features.

When we have n cells, all possible combinations constitute its power set. Sakurai and Chin[39] proposed to generate *all* possible features. Even though some heuristics are used to prune unpromising compositions, the composition algorithm cannot avoid exponential time complexity. This is a serious problem because the number n of initial cells often is large. Coles *et al.*[41] proposed to compose the cells into convex volumes only, but their approach is also subject to combinatorial explosion. Shah *et al.*[40] proposed a tractable composition algorithm which does not allow two features to share any cell. Starting from a cell, neighboring cells are combined one at a time such that the intermediate volume remains convex. When no more combination is possible, the volume is deleted from the set of cells. By selecting a new cell, the same procedure is repeated. This composition algorithm often leads to awkward decompositions of the delta volume, as shown in Fig. 8(c)[2]. The composition algorithm generates 14 features to decompose the delta volume which could be machined as one large pocket. The recognized machining features would not provide a basis for a practical and efficient machining strategy. Note that the example part is fairly straightforward: its cell decomposition results in a 2D array of regularly-shaped cells. In general cases where features intersect non-orthogonally, the resulting feature models may prove unwieldy.

Sakurai and Dave[44] made strenuous efforts to compose the cells into so-called "maximal volumes". The composition step starts from a cell and keeps adding cells adjacent to each other, but by contrast with the algorithm of Shah *et al.*[40], the intermediate and final volumes can be concave. Fig. 9 shows the set of maximal volumes produced

[2] The original algorithm proposed by Shah *et al.*[40] may leave large portions of the delta volume space unrecognized. Fig. 8(c) shows the result produced by the algorithm slightly modified by the author.

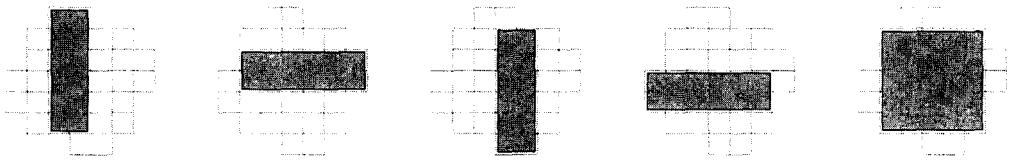


Fig. 9. Maximal volume composition.

from the part of Fig. 8(a). Each maximal volume is then classified into a machining feature through graph pattern matching. Sakurai and Dave[44] were successful to some extent in avoiding awkward machining feature models. However, the resulting maximal volumes may often be "unnecessarily complex and awkward in shape"[40]. They proposed an "enhancement" step for such awkward maximal volumes. However, it is not presented formally and no mathematical proof or justification from machining practice is given that the enhancement step always generates desirable maximal volumes.

One of the distinguishing characteristics of the cell-based decomposition approach is *multiple-step reasoning*: cell decomposition, cell composition and feature classification. Cell decomposition/composition generates a set of maximal volumes to which feature classification is applied *ex post facto*. The cell decomposition/composition is largely separated from the feature recognition/generation, and is not guided by the goal of recognizing specific types of features. In Sakurai and Dave's algorithm[44], a maximal volume may not match with any predefined feature type. They proposed a method to convert the "unrecognizable volumes" into features. Yet, no justification was given for it. Fig. 10(a), (b) and (c) show a stock, a part and the delta volume, respectively. A desirable decomposition might be the one shown in Fig. 10(d). In this example, the delta volume happens to be a cell and therefore a maximal volume. As Sakurai and Dave[44] did not discuss the feature classification step in detail, it is unclear whether the maximal volume becomes a recognizable volume or not. From a manufacturing viewpoint, it seems counter-intuitive to classify the maximal volume in Fig. 10(c) as a recognized volume (a single feature); and there seems to be no method for converting the single cell into the decomposition in Fig. 10(d). Sakurai and Dave[44] also provided basic rules to combine the maximal volumes into a more complex feature so that the five features in Fig. 9 can be combined into a single feature. However, success is not guaranteed, either.

The convex-hull decomposition approach[45-49] is also based on multiple-step reasoning and therefore shares the similar drawback discussed above. For detailed discussions, see[15].

5. Hint-based Approaches

Vandenbrande and Requicha[50,51] observed that searching for exact patterns of faces/edges/vertices is very likely to fail because such patterns are altered when features intersect. In response to these difficulties, they proposed hint-based reasoning to deal with intersecting features. Hint-based reasoning algorithms were designed and implemented first in OOFF (Object Oriented Feature Finder)[50] at USC, and recently in **F-Rex**[16] at the University of Maryland, **IF²** (Integrated Incremental Feature

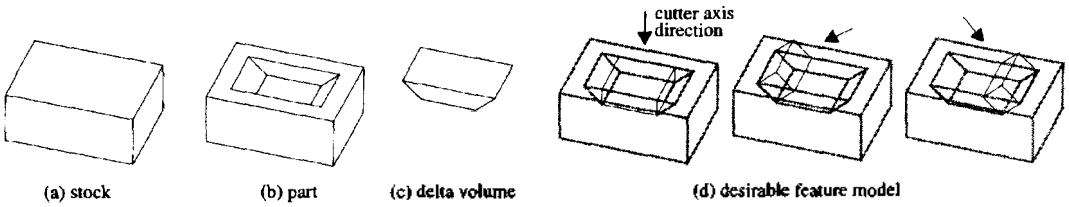


Fig. 10. Unclassified maximal volume.

Finder)[52] at USC and the FBMach System at Allied Signal Aerospace, Federal Systems Division[53]. This section discusses the hint-based reasoning algorithms with **IF²** example.

5.1 Description of hint-based techniques

Vandenbrande and Requicha[50] defined the so-called *presence rule*, which asserts, first of all, that a feature and its associated machining operation should leave a *trace* in the part boundary even when features intersect. Furthermore, the presence rule defines the *minimal indispensable* portion of a feature's boundary which should be present in the part. Consider a hole. Unless it is completely removed by other intersecting features, its machining operation leaves at least a face in the final part: the cylindrical wall face. This provides a *hint* for the potential existence of a hole.

Hints may comprise nominal geometries, design features, tolerances, and other design attributes associated with the CAD model. For example, a thread attribute may be taken as a hole hint. Most previous work focused on nominal-geometry hints, which are often simply called traces and can be identified on the boundary of the part. It should be noted, however, that hint-based algorithms can be extended to include hints based on other, non-geometric varieties of manufacturing information such as design features, tolerances and design attributes. See[15] for such an extension. (A trace may often mean a nominal-geometry hint only.) The basic components of a hint-based feature recognizer have been described by Regli[54,16]:

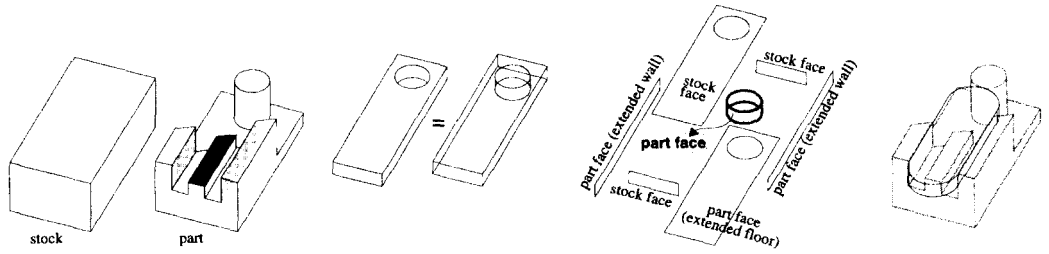
1. A set of feature types, M.
2. Each feature type M in M has associated with it a finite set of hint types $h_{M1}, h_{M2}, \ldots h_{Mk}$.
3. For each feature type, M, there is a *geometric completion* procedure $P_M( )$ which starts from the hint instances, performs extensive geometric reasoning and finally constructs feature instances of type M.

5.2 Description of IF²

IF² is a descendant of OOFF and can recognize holes, slots and pockets. Its recognition algorithm may be illustrated with a slot example. In **IF²**, a slot trace is defined to be the wall faces (side faces) of the slot. In other words, a slot trace is generated from nominal geometry when a pair of parallel opposing planar faces is encountered, which correspond to the slot walls. Given the part shown in Fig. 11(a), the lightly shaded faces constitute a slot trace.

The geometric completion procedures of **IF²** follow a *generate-test-repair* paradigm[55]. The *generate* step first finds the slot floor. Only the space between the wall faces is considered, and the part faces that are planar and perpendicular to the wall faces are taken as floor candidates. In Fig. 11(a), we can find several floor candidates and the heavily shaded face is an example. Then, the portion of the delta volume between the walls and above the floor, shown in Fig. 11(b), is proposed as a volume to be removed by a slot machining operation.

The *test* step checks the boundary of the proposed volume. The boundary is partitioned into 'stock faces' which originate from the stock and 'part faces' which originate from the part. 'Stock



(a) input stock and part (b) proposed removal volume (c) boundary analysis (d) recognized slot

Fig. 11. Slot recognition in **IF²**.

faces' are those to be 'removed' by feature machining operations, and 'part faces' are those to be 'created' by feature machining operations. For a slot, the proposed removal volume is not machinable as a whole if its boundary contains any 'part faces' besides the walls and floor. This is because such 'part faces' will be intruded, i.e. 'removed', by the parametrized slot feature volume which completely covers the proposed removal volume. Note that 'part faces' are those to be 'created'. The cylindrical face depicted in bold lines in Fig. 11(c) is such a 'part face'.

If the test step determines that the volume proposed by the *generate* step is not machinable as a whole, the *repair* step tries to instantiate a feature volume which is *maximally extended* but removes a subset of the proposed removal volume such that the machining operation *does not intrude* into the 'part face'. It is a kind of geometric fitting problem, and in the example **IF**^{2} finally produces a parametrized slot volume shown in Fig. 11(d).

5.3 Analysis of technique

A problem for hint-based approaches results from there being more traces than there are good features to recognize. A trace or hint is nothing but an implication for the possible existence of a feature, and therefore a significant number of traces may not lead to valid features. Even though the number of traces is bounded by polynomial (e.g. $O(n^2)$ slot traces where n is the number of planar faces of the delta volume), it is inefficient to perform expensive geometric reasoning on every trace. For example, (g1, g3) in Fig. 12 is a slot trace, but it does not evolve into a valid slot.

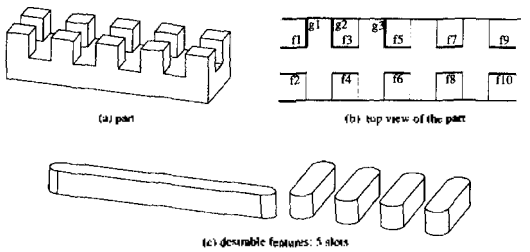


Fig. 12. An example of a part with many redundant traces.

On the other hand, several traces often lead to an identical volumetric feature. For example, in Fig. 12, five slot traces, (f1, f2), (f3, f4), (f5, f6), (f7, f8) and (f9, f10), will lead to the same slot, the long slot shown in (c).

IF^2 tackles this problem by assigning every trace a *heuristic strength*. The assigned value is a combined measure of (i) *preference* for such a feature (type) over alternative feature interpretations, and (ii) *belief* that the trace will lead to a valid machining feature. For example, in Fig. 12, (g1, g2) is ranked stronger than (g1, g3) based on a heuristic that narrow slots are more likely to occur than wide ones and the chances of having protrusions between the walls tend to be smaller when the gap is narrow.

The ranked traces are stored in a *priority queue* and serve as the system's *agenda*. **IF**^{2} selects the strongest trace from the priority queue and fires a geometric completion procedure on it. If geometric completion fails to construct a valid machining feature from the trace, the trace is discarded and the next highest-ranked trace is extracted. If completion succeeds, two tasks are done before selecting the next highest-ranked trace: (1) priority queue updating and (2) termination test.

The priority queue is updated to reflect the new feature's influence on other traces. For example, once a slot is recognized from (f1, f2), the strengths of (f3, f4), (f5, f6), (f7, f8) and (f9, f10) are reduced such that they attract less attention, as they would lead to redundant slots.

Initially, the material to be removed equals the delta volume. After updating the priority queue, **IF**^{2} updates the material to be removed by subtracting the new feature volume from it, and checks for a null solid. This is called *termination test*. If the result is null, the process stops because the delta volume is fully decomposed. Otherwise, **IF**^{2} takes the new top-ranked trace and repeats the same process.

IF^2 avoids unnecessary reasoning as much as possible by focusing on promising traces. **IF**^{2} also tries to produce a desirable interpretation (machining feature model) by focusing on preferred

traces. However, IF^2 does not always generate a desirable interpretation. Fig. 7 shows an example of *multiple interpretations*: the part can be machined in terms of two pockets, as shown in (d), or as a single complex pocket, as shown in (e). On the assumption that a 3-axis milling machine is used, the interpretation of a single pocket would be better because its cutter axis direction implies a single setup. In contrast, the interpretation of two pockets requires two setups. The current implementation of IF^2 generates the interpretation of a single pocket, but the heuristics are largely *ad hoc* and therefore success is not guaranteed for complex parts. (See[56] for examples and more detailed discussions.) However, IF^2 shows an effort for handling the problems of completeness and multiple interpretations. Section 6.2 discusses in detail both the issue of multiple interpretations and the problems of IF^2's method in tackling the issue.

6. Open Research Issues

In the previous sections, I surveyed three distinct approaches for feature recognition. This section formalizes the important issues discussed earlier and raises some new ones.

6.1 Recognition of intersecting features

The most critical issue in feature recognition has been said to be the capability of recognizing intersecting features. In the previous sections, I surveyed three approaches focusing on this issue. Those approaches have made progresses in dealing with intersecting features but also have their own deficiencies.

Among the current approaches, the hint-based approach seems to have demonstrated most promise. This is not surprising because a nominal-geometry hint is not an exact pattern of geometric entities and was introduced precisely to resolve the problem of recognizing intersecting features. However, the above survey shows that the problem of intersecting feature recognition is still the object of active research particularly because it is related with other problems discussed in the following.

6.2 Handling Multiple Interpretations

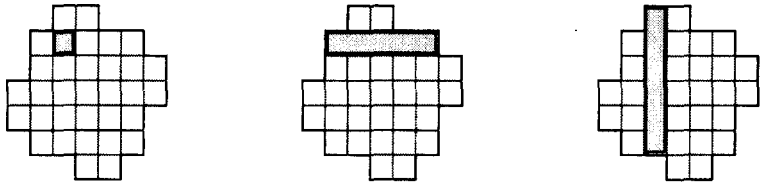
A part can be represented by more than one interpretation, as illustrated in Fig. 7. Multiple interpretations of a part roughly correspond to different ways to machine the part and therefore provide downstream applications with added flexibility. For example, design analysis or manufacturing planning activities typically attempt to optimize manufacturing cost or time, and therefore need to consider alternative interpretations.

Most previous feature recognition systems returned single interpretations. In a few cases, algorithms did produce alternative interpretations, usually on an *ad hoc* basis. Early work on handling multiple interpretations include *feature relaxation*[57], *feature algebras*[58], *feature aggregation/growing*[47]. More comprehensive works have been reported recently. Han[56] classifies them into two schools: one views an interpretation as a *machining sequence* and the other views an interpretation as a *feature cover*.

6.2.1 Machining sequence view

The machining sequence view focuses on partitioning a (subset of) delta volume into disjoint features. It has been gaining favor especially in the cell-based approach: Shah *et al.*[40], Tseng and Joshi[42], Sakurai and Dave[44], etc. Shah *et al.*[40] proposed an exhaustive method for generating *all* interpretations and sending them to a CAPP system for evaluation. Their composition algorithm discussed in Section 4 generates features and multiple interpretations simultaneously. Different feature models are generated depending on which cell is chosen as a starting point for cell composition and which direction is taken. Given a starting cell, we may have alternative moving directions determined by its neighboring cells. In the example of Fig. 13, we have two principal directions: ↔ and ↕. Once a direction is chosen, not only the concatenated volume (feature) but also (more importantly) a *family* of feature models is determined.

Suppose we start from the cell shown in Fig. 13(a). We can generate a feature along the ↔ direc-



(a) starting cell (b) a maximal convex volume (c) another maximal convex volume

Fig. 13. Multiple interpretations in cell decomposition/composition.

tion or ↕ direction, as shown in Fig. 13(b) and (c), respectively. The generated feature is deleted and the same procedure is repeated by selecting a new cell. The new cell will also have alternative moving directions. Each direction will determine a subfamily of feature models. There exists an exponential number of feature models for each (sub) family. A combinatorial explosion will rise as the number of cells becomes larger or the number of neighboring cells becomes larger.

In actuality, the cell composition algorithm of Shah *et al.* is a restricted simulation of machining operations. When a cutter starts material removal at a point, it usually has multiple choices of cutting directions. After an operation for machining a feature is executed along one direction, the cutter resumes material removal at an unremoved point - where it may also have multiple choices for the next direction. The nature of machining sequence generation is combinatorial.

6.2.2 Feature cover view

The feature cover view has been gaining favor, particularly in the context of the hint-based approaches: Vandenbrande[59], Gupta[60], Han[15], etc. In Gupta's dissertation[60], a feature model is viewed as a volumetric *set cover*[61] of the delta volume. As depicted in Fig. 14, we can view the delta volume Δ as a set of volumetric elements: $\Delta=\{e_1, e_2, e_3\}$. Each feature is a combination of adjacent volumetric elements and therefore is a subset of Δ, as shown in Fig. 14(d). We have a set of features, F={pocket1, pocket2, pocket3}. Then, a feature cover or an interpretation is a subset of F, whose members completely cover Δ. In the figure, we have two feature covers as shown in (e). To find an *optimal* interpretation is a set covering problem. In a pure set covering problem, *optimal* means a

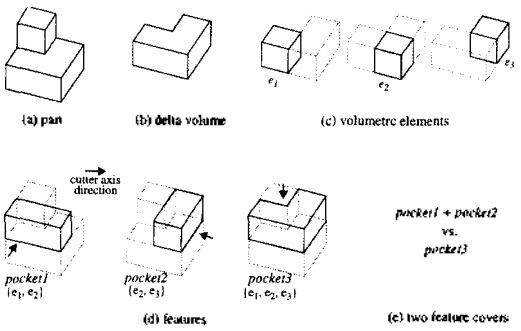


Fig. 14. Feature cover.

minimal number of subsets; in manufacturing applications, *optimal* may mean minimal cost, etc.

Gupta proposed to compute an *optimal* feature model starting from a *feature set*, which is an output of Regli's **F-Rex** feature recognizer and is formally defined in[16]. A feature set usually contains redundant features. In a feature set $F=\{f_1, f_2, \ldots, f_n\}$, f_i is a redundant feature if $F-\{f_i\}$ can completely decompose the delta volume. For each feature from the set F, an *effective removal volume* is computed by intersecting the feature with the stock. (If the feature occurs in a valid process plan, the effective removal volume represents the maximal volume that might be machined in the single operation represented by the feature.) Then, *every* set of effective removal volumes forming a *volumetric cover* of the delta volume is computed.

For each computed cover, feature models are generated by replacing the effective removal volumes by corresponding features. When a feature model is generated, possible machining plans are also generated and evaluated. Any feature model that is not expected to result in a plan better than existing ones is discarded by several pruning heuristics based on properties of the manufacturing

process being considered. Gupta used a set-covering algorithm in computing *all* volumetric covers of the delta volume. Set covering is a well-known NP-hard problem, and his algorithm for computing an optimal interpretation is subject to combinatorial explosion in the worst case.

6.2.3 Sub-optimal or nearly optimal interpretations

Whether or not one can compute the optimal interpretation depends on the part. For relatively simple parts with a small number of features, one can compute the optimal interpretation in a tractable time. (The complexities of a part are discussed in Section 6.5.) In contrast, for complex parts, it might not always be feasible to find the optimal interpretation, and the issue appears to be how to generate a set of nearly optimal or satisfactory interpretations in a tractable time.

As discussed in Section 4, the cell composition algorithm of Sakurai and Dave[44] generates so-called maximal volumes, which may intersect with each other. They view multiple interpretations as all possible linear orders of the maximal volumes that cover the delta volume. Obviously, the complexity of enumerating all sequences is $n!$ where n is the number of maximal volumes. They concluded that generating all the interpretations is unrealistic and proposed to generate *an* interpretation using some heuristics discussed in[62].

Han[15] considered an interpretation as a feature cover and proposed to produce a single *satisficing*[63] interpretation, as discussed in Section 5. In his system IF^2, promising/preferred traces are given higher strengths and tried first. Consequently most preferred features constitute the resulting interpretation. However, the heuristics are largely *ad hoc* in the sense that they are based only on partial/local information, and therefore some traces may be paid less attention than they should be. As a result, potentially useful interpretations could be eliminated from consideration. To tackle this problem, he developed a mechanism for generating alternative interpretations *on demand* from a human user. Ideally, these demands should come from the process planing system. The design and implementation of communication architecture between feature recognizers and CAPP would be a challenging issue, but he made little discussion on this.

6.3 Incorporation of manufacturing knowledge

Feature recognition is considered as a front-end of process planning, but there has been a wall between feature recognition and process planning. Much of the manufacturing knowledge such as manufacturing resources and tooling, typically used in process planning, is rarely incorporated into feature recognition. A feature recognizer simply outputs a set of features for a process planner, and there is little communication between them. As an example, consider hole recognition. A hole is typically recognized from a cylindrical cavity of an input part. However, if there is no drilling cutter with the same radius as the cylindrical cavity, the recognized feature might not be machined as a hole. Instead, a pocket interpretation would be required. This example shows that the features produced by many feature recognizers are not always machinable.

As shown in Fig. 2(c), a pocket's profile should not have convex vertices because these cannot be milled by physical cutters with nonzero diameters. However, Han[15] assumes that the milling cutter has an infinitesimal radius, and therefore a profile with a convex vertex is considered acceptable. The rationale for this assumption is two-fold. First of all, we cannot determine the machinability of a pocket until the available tool set is known. For example, the pocket shown in Fig. 2(c) will not be machinable if all available milling cutters' radii are greater than the radius of the pocket's cylindrical face (pocket corner). Secondly, since a pocket usually has associated tolerances, one may be able to machine a pocket with rounded corners whose radii are sufficiently small to satisfy the specified tolerances. Only when the available tool set is known and tolerances are examined can we determine a pocket's machinability.

These two examples show that feature recognition systems require information beyond geometry of the part to be manufactured. Roberts *et al.*[64] proposed a system which considered available factory resources. Regli[16] introduced algorithms for

accounting for tool assembly interference, tool shapes, and tool radii during feature recognition. In order to produce most process-specific features, feature recognition systems may have to interact with a *manufacturing resource agent*, which would have access to the available tool database, as described in[65]. Based on the information provided by the agent, for example, the feature recognizer would not recognize a cylindrical hole that cannot be drilled with the available tools.

6.4 Application to other manufacturing domains

To reason about CAD data across different manufacturing processes and throughout their lifecycle, we need different sets of feature concepts and feature definitions. As solid modeling technology has advanced, increasingly realistic and complex real world parts can be modeled. This advance in solid modeling has not produced a corresponding advance in feature recognition technology to handle the more realistic parts and more varied manufacturing domains.

Many feature recognition efforts are still focusing on domains of polyhedral parts. Several other manufacturing processes of critical importance include casting, forging, machining of free-formed surfaces, assembly[66-68], and layered manufacturing. The features for these domains and the algorithms for recognizing them may prove vastly different from those touched on above.

6.5 Scalability

The issue of speed has rarely been addressed in the context of feature recognition. As discussed earlier, some algorithms show exponential complexity. Even if many existing feature recognizers have polynomial time complexities, they often run minutes for simple parts with a few dozen surfaces. If the CAD input for feature recognition systems include more complex realistic parts typically used in industry or the systems consider more manufacturing issues such as tool accessibility, time can become a computational bottleneck. To alleviate the computational costs of geometric reasoning and exploit the growing power of networked and multiprocessor computers, Regli *et al.*[54] developed multiprocessor feature recognition algorithms. In most cases, however, the issue of speed and scalability has been ignored. More studies are required to bring computationally intensive geometric reasoning algorithms into everyday use.

7. Conclusions

In this paper, I have presented an overview of three distinct approaches in feature recognition for machining applications: graph-based algorithms, volumetric decomposition and hint-based reasoning. I have attempted to show where these approaches have had some success and where some of the issues for future work lie. In the machining domain, specific problems identified for which complete solutions have not yet been found include recognizing intersecting features, handling multiple interpretations within a single domain, controlling computational complexity, and associating manufacturing information with the recognized features.

There have been two important efforts worth mentioning. A special panel session for feature recognition[69] was organized at the 17th ASME International Computers in Engineering Conference, which was held in September of 1997. Four research groups participated in the panel session: (1) G. Little, R. Tuttle, D. Clark and J. Corney (UK Heriot-Watt University)[70], (2) E. Wang and Y. Kim (University of Illinois at Urbana-Champaign)[71], (3) J. Han (Sung Kyun Kwan University in Korea; formerly at USC and NIST), W. Regli (Drexel University) and S. Brooks (Allied Signal)[72], and (4) R. Sonthi and R. Gadh (University of Wisconsin)[73]. Each group reported test results with a collection of benchmark parts, which were submitted by the four participating groups. The panel session showed the current status of existing feature recognition systems, and there were discussions on the future research directions.

At the U.S. National Institute of Standards and Technology (NIST), there have been efforts for establishing a part repository for feature recognition, process and assembly planning. The part reposi-

tory is available through the World Wide Web at http://www.parts.nist.gov. (The parts for the special panel session mentioned above are also stored in the repository.) The part repository has helped research and development efforts obtain and share examples, focus on benchmarks, and identify areas of research need.

References

1. Hoffmann, C.M and Rossignac, J.R., A Road Map To Solid Modeling. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 2, No. 1, pp. 3-10, 1996.
2. Requicha, A.A.G., Representation for Rigid Solids: Theory, Methods, and Systems. *ACM Computing Surveys*, Vol. 12, No. 4, pp. 437-464, 1980.
3. Requicha, A.A.G. and Rossignac, J.R., Solid Modeling and Beyond. *IEEE Computer Graphics and Applications*, Vol. 12, No. 5, pp. 31-44, 1992.
4. Requicha, A.A.G. and Voelcker, H.B., Solid Modeling: A Historical Summary and Contemporary Assessment. *IEEE Computer Graphics and Applications*, Vol. 2, No. 2, pp. 9-24, 1982.
5. Requicha, A.A.G. and Voelcker, H.B., Solid Modeling: Current Status and Research Directions. *IEEE Computer Graphics and Applications*, Vol. 3, No. 7, pp. 25-37, 1983.
6. Sutherland, I.E., Sketchpad: A man-machine graphical communication system. In *Proc. of Spring Joint Computer Conference*, pp. 329-349, 1963.
7. Bedworth, D.D., Henderson, M.R. and Wolfe, P.M., *Computer-Integrated Design and Manufacturing*. McGraw-Hill, 1991.
8. Shah, J.J. and Mäntylä. *Parametric and Feature-Based CAD/CAM*. A Wiley-Interscience Publication, John Wiley & Sons, Inc., New York, 1995.
9. Lycourgos K. Kyprianou. *Shape Classification in Computer Aided Design*. PhD thesis, Christ College, University of Cambridge, Cambridge, United Kingdom, July, 1980.
10. Shah, J.J., Assessment of Features Technology. *Computer Aided Design*, Vol. 23, No. 5, pp. 331-343, 1991.
11. Subrahmanyam, S. and Wozny, M., An Overview of Automatic Feature Recognition Techniques for Computer-Aided Process Planning. *Computers in Industry*, Vol. 26, No. 1, pp. 1-21, 1995.
12. Allada, V. and Anand, S., Feature-based Modeling Approaches for Integrated Manufacturing: State-of-the-Art Survey and Future Research Directions. *Int. J. Computer Integrated Manufacturing*, Vol. 8, No. 6, pp. 411-440, 1995.
13. Silva, C.E., Alternative definitions of faces in boundary representations of solid objects. Technical Report 36, Production Automation Projection, University of Rochester, 1981.
14. Requicha, A.A.G. and Vandenbrande, J.H., Automated Systems for Process Planning and Part Programming. In AI: *Implications for CIM*, pp. 301-326. IFS Publications, Kempston, U.K and Springer-Verlag, New-York, 1988.
15. Han, J., *3D Geometric Reasoning Algorithms for Feature Recognition*. PhD thesis, Computer Science Department, University of Southern California (USC), 1996.
16. William C. Regli. *Geometric Algorithms for Recognition of Features from Solid Models*. PhD thesis, The University of Maryland, College Park, MD, 1995.
17. William C. Regli, Satyandra K. Gupta, and Dana S. Nau. Extracting alternative machining features: An algorithmic approach. *Research in Engineering Design*, Vol. 7, No. 3, pp. 173-192, 1995.
18. Tien-Chien Chang. *Expert Process Planning for Manufacturing*. Addison-Wesley Publishing Co., Reading, Massachusetts, 1990.
19. Cutkosky, M.R. and Jay M., Tenenbaum. Towards a framework for concurrent design. *International Journal of System Automation: Research and Applications*, Vol. 1, No. 3, pp. 239-261, 1992.
20. C. Smith and P. Wright. Cybercut: A world wide web based design-to-fabrication tool. *Journal of Manufacturing Systems*, Vol. 15, No. 6, pp. 432-442, 1996.
21. Han, J., Survey of Feature Research. Technical Report IRIS-96-346, Institute for Robotics and Intelligent Systems, University of Southern California, 1996.
22. Dixon, J.R., Libardi, E.C. and Nielsen, E.H., Unresolved Research Issues in Development of Design-with-Features Systems. *Geometric Modeling for Product Engineering, IFIP1990*, pp. 93-123, 1990.
23. Shah, J.J., Mäntylä, M. and Nau, D.S., Introduction to Feature Based Manufacturing. In Shah, J.J., Mäntylä, M. and Nau, D.S. editors, *Advanced in Feature Based Manufacturing*, pp. 1-11. Elsevier Science B.V., Amsterdam, The Netherlands, 1994.
24. Joshi, S. and Chang, T.C., Graph based Heuristics for Recognition of Machined Features from a 3-D Solid Model. *Computer Aided Design*, Vol. 20, pp. 58-66, 1988.
25. Reinhold Geelink, Otto W. Salomons, Fjodor van Slooten, and Fred J.A.M. van Houten. Unified fea-

ture definition for feature-based design and feature-based modeling. In Busnaina, A.A., editor, *ASME Computers in Engineering Conference*, pp. 517-534, New York, NY 100017, September 17-20, Boston, MA 1995.

26. van Houten, F.J.A.M., PART: *A Computer Aided Process Planning System*. PhD thesis, University of Twente, 1991.
27. Michael, R. Garey and David S., Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, W.H. and Company, New York, 1979.
28. Trika, S.N. and Kashyap, R.L., Geometric Reasoning for Extraction of Manufacturing Features in Iso-Oriented Polyhedrons. *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 16, No. 11, pp. 1087-1100, 1994.
29. Marefat, M. and Kashyap, R.L., Geometric Reasoning for Recognition of 3-D Object Features. *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 12, No. 10, pp. 949-965, 1990.
30. Shafer, G., *A Mathematical Theory of Evidence*. Princeton University Press, Prinston, NJ, 1976.
31. Marefat, M. and Ji, Q., Extraction and Identifying Form Features: A Bayesian Approach. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pp. 1959-1964, 1994.
32. Ji, Q., Marefat, M. and Lever, P.J., An Evidential Reasoning Approach for Recognizing Shape Features. In *Proc. 11th IEEE Conf. on AI for Automations*, pp. 162-169, 1995.
33. Ji, Q. and Marefat, M., Bayesian Approach for Extracting and Identifying Features. *Computer Aided Design*, Vol. 27, No. 6, pp. 435-454, 1995.
34. Pearl, J., *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann Publishers, 1988.
35. Noel C. Christensen, James D. Emory and Maurice L. Smith. Phoenix method for automatic conversion between geometric models. Allied Signal Incorporated, Kansas City, MO. US Patent 728367, 1983.
36. Arbab, F., *Requirements and Architecture of CAM oriented CAD systems for design and manufactures of mechanical parts*. PhD thesis, UCLA, 1982.
37. Shapiro, V. and Vossler, D., Construction and Optimization of CSG Representations, *Computer Aided Design*, Vol. 23, No. 1, pp. 8-20, 1991.
38. Sakurai, H. and Chin, C., Defining and Recognizing Cavity and Protrusion by Volumes. In *Proc. ASME Computers In Engineering Conference*, pp. 59-65, 1993.
39. Sakurai, H. and Chin, C., Definition and Recognition of Volume Features for Process Planning. In Shah, J.J. Mäntylä, M. and Nau, D.S., (Eds.), *Advances in Feature Based Manufacturing*, pp. 65-80. Elsevier Science B.V., Ameterdam, The Netherlands. 1994.
40. Shah, J.J. Shen, Y. and Shirur, A., Determination of machining volumes from extensible sets of design features. In Shah, J.J., Mäntylä, M. and Nau, D.S., (Eds.), *Advances in Feature Based Manufacturing*, pp. 129-157. Elsevier Science B.V., Ameterdam, The Netherlands. 1994.
41. Coles, J. Crawford, R. and Wood, K., Form Feature Recognition using Base Volume Decomposition. In *Proc. ASME Design Automation Conference*, pp. 281-297, 1994.
42. Tseng, Y.J. and Joshi, S.B., Recognizing Multiple Interpretations of Interaction Machining Features. *Computer Aided Design*. Vol. 26, No. 9, pp. 667-688, 1994.
43. Trika, S.N. and Kashyap, R.L., A Probably Correct Feature Extraction for Parts with Cylindrical and Planar Surfaces. In *Proc. Solid Modeling '95*, pp. 131-140, 1995.
44. Sakurai, H. and Dave, P., Volume Decomposition and Feature Recognition, Part II: Curved Objects. *Computer Aided Design*, Vol. 28, No. 6(7), pp. 519-537, 1996.
45. Kim, Y., *Convex Decomposition and Solid Geometric Modeling*. PhD thesis, Stanford University, 1990.
46. Kim, Y., Recognition of Form Features using Convex Decomposition. *Computer Aided Design*, Vol. 24, No. 9, pp. 461-476, 1992.
47. Waco, D. and Kim, Y.S., Geometric Reasoning for Machining Features using Convex Decomposition. In *2nd ACM Solid Modeling Symposium*, pp. 323-331, 1993.
48. Waco, D. and Kim, Y.S., Considerations in Positive to Negative Conversion for Machining Features using Convex Decompositions. In *Proc. ASME Computers In Engineering Conference*, pp. 35-45, 1993.
49. Waco, D. and Kim, Y.S., Geometric Reasoning for Machining Features using Convex Decomposition. *Computer Aided Design*, Vol. 26, No. 6, pp. 477-489, 1994.
50. Vandenbrande, J.H. and Requicha, A.A.G., Spatial Reasoning for the Automatic Recognition of Machinable Features in Solid Models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 15, No. 12, pp. 1-17, 1993.
51. Vandenbrande, J.H. and Requicha, A.A.G., Geometric Computation for the Recognition of Spatially Interacting Machinable Features. In Shah, J.J., Mäntylä, M. and Nau, D.S., (Eds.), *Advances in Feature Bas-*

ed Manufacturing, pp. 83-106. Elsevier Science B. V., Amsterdam, The Netherlands, 1994.

52. Han, J. and Requicha, A.A.G., Hint Generation and Completion for Feature Recognition. In *Proc. International Symposium on Automotive Technology and Automation (ISATA)*, pp. 89-96, 1996.
53. Steven L. Brooks and Bryan, R., Greenway Jr. Using STEP to integrate design features with manufacturing features. In Busnaina, A.A., (Ed.), *ASME Computers in Engineering Conference*, pp. 579-586, New York, NY 10017, September 17-20, Boston, MA 1995. ASME.
54. William C. Regli, Satyandra K. Gupta, and Dana S. Nau. Toward multiprocessor feature recognition. *Computer Aided Design*. Vol. 29, No. 1, pp. 37-51, 1997.
55. Simmons, R., A Theory of Debugging Plans and Interpretations. In *Proc. AAAI*, pp. 94-99, 1988.
56. Han, J., On Multiple Interpretations. In *4th ACM SIGGRAPH Symposium of Solid Modeling and Application*, pp. 311-321, 1997.
57. Mäntylä, M., Opas, J. and Puhakka, J., Generative Process Planning of Prismatic Parts by Feature Relaxation. In *ASME Advances in Design Automation*, pp. 49-60, 1989.
58. Karinthi, R.R. and Nau, D.S., An Algebraic Approach to Feature Intersections. *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 14, No. 4, pp. 469-484, 1992.
59. Vandenbrande, J.H., *Automatic Recognition of Machinable Features in Solid Models*. PhD thesis, Elec. Eng. Dept., Univ. of Rochester, 1990.
60. Gupta, S.K., *Automated Manufacturability Analysis of Machined Parts*. PhD thesis, University of Maryland, 1994.
61. Cormen, T.H., Leiserson, C.E. and Rivest, R.L., *Introduction to Algorithms*. McGraw-Hill, 1990.
62. Sakurai, H., Volume Decomposition and Feature Recognition, Part I: Polyhedral Objects. *Computer Aided Design*, Vol. 27, No. 11, pp. 833-843, 1995.
63. Simon, H.A., *The Science of the Artificial*. The MIT Press, 1969.
64. Roberts, C., Stage, R., Hubele, N., Henderson, M. and Perez, E., A New Approach to Manufacturing Features for Evaluation and Operational Planning. In *Proc. 5th IFIP Workshop*, WG 5.2, 1996.
65. Kevin K. Jurrens, James E. Fowler and Mary Elizabeth A. Algeo. Modeling of manufacturing resource information: Requirements specification. Technical Report NISTIR 5707, National Institute of Standards and Technology, Gaithersburg, MD, 20899, July 1995.
66. Latombe, J.-C. and Wilson, R., Assembly sequencing with toleranced parts. In *Third ACM/IEEE Symp. on Solid Modeling and Applications*, May 1995.
67. Wilson, R., *On Geometric Assembly Planning*. PhD thesis, Dept of Computer Science, Stanford University, March 1992.
68. Wilson, R. and Latombe, J.-C., Geometric reasoning about mechanical assembly. *Artificial Intelligent*, Vol. 71, No. 2, pp. 371-396, 1994.
69. Han, J., Regli, W.C. and Rosen, D., Special Panel Session for Feature Recognition. In *Proc. ASME Computers In Engineering Conference*, pp. 97-DETC:CIE-4423, 1997.
70. Little, G., Tuttle, R., Clark, D. and Corney, J., The Heriot-Watt FeatureFinder: A Graph-based Approach to Recognition. In *Proc. ASME Computers In Engineering Conference*, pp. 97-DETC:CIE-4425, 1997.
71. Wang, E. and Kim, Y., 1997 Status of the Form Feature Recognition Method using Convex Decomposition. In *Proc. ASME Computers In Engineering Conference*, pp. 97-DETC:CIE-4424, 1997.
72. Han, J., Regli, W.C. and Brooks, S., Hint-based Reasoning for Feature Recognition: Status Report. In *Proc. ASME Computers In Engineering Conference*, pp. 97-DETC:CIE-4485, 1997.
73. Sonthi, R. and Gadh, R., MMCs and PPCs and as Constructs of Curvature Regions for Form Feature Determinants. In *Proc. ASME Computers In Engineering Conference*, pp. 97-DETC:CIE-4426, 1997.

한 정 현

1988년 서울대학교 공과대학 컴퓨터공학과 학사
1991년 미국 신시내티대학교 전산학과 석사 (Computer Science, University of Cincinnati)
1996년 미국 USC 전산학과 박사 (Computer Science, University of Southern California)
1996년 ~ 1997년 미국 상무성 산하 표준기술연구원 (National Institute of Standards and Technology)
1997년 ~ 현재 성균관대학교 전기전자 및 컴퓨터공학부 전임강사
관심분야 : Solid-modeling, Computer Graphics, Computer Vision