Manufacturable Feature Recognition and its Integration with Process Planning

JungHyun Han Inho Han
School of Electrical and Computer Engineering
Sung Kyun Kwan University
Suwon, Korea 440-746
{han,ihhan}@ece.skku.ac.kr
http://graphics.skku.ac.kr

ABSTRACT

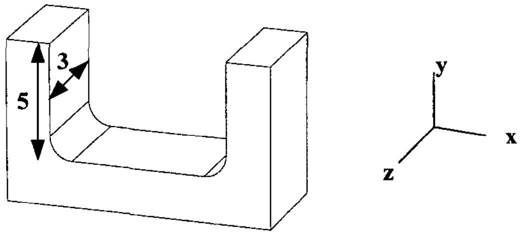
Despite the long history of research on feature recognition, its research results have rarely been transferred into industry. One of the reasons may be the separation of feature recognition and process planning. This paper proposes to integrate the two activities, and presents efforts towards it: feature recognition for manufacturability and setup minimization, feature dependency construction, and generation of an optimal feature-based machining sequence.

Keywords: feature recognition, process planning, manufacturability, feature dependency, machining sequence.

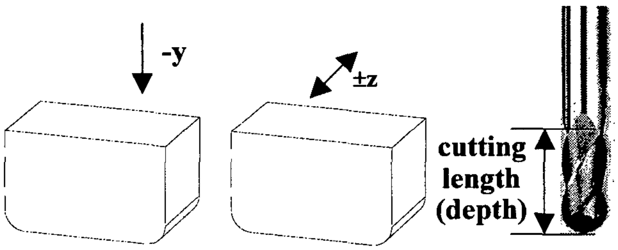
1. INTRODUCTION

Computer Aided Process Planning (CAPP) plays a key role by linking Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM). Given CAD data of a part, CAPP is to generate a sequenced set of instructions to manufacture the specified part. In order to do that, CAPP has to extract manufacturing *features* of the part. Therefore, feature recognition acts as a front-end of CAPP and has been the subject of research for two decades.

Fifth Symposium on Solid Modeling Ann Arbor MI



(a) part



(b) ***pocket*** $_1$ **(c)** ***pocket*** $_2$ **(d) ball end mill**

Figure 1: Manufacturability

An important issue was raised by [Han97a]: "Feature recognition has been considered as a front-end of process planning, but there has existed a *wall* between these two activities." Let us take the example part in Figure 1-(a), which is similar to the part discussed in [Gain98]. The part can be decomposed by either *pocket*$_1$ with the tool axis direction –y, as shown in (b), or *pocket*$_2$ with the tool axis direction ±z, as shown in (c). The arrow associated with each pocket indicates the tool axis vector for machining the pocket. From the manufacturing viewpoint, *pocket*$_1$ and *pocket*$_2$ should be taken as different features even though their shapes are geometrically equivalent. Suppose that the heights of *pocket*$_1$ and *pocket*$_2$ (along their tool axis directions) are 5 and 3, respectively. Suppose also that we have a single ball end mill with a cutting length (depth) 4, as shown in (d). Then, *pocket*$_2$ is manufacturable with the ball end mill whereas *pocket*$_1$ is not. (For simplicity of discussions, we assume that an end mill can intrude into the part material up to its cutting length.) Therefore, the part should be decomposed into $pocket_2$, not into $pocket_1$. However, most of existing feature recognition systems generate a set of features primarily based on geometric information of the part solid model, and do not care about its manufacturability.

This shows just an example of the high thick walls between feature recognition and process planning. This paper presents efforts to break the walls: features are recognized with manufacturability guaranteed, and simultaneously dependencies among features are analyzed. Finally, this paper shows how an optimal machining sequence can be generated by the aid of feature dependencies.

The literature on feature recognition is huge, but not much work has been reported on the issues this paper will address. An integrated system for feature recognition and process planning was developed at the University of Maryland[Gupt94, Regl95]. Feature recognition focused on manufacturability has been exploited at University of Illinois[Gain97, Gain98]. Their feature recognition system is made adaptive to resources (including tools) and does early process planning tasks. Dong and Vijayan's system recognizes features such that maximum amount of material can be removed in each setup in order to minimize manufacturing cost[Dong97a, Dong97b].

It is widely accepted that generic process planning research is saturated, but research based on feature-based technique is required to enhance the state-of-the-art[Maro95]. In this paper, the research goal is not to resolve the general process planning problems, but to develop a feature recognition system in accordance with the requirements of process planning.

2. GEOMETRIC REASONING FOR FEATURE RECOGNITION

This section briefly overviews the geometric reasoning kernel of our feature recognition system, which has already been published in literature[Han97b, Han98a]. It is crucial to overview our system's foundations in order to understand the research results presented at Sections 3 and 4.

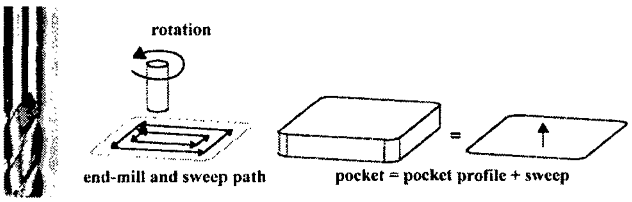
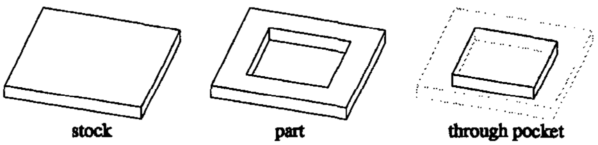
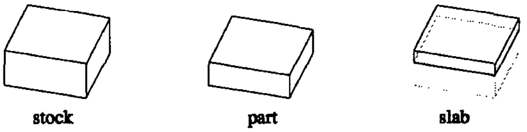


Figure 2: Pocket Definition



(a) through pocket



(b) slab

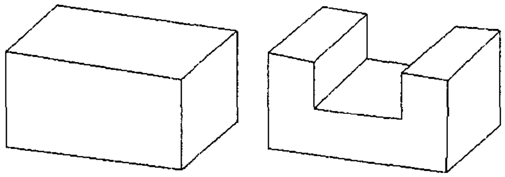
Figure 3: Through Pocket vs. Slab

2.1 Hint-based Reasoning

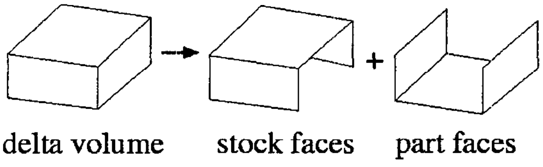
The first author of this paper designed and implemented Integrated Incremental Feature Finder (IF^2) at USC. Since then, IF^2 has been extended along several directions at NIST and Sung Kyun Kwan University. IF^2 recognizes holes, slots and pockets. This paper will focus on pockets because a slot is a specific instance of a pocket and a hole can also be taken as an instance of a pocket if we consider only milling machine centers. In this paper, we restrict discussions on flat end milling and ball end milling.

A pocket is machined by a series of cuts, as depicted in Figure 2. The pocket in Figure 2 is made by a flat end mill, and is represented by an arbitrarily-shaped planar *profile (floor)* and a sweeping vector. The sweeping vector is perpendicular to the profile and its length determines the pocket's height.

IF^2 is a *hint-based reasoning* system. Conceptually, a hint is a suggestion that a specific machining feature might exist in a part. Vandenbrande and Requicha[Vand93] 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 that must be present in the part. Consider a pocket. A pocket can leave, in the part, a floor set or a wall set. A floor is perpendicular to the tool axis direction. The wall set is composed of aligned faces, all of which are parallel to the tool axis direction. A recognizable pocket does not necessarily have both the floor set and the wall set. One of them can be missing. For example, a through pocket does not have a floor and a slab does not have a wall, as depicted in Figure 3-(a) and -(b), respectively. Therefore we have two types of hints for a pocket presence: floor and wall.

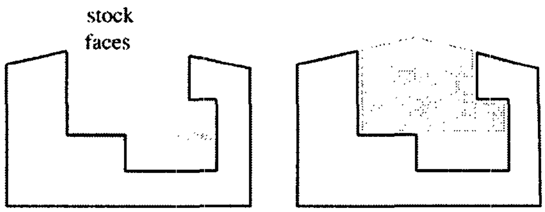


(a) stock (b) part

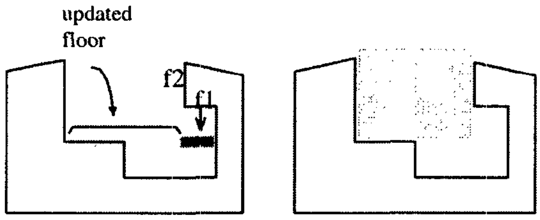


(c) partition of delta volume faces

Figure 4: Delta Volume

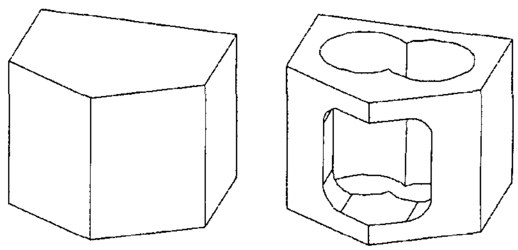


(a) extended floor (b) V*

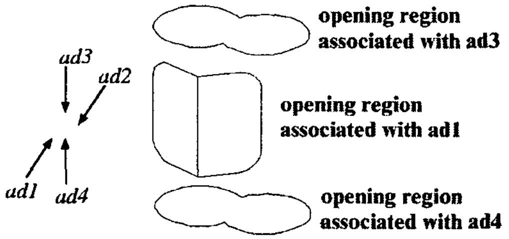


(c) floor update (d) pocket

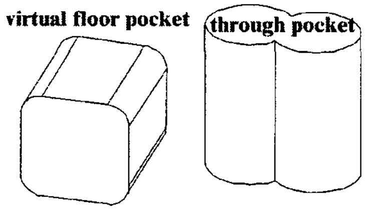
Figure 5: Floor-based Pocket Recognition Algorithm



(a) stock and part


opening region associated with ad3
ad3
ad2
opening region associated with ad1
ad1
ad4
opening region associated with ad4

(b) axis directions and associated opening regions



(c) floorless pockets

Figure 6: Wall-based Pocket Recognition Algorithm

2.2 Pocket Recognition: Floor-based Algorithm

The material to be removed by machining, called *delta volume*, is computed by subtracting the part from the stock. The faces of the delta volume are partitioned into 'part faces' to be *produced* by machining, and 'stock faces' to be *removed* (see Figure 4).

Given a floor hint, completion proceeds as shown in Figure 5-(a) through -(d). The plane of the floor is intersected with the delta volume, and a maximal, connected, non-intrusive extension of the floor is computed as shown in Figure 5-(a). The extended floor is swept along its normal vector beyond the delta volume, to produce a volume V. A pocket removal volume V* is proposed by intersecting V with the delta volume - see Figure 5-(b). V* is tested to see if its boundary has any 'part faces' besides the floor and walls. If V* has none, we instantiate a valid pocket from it. A local coordinate system is associated with it so that the floor normal coincides with the local Z axis. Computing an enclosing box for V* in its local frame provides us with the height of the pocket.

If additional 'part faces' are found in the boundary of V*, they are projected on the extended floor, as shown in Figure 5-(c), and subtracted from the floor. We then sweep the updated floor along the normal and intersect it with the delta volume. The height of the intersection is calculated as before, so as to instantiate a pocket, as shown in Figure 5-(d).

2.3 Pocket Recognition: Wall-based Algorithm

Not every pocket has a floor. Figure 6 shows two floorless pockets recognized by **IF^2**. We have to recognize a floorless pocket starting from its walls. When features intersect, however, it is very hard to collect the complete set of walls of a floorless pocket from the part boundary. Therefore we have to be able to reason from its subset.

Our procedures for recognizing floorless pockets begin by computing possible orientations for the axis of the mill. We look for cylindrical faces, or for pairs of non-parallel planar faces, which might be part of a floorless pocket's walls. For the cylindrical face's axis vector *A*, or for the cross product *A* of two non-parallel planar faces' normals, two hints are generated: one is *A* itself and the other is *-A* (opposite direction of *A*). For example, from the part in Figure 6, we obtain four floorless pocket hints: *ad1*, *ad2*, *ad3* and *ad4*. We will call them *axis hints* (whereas we call the hints in Section 2.2 *floor hints*).

Given an axis hint *A*, a face is said to be *visible* if it has at least a point whose normal vector on the face makes a negative dot product with *A*. Planar or cylindrical faces parallel to *A* are considered invisible. In a delta volume, a connected set of 'stock faces' which are visible along *A* constitutes an *opening region*. Figure 6-(b) shows the opening regions associated with three axis hints. The axis hint *ad2* does not have an opening region, and therefore is discarded. Once an opening region is found, it is swept along the axis hint direction beyond the delta volume, and intersected with the delta volume to generate V*. In order to analyze V*, a procedure similar to that discussed in Section 2.2 is performed. For the detailed discussion for the wall-based algorithms, readers are referred to [Han96, Han98a].

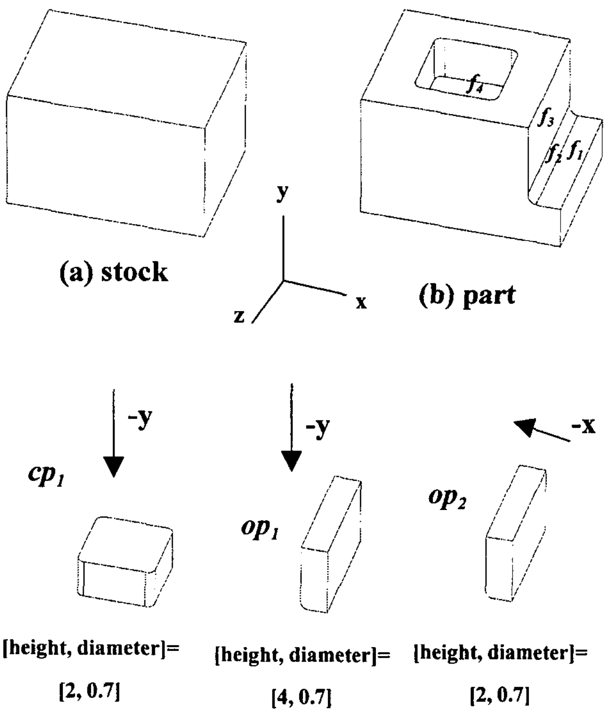
2.4 Control Mechanism

The set of features used to create a part is often called a *feature model* or an *interpretation* of the part. A part can be represented by more than one interpretation. Figure 1 shows a good example. Multiple interpretations of a part typically correspond to different ways to machine the part. Some research groups proposed to generate *all* possible interpretations or an *optimal* interpretation[Shah94, Tsen94, Gupta94]. Han showed that such an approach is subject to combinatorial explosion [Han97a], and proposed to generate *satisficing*[Simo69] interpretations.

IF² generates all hints at a time and *ranks* them by assigning a heuristic strength to each hint[Han97b]. The ranked hints constitute a priority queue and guide the focus of attention of the system.

Initially, the total volume to be removed is the entire delta volume. **IF²** processes the ranked hints in order of decreasing strength until the delta volume is completely decomposed. If a hint does not lead to a valid machining feature, it is deleted and the next highest-ranked hint is extracted from the priority queue. Otherwise, **IF²** updates the total volume to be removed by subtracting from it the new feature, and checks for a null solid. If the result is null, **IF²** stops because the delta volume is fully decomposed. Otherwise, **IF²** takes the new top-ranked hint and repeats the same process. We call the repeated process *recognize-test cycle*.

Note that **IF²** ranks hints based on heuristics and generates a single interpretation using the ranked hints. **IF²** tries to generate a satisficing interpretation. However, the interpretation may not be satisficing to users. **IF²** provides users with the capability of *alternative interpretations on demand*, presented in [Han97a].


f4
f3
f2
f1
y
x
z
-y
-y
-x
cp1
op1
op2
[height, diameter]= [2, 0.7]
[height, diameter]= [4, 0.7]
[height, diameter]= [2, 0.7]

(a) stock

(b) part

(c) features

ID	Type	Size (diameter)	Cutting Length (depth)
t_1	flat-end mill	0.5	4
t_2	flat-end mill	0.8	6
t_3	ball end mill	0.7	3.5

(d) tool database

Figure 7: Feature Recognition for Setup Minimization

3. MANUFACTURABLE FEATURE RECOGNITION

Section 2 overviewed **IF²**'s geometric reasoning kernel and control mechanism. Sections 3 and 4 will present recent research results, implemented in **IF²**, towards the integration with process planning.

3.1 Efforts for Setup Minimization

In process planning, a smaller number of setups is preferred because setup operations are costly and adversely affect precision. **IF²** tries to generate an interpretation which requires as small number of setups as possible. Recall that all pockets in **IF²** are associated with tool axis directions. In 3-axis machining, the number of setups is directly related with the tool axis directions.

We distinguish between closed pockets and open pockets. A closed pocket may or may not have a floor. (A good example of floorless closed pockets is a through pocket that crosses the entire part.) Note that, to a *closed* pocket that has a *floor*, an end mill has *a single* approachable direction, which is the opposite of the pocket's floor normal. In other words, the tool axis direction determined by such a pocket leads to an absolutely required setup in 3-axis machining[1]. For example, in Figure 7-(b), the closed pocket's floor f_4 determines a required setup –y. In contrast, a part with open pockets may be machined in multiple setups. For the part of Figure 1 which can be decomposed into either $pocket_1$ or $pocket_2$, we have three possible setups (-y and ±z).

The strategy of **IF**^{2} for minimizing the number of setups is as follows. When collecting hints, **IF**^{2} pays special attention to the floor hints for closed pockets and obtains the absolutely required setups determined by their floor normals. Floor hints for closed pockets can be easily found. If every edge of a floor is shared by a part face at a concave angle, the floor is a hint for a closed pocket. In the setup determined by the closed pocket hint, **IF**^{2} tries to recognize as many pockets as possible. When **IF**^{2} recognizes all possible pockets that can be machined at those absolutely required setups but the delta volume is not fully decomposed, the standard recognize-test cycle is repeated for unprocessed hints.

3.2 Feature Verification for Manufacturability

Given the part in Figure 7, **IF**^{2} discovers the closed pocket hint f_4 and recognizes all pockets that can be machined by mills approaching along –y direction. **IF**^{2} recognizes a closed pocket cp_1 from f_4, as shown in Figure 7-(c). For cp_1, [height, diameter] = [2, 0.7] indicates that cp_1's height along –y is 2 and its cylindrical corner between walls has a diameter 0.7.

At every iteration of the recognize-test cycle, a feature is tested if it is manufacturable with the available tool set. Manufacturability test is indispensable for generating a satisficing interpretation, discussed in Section 2.4. Suppose that **IF**^{2} is linked with a tool database such as the table shown in Figure 7-(d). For the closed pocket cp_1, **IF**^{2} finds the suitable tool t_1. (Another flat end mill t_2 is not suitable because its diameter is larger than that of the cylindrical face of cp_1. The ball end mill t_3 cannot be used because cp_1 needs flat end milling.) Therefore cp_1 is taken as a valid (manufacturable) feature, and the recognize-test cycle is repeated for the next hint f_1, which leads to the open pocket op_1. For op_1, [height, diameter] = [4, 0.7] indicates that its height is 4 along –y and the cylindrical blend between the floor and wall has a diameter 0.7. However, **IF**^{2} cannot find any suitable tool set for op_1. The available ball end mill t_3's cutting length (depth) is shorter than the height of op_1. Therefore, op_1 is rejected as an invalid feature.

All hints have been processed along –y direction, but the delta volume is not fully decomposed. Therefore the standard recognize-test cycle is repeated. The floor hint f_3 will then lead to the open pocket op_2 shown in Figure 7-(c). Even though op_1 and op_2 have an identical shape, they are different pockets with different tool axis directions. The ball end mill t_3's dimensional parameters are appropriate for machining op_2, and therefore op_2 is accepted as a valid feature. The delta volume is now completely decomposed by the interpretation $\{cp_1, op_2\}$, and the recognize-test cycle is terminated.

3.3 Feature Dependency

Starting from a hint, **IF**^{2} recognizes a *maximally extended* feature volume that is compatible with the hint. For example, given the stock and part shown in Figure 8-(a) and (b), respectively, the pocket hint f_D leads to the maximally extended pocket D shown in (c). Notice that D is extended vertically up to the stock face. In feature recognition research, maximal extension method has been widely accepted due to flexibility provided for process planning[Vand93, Saku96, Regl95].

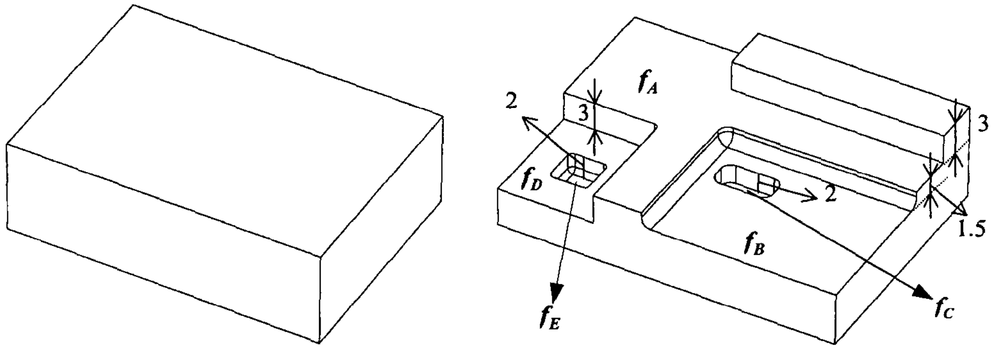
In Figure 8, the open pocket D's height is 6 and its cylindrical corner's diameter is 1, as denoted by [6, 1]. D should be machined by a flat end mill. However, as shown in the tool database of (d), the flat end mill with diameter 1 has a cutting length (depth) 3. Then, D would have to be rejected because it is not manufacturable with the available tool set. However, D turns out to be manufacturable if A is machined prior to it. This relation leads to *feature dependency*.

In **IF**^{2}, feature dependency construction is integrated with manufacturability test. In order to determine a feature's manufacturability, **IF**^{2} computes the portions of its wall faces which contact the part. In Figure 8-(e), such part-contacting portions of D are illustrated in a dark color[2]. As discussed in Section 2.2, a local coordinate system is introduced so that a pocket's floor normal coincides with the local Z axis. Computing an enclosing box for the part-contacting portions along the local Z axis leads to the pocket's so-called *required volume* range. It is called 'required' because it is *required* to be machined even after A is machined prior to D. The other range is often called *optional.*

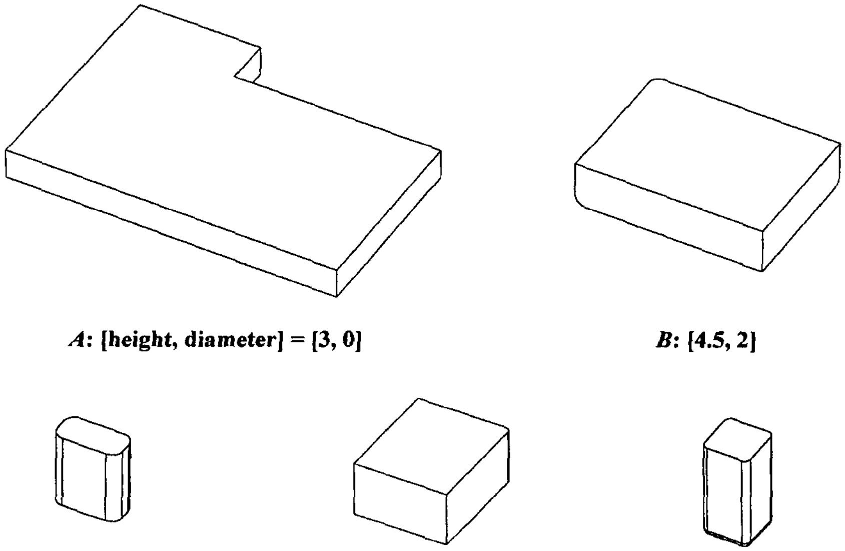
---

[1] Even though we consider only pockets in this paper, holes act a similar role for determining required setups. For a hole with an axis vector A, either A or $-A$ (opposite direction of A) leads to the required setup.

[2] **IF**^{2} is implemented using the Boundary Representation (BRep) modeler Parasolid, a commercial system marketed by EDS/Unigraphics. Extraction of part-contacting portions is easily achieved through Parasolid's Boolean operation and *attribute* facilities.



(a) stock **(b) part**



A: [height, diameter] = [3, 0]

B: [4.5, 2]

C: [6.5, 2]

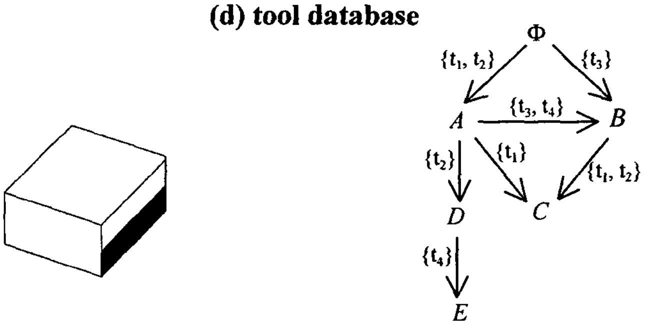
D: [6, 1]

E: [8, 1]

(c) features

ID	Type	Size(diameter)	Cutting Length (depth)
t_1	flat end mill	2	5
t_2	flat end mill	1	3
t_3	ball end mill	2	5
t_4	ball end mill	1	3

(d) tool database



(e) part-contacting portions of *D* **(f) feature dependencies**

Figure 8: Pocket Recognition Example

By checking the blends among boundary faces of a pocket ***P***, we can determine whether ***P*** needs flat end milling or ball end milling. (For *D*, a flat end milling is needed.) For selecting appropriate milling processes, $\mathbf{IF}^2$ collects every milling tool with a diameter smaller than or equal to that of ***P***'s blend. Those tools are candidates that could be used to remove ***P***. (For *D*, t_2 is chosen.) For each tool ***T***, collect every pocket ***Q*** whose floor position is between the top of ***P***'s required volume range and the top of ***T***'s cutting length (depth). (For *D* and t_2, *A* is collected.) Check if the profile (floor) of ***Q*** overlaps ***P***'s profile when they are projected along the tool axis direction. (It is the case for *D* and *A*) If so, $\mathbf{IF}^2$ decides that ***P*** depends on ***Q*** with respect to ***T***. In other words, ***Q*** should be machined prior to ***P*** if ***T*** is used for machining ***P***. (The pocket *A* should be machined prior to *D* if t_2 is used for machining *D*.) We denote such dependency by $Q \rightarrow P$, and assign ***T*** on the arrow. (An arrow is drawn from *A* to *D*, and assigned $\{t_2\}$.) If we cannot find such ***Q***, ***P*** cannot be machined with ***T***. If we cannot find such ***Q*** for every candidate tool (and there exists no tool that can directly machine the entire volume of ***P***), we decide ***P*** is not manufacturable at all.

All pairs of such feature dependencies result in a partially ordered graph, as shown in Figure 8-(f). Let us take a more complex example in Figure 8. For machining *C*, t_1 and t_2 are collected as candidate tools. For t_1, both *A* and *B* satisfy the above requirements, and so $A \rightarrow C$ and $B \rightarrow C$ hold with respect to t_1. For t_2, *B* satisfies the above requirements, and so only $B \rightarrow C$ holds with respect to t_2. These can be combined into (1) $A \rightarrow C$ with respect to $\{t_1\}$ and (2) $B \rightarrow C$ with respect to $\{t_1, t_2\}$. These dependencies imply that *C* can be machined with t_1 if *A* is removed first, ***or*** *C* can be machined with either t_1 or t_2 if *B* is removed first. It is important to understand that both of *A* and *B* are not pre-requisites for machining *C*. Instead, either *A* or *B* is a pre-requisite. In other words, *C* becomes manufacturable if either *A* or *B* is removed first.

In the graph, Φ represents *no pre-requisite* and therefore *A* and *B* are taken as manufacturable with no dependency on other features. Even for such features, however, $\mathbf{IF}^2$ constructs feature dependency. For example, *B* can be immediately taken as manufacturable due to t_3. Therefore, in the graph, $\Phi \rightarrow B$ appears with respect to $\{t_3\}$. However, if we follow the dependency construction procedure, we can add in the graph $A \rightarrow B$ with respect to $\{t_3, t_4\}$. These dependencies imply that *B* can be machined with t_3 without considering any other features, ***or*** *B* can be machined with either t_3 or t_4 if *A* is removed first. As demonstrated in this example, dependency construction for an already-manufacturable feature allows us to use one more tool (t_4) and therefore helps us achieve optimization for process planning.

4. MACHINING SEQUENCE GENERATION

Let us show how the feature dependency network can be used for machining sequence generation. In Figure 8, $\mathbf{IF}^2$ recognizes five manufacturable pockets, *A*, *B*, *C*, *D* and *E*, and constructs the feature dependency graph. Let us generate a machining sequence based on the partial ordering described in the graph. We could adopt a simple *topological sorting* algorithm [Corm90] to get a total ordering such as $A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$. However, this sequence of machining may not be optimal. For example, this sequence requires four tool changes. (The installation of the first tool is also counted as a tool change.) In contrast, some other sequences require only three tool changes.

In process planning, pursuit of optimality requires a number of considerations such as machining cost, tool change cost, setup cost, part reorientation cost, etc. In our work, all recognized features are associated with specific setups, and we pursue an optimal machining sequence in each setup. We may then measure optimality by the sum of machining cost and tool change cost, and try to minimize it.

Given the part material, the feature type/dimensions and the tool material/dimensions, it is possible to select appropriate cutting parameters such as feed. Once the cutting parameters are determined, we can compute the machining cost using well-developed formulas such as that in [Khos94]. Machining cost estimation is now a regular part in process planning. At the time of writing this paper, however, the database (lookup table) corresponding to the machining data handbook is under construction, and therefore we focus on tool change cost in this paper. Note that, however, machining cost can be immediately incorporated in the scheme described below.

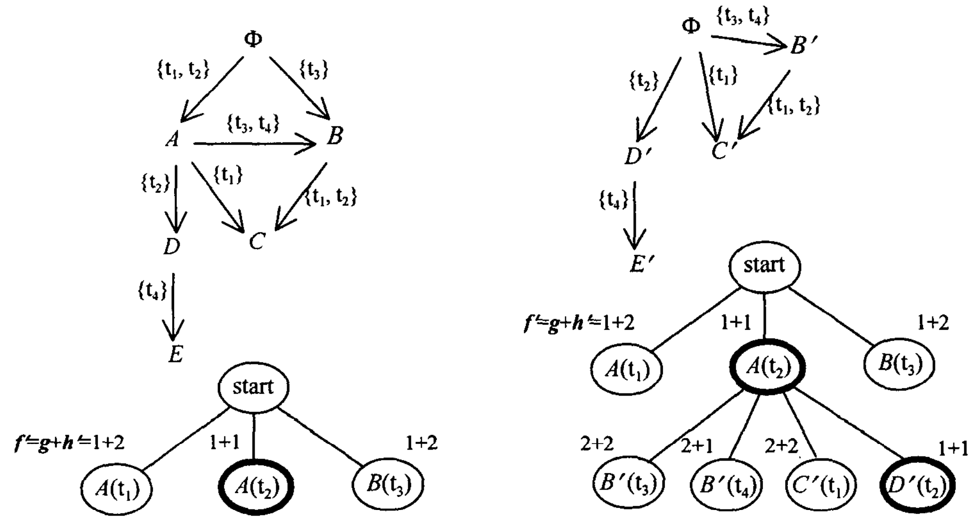
If we pursue an *optimal* machining sequence, the problem becomes a search problem. The optimal path to a goal state in a search space can be found by A* algorithm which was first presented by [Hart et al., 1968]. In A* algorithm, we need a heuristic function f' that evaluates each state we generate. The prime on f indicates that it is an approximation to a function f that gives the true evaluation of the state. The function f' is defined as the sum of g and h' where g is a measure of the cost of getting from the start state to the current state and h' is an evaluation of the additional cost of getting from the current state to a goal state. In other words, f' is an evaluation of the cost of getting from the start state to a goal state along the path that generated the current path. In our application, g is a measure of "how many tool changes have occurred," and h' is a guess of "how many tool changes will occur." We reach a goal state when all features are manufactured (with the minimum cost).

Adopting a search algorithm for process planning is not a new idea. For example, Gupta used branch-and-bound algorithm[Gupt94], and Sormaz used A* algorithm for optimal process planning[Sorm94]. However, their approaches are based on unrefined feature precedence relations, which could prevent an optimal plan from being generated.

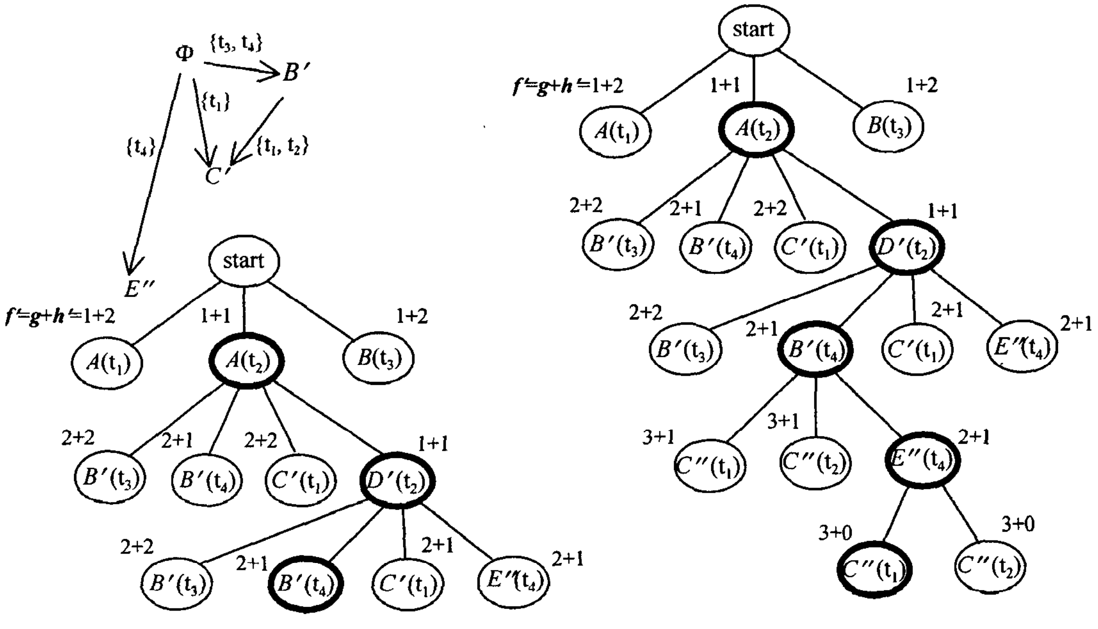
Figure 9 shows the search trees spanned until a goal is found. Initially, there is only one state: the start state. According to the feature dependency graph shown in Figure 8, we can start machining either *A* or *B*, which is pointed by Φ

	t1	t2	t3	t4
A	✓	✓		
B			✓	✓
C	✓	✓		
D		✓		
E				✓

(a) feature-tool table

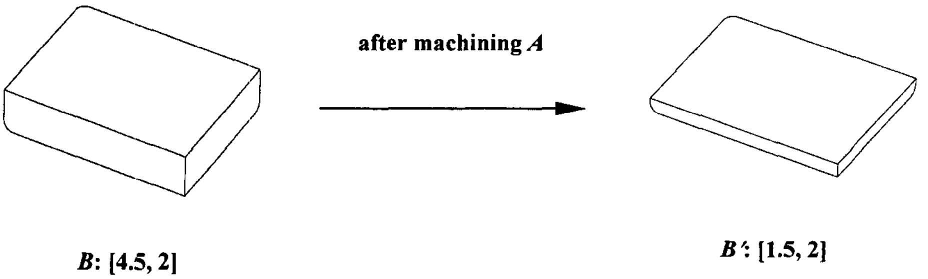


(b) level 2 **(c) level 3**

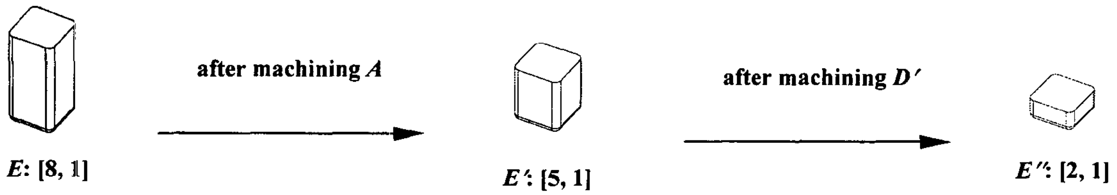


(d) level4 **(e) when a goal is found**

Figure 9: Application of A* Algorithm



(a) update of feature B



(b) update of feature E

Figure 10: Feature Volume Update

and called a *maximal* element. According to the links from Φ, we can see that A can be machined by either t_1 or t_2 whereas B can be only by t_3. Therefore, as shown in Figure 9-(b), the start state has three branch states: $A(t_1)$, $A(t_2)$ and $B(t_3)$, where, for example, $A(t_1)$ represents machining A with t_1. For each state, we compute f'. The g component of f' simply counts how many tools have been changed. For the state $A(t_1)$, g is 1 because the first tool installation is counted as a tool change. For every state at the second level of the search tree, g is 1.

For computing h', we repeatedly use a *greedy heuristic* [Corm90]. From the feature dependency graph, we can create the table in Figure 9-(a), where all possible tools are listed for each feature. This table is helpful for applying the greedy heuristic. The state $A(t_1)$ says that t_1 is selected for machining A. Our greedy heuristic proposes that, as t_1 is already selected, all remaining features that can be machined by t_1 should be machined by it. The table shows that C can be machined by t_1. Then, A and C are assumed to be machined out, and B, D and E remain. Computing h' is to guess how many tool changes will be needed to manufacture these remaining features B, D and E. Let us again take the greedy strategy. Among the tools that can machine them, choose a tool *with most occurrences*. It is t_4 that can machine B and E. Then, only D remains and it can be machined with t_2. Our greedy heuristic sets h' to 2: i.e. from t_1 to t_4, and then to t_2. Therefore, f' is set to 3, which is sum of g and h'.

Let us compute f' for the state $A(t_2)$, which says t_2 is selected for machining A. Assuming that all remaining features that can be machined by t_2 will also be machined by it, we decide that we can machine out C and D with t_2. Then, the remaining features will be B and E, both of which can be machined by t_4. Therefore, h' is set to 1, and f' to 2.

For the right-most child $B(t_3)$ of the start state, f' is computed to be 3. Therefore, among the three children, $A(t_2)$ looks most promising and is chosen to be expanded at the next stage. Because A is machined out, the feature dependency graph is changed as shown in Figure 9-(c). Now the maximal elements are B', C' and D'. The prime on each feature indicates the updated feature volume resulting from machining A prior to the feature. For example, B is reduced to B' with height 1.5, as depicted in Figure 10-(a). Machining cost for the updated feature volume B' is usually cheaper than that for B, and the cost for B' should be used when we compute f' at the next stage (even though we do not include the machining cost for the current implementation). Without dynamically updating the feature volume and evaluating the machining cost for it, optimality cannot be achieved.

B' can be machined by either t_3 or t_4, C' only by t_1, and D' only by t_2. Therefore, as depicted in Figure 9-(c), state $A(t_2)$ has four branch states: $B'(t_3)$, $B'(t_4)$, $C'(t_1)$ and $D'(t_2)$. Let us traverse the search space. In Figure 9-(c), four states $B'(t_3)$, $B'(t_4)$, $C'(t_1)$ and $D'(t_2)$ are assigned f' values 4, 3, 4 and 2, respectively. Among all terminal nodes in the search tree, $D'(t_2)$ has the smallest f' value, and so is selected to be expanded at the next stage. When D' is machined out, B', C' and E'' become new maximal elements as shown in Figure 9-(d), and therefore $D'(t_2)$ has four children $B'(t_3)$, $B'(t_4)$, $C'(t_1)$ and $E''(t_4)$. E'' denotes the reduced feature volume resulting from machining A and D', as depicted in Figure 10-(b). Figure 9-(d) shows their f' values. $B'(t_4)$, $C'(t_1)$ and $E''(t_4)$ are assigned 3. Note that their f'

values are the same as those of $A(t_1)$ and $B(t_3)$ at the second level, and that of $B'(t_4)$ at the third level. We resolve this in favor of the path we are currently following such that we can avoid *backtracking* if possible[3]. At a level, let us choose the left-most state. Therefore, we select $B'(t_4)$. If we keep searching this way, we will end up with the expanded tree shown in Figure 9-(e). We find an optimal path $A \rightarrow D' \rightarrow B' \rightarrow E'' \rightarrow C''$ which requires only three tool changes: $t_2 \rightarrow t_4 \rightarrow t_1$.

If we can guarantee that $\boldsymbol{h'}$ never *overestimates* $\boldsymbol{h}$, the A* algorithm is guaranteed to find an optimal path to a goal, if one exists [Nils80]. It is very important to note that we do not take the feature dependencies into account when we compute $\boldsymbol{h'}$. Instead, we simply use a greedy heuristic. Therefore, $\boldsymbol{h}$ should be greater than or equal to the value of $\boldsymbol{h'}$, i.e. $\boldsymbol{h'}$ can never be an *overestimate*. Consequently, application of A* algorithm in the machining sequence generation always generates an optimal solution, i.e. the minimum number of tool changes.

5. DISCUSSION

So far, feature recognition research has largely focused on finding all possible features, and the task of manufacturability analysis is shifted to process planners. Especially when a part has multiple interpretations, this often causes serious problems: an inefficient plan may be generated or it may be impossible to generate a feasible plan. Our system interacts with the tool database and does manufacturability analysis together with feature dependency construction. These are the main contributions of this paper. In many works on feature recognition, feature dependency relations are defined among *nested features*, and then the machining sequences are determined based on simple rules such as *outer-feature-first*. Such methods do not necessarily lead to efficient/optimal plans. The feature dependency presented in this paper guides the state space search for an optimal machining sequence.

Vandenbrande defined a feature's optional volume to be the portion of the feature that is shared with other features and therefore may be removed as a side effect of machining the other features[Vand93]. For example, if two slots cross each other, both of them have common optional volumes in the middle. The main reason for partitioning a feature into required and optional volumes was to avoid unnecessary process on the optional volumes. However, it has not been clearly demonstrated how the process planner can utilize the information on optional volumes common to multiple features. In contrast, we define the required/optional volumes along the tool axis direction, and it is for determining feature dependency. Two crossing slots do not have any optional volumes. Unnecessary process on the void region can be avoided by the aid of the feature volume update procedure, discussed in Section 4.

We may often be unable to generate an interpretation when some features are not manufacturable with the available tool set. For example, $pocket_2$ shown in Figure 1 cannot be manufactured if all available mills' radii are greater than the radius of the pocket's cylindrical face (pocket corner). Suppose that, however, the tolerances of $pocket_2$ allow a milling operation with a mill whose radius is larger than that of the pocket corner. Then, $pocket_2$ turns out to be manufacturable. In actuality, manufacturability of a feature can be determined not only when the available tool set is known but also tolerances associated with the feature are examined. Note that tolerance can also affect feature dependency. **IF^2** is currently being extended so as to be able to do tolerance analysis.

6. IMPLEMENTATION

The proof-of-concept implementation of the algorithms discussed in this paper was done. Originally **IF^2** was written in a combination of C++ and LISP, and operated in Unix. However, we changed its base platform into Windows NT on PC, and rewrote the LISP portion of **IF^2** in C++, primarily for speedup. In order to guarantee features' manufacturability, we also added to **IF^2** the capability of cooperating with the tool database. We use Microsoft ODBC APIs which are functions to access various DBMSs. **IF^2** obtains geometric services from the Parasolid modeler. The interface between **IF^2** and Parasolid is discussed in [Han98b].

7. CONCLUSION

Features play a key role in achieving the goal of CAD/CAM integration. However, such a goal still seems remote despite two-decades of research on feature recognition. One of the reasons is that feature recognition is not guided by the requirements of downstream applications such as process planning. Much of the manufacturing knowledge, which is typically used in process planning, is rarely incorporated into feature recognition. After the output of a feature recognizer is fed into a process planner, there is little communication between these two activities. This paper proposes to integrate the two activities, and presents efforts towards it: feature recognition for manufacturability and setup minimization, feature dependency construction, generation of an optimal feature-based machining sequence, etc.

However, more considerations are required for generating manufacturable/satisficing interpretations. They include tolerance analysis, fixturing analysis, etc. However, research

---

[3] If all of four states at the fourth level $B'(t_3)$, $B'(t_4)$, $C'(t_1)$ and $E''(t_4)$ had 4 as their f' values, we would have to backtrack to $B'(t_4)$ at the third level.

results on these issues have rarely been reported in the open literature. We believe that the research works reported in this paper provide a framework for the utilization of such information.

ACKNOWLEDGEMENT

The work presented in this paper has been supported by the University Basic Research Program of the Ministry of Information and Communication, Korea.

REFERENCES

[Corm90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, McGraw-Hill, 1990.

[Dong97a] J. Dong and S. Vijayan, "Manufacturing Feature Determination and Extraction - Part1: Optimal Volume Segmentation," Computer Aided Design, June 1997, Vol. 29, No. 6, pp. 427-440.

[Dong97b] J. Dong and S. Vijayan, "Manufacturing Feature Determination and Extraction - Part2: A Heuristic Approach," Computer Aided Design, July 1997, Vol. 29, No. 7, pp. 475-484.

[Gain97] D. M. Gains and C. C. Hayes, "A Constraint-based Algorithm for Reasoning about the Shape Producing Capabilities of Cutting Tools in Machined Parts," *Proc. DFM97*, DETC97/DFM-4322, 1997.

[Gain98] D. M. Gains, F. Castano and C. C. Hayes, "MEDIATOR: A Resource Adaptive Feature Recognizer that Interwines Feature Extraction and Manufacturing Analysis," Accepted in *ASME Journal of Mechanical Design.*

[Gupt94] S. K. Gupta. *Automated Manufacturability Analysis of Machined Parts.* PhD thesis, University of Maryland, 1994.

[Han96] JungHyun Han, *3D Geometric Reasoning Algorithms for Feature Recognition*, Ph.D. Dissertation, Computer Science Department, USC, 1996.

[Han97a] JungHyun Han, "On Multiple Interpretations," *4th ACM SIGGRAPH Symposium on Solid Modeling and Applications*, Atlanta, Georgia, May 14-16, 1997, pp. 311-321.

[Han97b] JungHyun Han and Aristides Requicha, "Integration of Feature Based Design and Feature Recognition," *Computer Aided Design*, May 1997, Vol. 29, No. 5, pp. 393-403.

[Han98a] J. Han and A. A. G. Requicha, "Feature Recognition from CAD Models," *IEEE Computer Graphics and Applications*, March/April 1998, Vol. 18, No. 2, pp. 80-94.

[Han98b] JungHyun Han and Aristides Requicha, "Modeler Independent Feature Recognition in a Distributed Environment," *Computer Aided Design (Network-centric CAD: special issue)*, May 1998, Vol. 30, No. 6, pp. 453-463.

[Hart68] P. E. Hart, N. J. Nilsson, B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on SSC* 4:100-107.

[Khos94] B. Khoshnevis, J. Park and D. Sormaz, "A Cost-based System for Concurrent Part and Process Design," *Engineering Economist*, Vol. 40, No. 1, pp. 101-124.

[Maro95] P. G. Maropoulos, "Review of Research in Tooling Technology, Process Modeling and Process Planning, Part II: Process Planning," *Computer Integrated manufacturing Systems*, 8(1) pp. 13-20, 1995

[Nils80] N. J. Nilsson, *Principles of Artificial Intelligence*, Morgan Kaufmann, 1980.

[Regl95] W. C. Regli. *Geometric Algorithms for Recognition of Features from Solid Models.* PhD thesis, Computer Science Department, University of Maryland, 1995.

[Saku96] H. Sakurai and P. Dave. Volume Decomposition and Feature Recognition, Part II: Curved Objects. *Computer Aided Design.* 28(6-7):519-537, 1996.

[Shah94] J. J. Shah, Y. Shen, and A. Shirur. Determination of machining volumes from extensible sets of design features. In J. J. Shah, M. M ntyl , and D. S. Nau, editors, *Advances in Feature Based Manufacturing,* pages 129-157. Elsevier Science B. V., Amsterdam, The Netherlands, 1994.

[Simo69] H. A. Simon, *The Science of the Artificial*, The MIT Press, 1969.

[Sorm94] D. Sormaz, *Knowledge-based Integrative Process Planning System using Feature Reasoning and Cost-based Optimization*, Ph.D. Dissertation, Industrial and Systems Engineering Department, USC, 1994.

[Tsen94] Y. J. Tseng and S. B. Joshi. Recognizing Multiple Interpretations of Interacting Machining Features. *Computer Aided Design.* 26(9):667-688, 1994.

[Vand93] J. H. Vandenbrande and A. A. G. Requicha. Spatial Reasoning for the Automatic Recognition of Machinable Features in Solid Models. *IEEE Trans. Pattern Analysis and Machine Intelligence,* 15(12):1-17, 1993.