# 기하 추론 및 탐색 알고리즘에 기반한 CAD/CAM 통합

## (CAD/CAM Integration based on Geometric Reasoning and Search Algorithms)

한 정 현 † 한 인 호 ††
(JungHyun Han) (Inho Han)

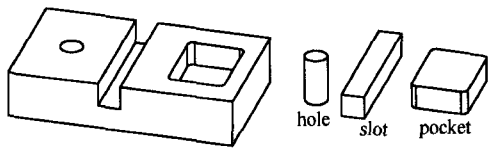
**요 약** 자동공정계획은 CAD 모델을 자동적으로 해석하여 CAM을 구동시키는 것을 목표로 하는데, 이를 위해서는 우선 CAD 모델로부터 특징형상을 인식하여야 한다. 특징형상 인식에 관한 연구는 근 20년간의 역사를 가지고 있지만, 그 연구 성과는 실용화되지 못하고 있다. 그 이유 중 하나는, 특징형상 인식과 자동공정계획 연구가 분리되어 진행되어왔기 때문이다. 본 연구에서는 인공지능 기법을 토대로 이 두 분야를 통합하여, 제조가능한 특징형상을 인식하고, 셋업을 최소화하며, 특징형상 간의 의존 관계를 설정하고, 최적의 가공 순서를 결정하였다.

***Abstract*** Computer Aided Process Planning (CAPP) plays a key role by linking CAD and CAM. Given CAD data of a part, CAPP has to recognize manufacturing features of the part. Despite the long history of research on feature recognition, its research results have rarely been transferred into industry. One of the reasons lies in the separation of feature recognition and process planning. This paper proposes to integrate the two activities through AI techniques, and presents efforts for manufacturable feature recognition, setup minimization, feature dependency construction, and generation of an optimal 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 recognize features of the part such as holes, slots and pockets. Fig. 1 shows feature examples. Therefore, feature recognition acts as a front-end of CAPP.

Recently, an important issue was raised by [9]:




(a) part (b) volumes to be removed as maching features

Fig. 1 Feature Examples

"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 an example in pocket recognition. A pocket is machined by a series of cuts, as depicted in Fig. 2. In this paper, we restrict discussions on flat end milling and ball end milling, and the pocket in Fig. 2 is made by a flat end mill. A pocket is represented by an arbitrarily-shaped planar *floor (profile)* and a sweeping vector. The sweeping vector is per-

† 정 회 원 : 성균관대학교 전기전자및컴퓨터공학부 교수
han@ece.skku.ac.kr
†† 비 회 원 : 성균관대학교 전기전자및컴퓨터공학부
ihhan@ece.skku.ac.kr
논문접수 : 1999년 7월 5일
심사완료 : 1999년 11월 1일

pendicular to the floor and its length determines the pocket's height.

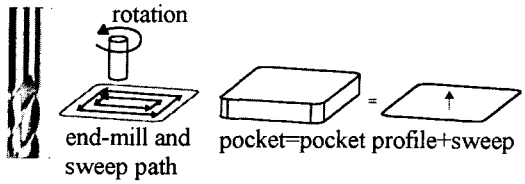
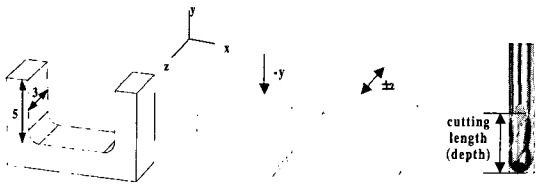



Fig. 2 Pocket Definition

The part in Fig. 3 can be decomposed by either $pocket_1$ or $pocket_2$. 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. 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.



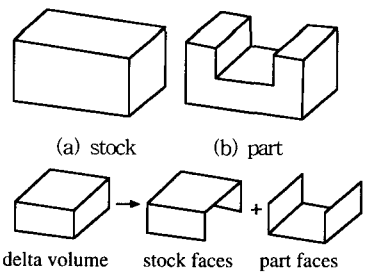

(a) part (b) $pocket_1$ (c) $pocket_2$ (d) ball end mill

Fig. 3 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 dependency among features is analyzed. This paper also shows how an optimal machining sequence can be generated by the aid of feature dependency.

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[7]. Feature recognition focused on manufacturability has been exploited at University of Illinois[5, 6]. 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[3, 4].

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[12]. 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.




(c) partition of delta volume faces

Fig. 4 Delta Volume

## 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[10, 11].

### 2.1 Trace-based Reasoning

We designed and implemented **I**ntegrated **I**ncremental **F**eature **F**inder (**IF**$^2$)[8]. **IF**$^2$ recognizes holes, slots and pockets. This paper will focus on pockets for the sake of simplicity. **IF**$^2$ is a *trace-*

*based reasoning* system. Vandenbrande and Requicha [16] claimed that a feature and its associated machining operation should leave a *trace* in the part boundary even when features intersect. For a pocket, its floor is taken as a trace. Starting from the trace, **IF**$^2$ performs extensive geometric reasoning to recognize a feature.

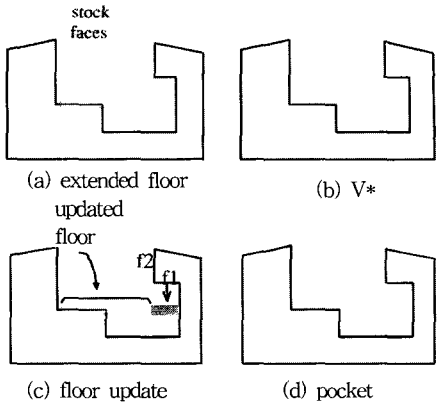



Fig. 5 Floor-based Pocket Recognition Algorithm

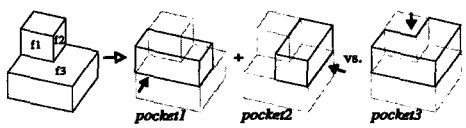



Fig. 6 Pocket Recognition Example

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

Given a floor trace, completion proceeds as shown in Fig. 5-(a) through -(d). The plane of the floor is maximally extended as shown in Fig. 5-(a). The extended floor is swept along its normal vector, to produce a volume V. A pocket removal volume V* is proposed by intersecting V with the delta volume - see Fig. 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 Fig. 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 in Fig. 5-(d).

### 2.2 Control Mechanism

From the part solid model, **IF**$^2$ detects all traces at a time and *ranks* them by assigning a heuristic strength to each trace[10]. The ranked traces constitute a priority queue. Consider the example in Fig. 6. We have three pocket traces (floors): f1, f2, and f3, which lead to *pocket1*, *pocket2* and *pocket3*, respectively. As depicted in the figure, the delta volume can be decomposed either by {*pocket3*} or by {*pocket1*, *pocket2*}. The decomposition by {*pocket3*} needs only one setup whereas the other decomposition {*pocket1*, *pocket2*} needs two setups. A smaller number of setups is preferred. In general, the less pockets we have, the less setups may be needed. Our experiments show that we are likely to recognize a small number of pockets if we give higher priorities to pocket (floor) traces with *more edges*. This heuristic gives priority to f3 over f1 or f2 in Fig. 6.

Initially, the total volume to be removed is the entire delta volume. **IF**$^2$ processes the ranked traces in order of decreasing strength until the delta volume is completely decomposed. If a trace does not lead to a valid machining feature, it is deleted and the next highest-ranked trace is extracted from the priority queue. Otherwise, **IF**$^2$ 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, **IF**$^2$ stops because the delta volume is fully decomposed. Otherwise, **IF**$^2$ takes the new top-ranked trace and repeats the same process. We call the repeated process *recognize-test cycle*.

## 3. Feature Recognition

### 3.1 Efforts for Setup Minimization

We distinguish between closed pockets and open

pockets. Note that, to a *closed* pocket, an end mill has *a single* approachable (tool axis) 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. For example, in Fig. 7-(b), the closed pocket's floor $f_4$ determines a required setup -y.

As discussed in Section 2.2, **IF**$^2$ tries to generate features which require as small number of setups as possible. When collecting traces, **IF**$^2$ pays special attention to the floor traces for *closed* pockets and obtains the *absolutely required setups* determined by their floor normals. Such an absolutely required setup **d** works as additional *evidence* for the pocket traces (floors) whose normals are **-d**, and increases their strengths.

AI uncertain reasoning techniques are useful for taking all evidence and computing the overall strength of the trace. Our current implementation uses the well-known Certainty Factor (CF) model[1], which is easy to implement and has a good experimental track record for our purposes.

Through the CF model, the pocket traces $f_1$ and $f_4$ in Fig. 7-(b) get reinforced by the additional evidence (the absolutely required setup direction) and become stronger than $f_2$ and $f_3$. The recognize-test cycle is done with the updated priority queue, and $op_1$ and $cp_1$ are recognized. We end up with $\{cp_1, op_1\}$, which requires a single setup.

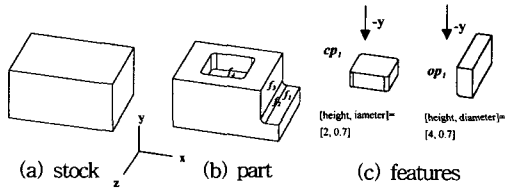
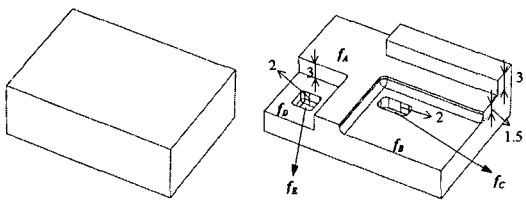



Fig. 7 Feature Recognition for Setup Minimization

### 3.2 Manufacturability and Feature Dependency

Starting from a trace, **IF**$^2$ recognizes a *maximally extended* feature volume. For example, given the stock and part in Fig. 8, the pocket trace $f_D$ leads to the maximally extended pocket $D$ shown in (c). Its height is 6 and cylindrical corner's diameter is 1, as denoted by [6, 1].




| 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

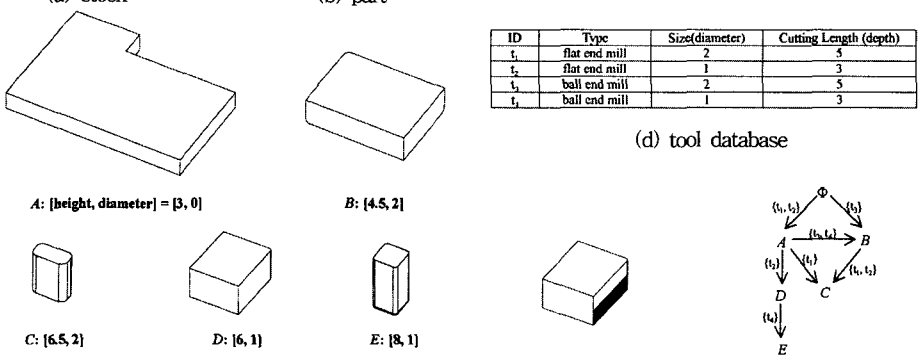
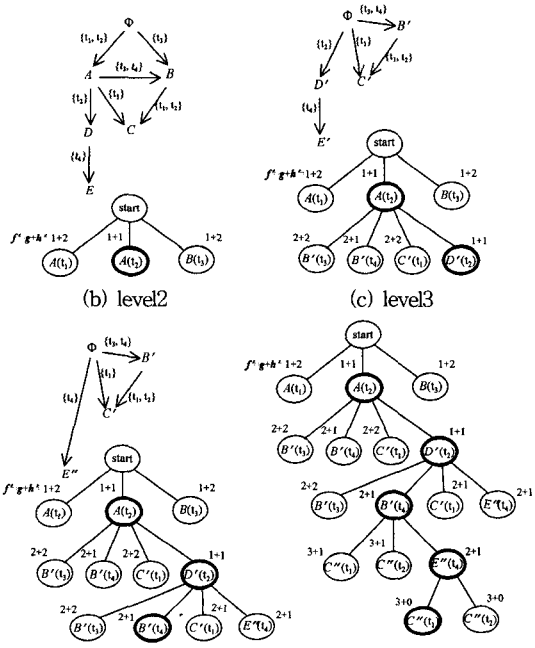



Fig. 8 Pocket Recognition Example

At every iteration of the recognize-test cycle, a feature is tested if it is manufacturable with the available tool set. Suppose that **IF**$^2$ is linked with a tool database such as the table shown in Fig. 8-(d). We can see that $D$ should be machined by a flat end mill. We find two flat end mills from the tool database, but $t_1$ is not suitable for machining $D$ because its diameter is larger than that of the cylindrical corner of $D$. In contrast, $t_2$'s diameter is identical to that of the cylindrical corner of $D$, and so might be good for machining $D$. However, $t_2$'s cutting length (depth) is 3, which is too short to machine $D$ with height 6. Then, $D$ would have to be determined 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*.

| | $t_1$ | $t_2$ | $t_3$ | $t_4$ |
|---|---|---|---|---|
| $A$ | V | V | | |
| $B$ | | | V | V |
| $C$ | V | V | | |
| $D$ | | V | | |
| $E$ | | | | V |

(a) feature-tool table




(b) level2 (c) level3

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

Fig. 9 Application of A* Algorithm

In order to determine a feature's manufacturability, **IF**$^2$ computes the portions of its wall faces which contact the part. In Fig. 8-(e), such part-contacting portions of $D$ are illustrated in a dark color[1]. It is called required volume because it is *required* to be machined even after $A$ is machined prior to $D$.

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

All pairs of such feature dependencies result in a partially ordered graph, as shown in Fig. 8-(f). In the graph, $\Phi$ represents *no pre-requisite* and therefore $A$ and $B$ are taken as manufacturable with no dependency on other features.

## 4. Machining Sequence Generation

1) **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.

Let us show how the feature dependency graph can be used for machining sequence generation. In Fig. 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.

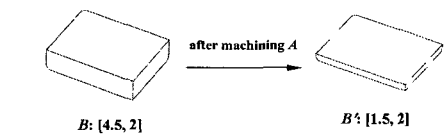
In our work, all recognized features are associated with specific setups, and we pursue an optimal machining sequence in each setup. We may then measure optimality by the sum of machining cost and tool change cost, and try to minimize it. For simplicity of discussion, we focus on tool change cost in 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[13]. In A* algorithm, we need a heuristic function $f'$ that evaluates each state we generate. 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."

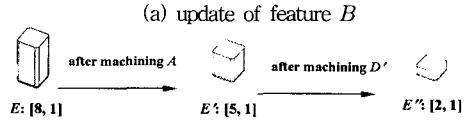
Fig. 9 shows the search trees spanned until a goal is found. Initially, there is only one state: the start state. According to the feature dependency graph shown in Fig. 8, we can start machining either *A* or *B*, which is pointed by $\Phi$ and called a *maximal* element. According to the links from $\Phi$, 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 Fig. 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 strategy* [2]. From the feature dependency graph, we can create the table in Fig. 9-(a), where all possible tools are listed for each feature. The state $A(t_1)$ says that $t_1$ is selected for machining *A*. Our greedy strategy 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 strategy 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 $h'$.

Similarly, $f'$ values for $A(t_2)$ and $B(t_3)$ are computed to be 2 and 3 respectively. 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 Fig. 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 Fig. 10-(a).




(a) update of feature *B*




(b) update of feature *E*

Fig. 10 Feature Volume Update

$B'$ 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 Fig. 9-(c), state $A(t_2)$ has four branch states: $B'(t_3)$, $B'(t_4)$, $C'(t_1)$ and $D'(t_2)$. In Fig. 9-(c), four states $B'(t_3)$, $B'(t_4)$, $C'(t_1)$ and $D'(t_2)$ are assigned $\boldsymbol{f'}$ values 4, 3, 4 and 2, respectively. Among all terminal nodes in the search tree, $D'(t_2)$ has the smallest $\boldsymbol{f'}$ 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 Fig. 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 Fig. 10-(b). If we keep searching this way, we will end up with the expanded tree shown in Fig. 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 [13]. 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 strategy. 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

A feature may not be manufacturable with the available tool set. For example, *pocket*$_2$ shown in Fig. 3 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. In actuality, manufacturability of a feature can be determined not only when the available tool set is known but also when tolerances associated with the feature are examined.

The work reported in this paper assumes that every recognized feature can be associated with an appropriate fixture. However, it is not always the case. Fixture analysis is difficult, but is essential for determining manufacturability. **IF**$^2$ is currently being extended so as to be able to do tolerance and fixture analysis.

## 6. Implementation

The proof-of-concept implementation of the algorithms discussed in this paper was done. **IF**$^2$ is written in C++ at Windows NT. In order to guarantee features manufacturability, we also add 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.

## 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. This research work also provides a framework for the utilization of tolerance and fixture information, which is indispensible for manufacturability analysis. In particular, the algorithms presented in this paper can be used in conjunction with existing software systems for process planning, for which human users manually provide feature information. We believe that this research work will serve as a stepping stone toward the ambitious goal of CAD/CAM integration.

## References

[1] D. Heckerman, Probabilistic Interpretation for

MYCIN's Certainty Factors, In Kanal and Lemmer, editors, Uncertainty in AI, pp. 167-196, Amsterdam: North-Holland, 1986.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.*

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

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

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

[10] 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.

[11] 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.

[12] 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

[13] N. J. Nilsson, *Artificial Intelligence: A New Synthesis*, Morgan Kaufmann, 1998.

[14] 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.

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

[16] 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.

한 정 현

1988년 서울대학교 공과대학 컴퓨터공학과 학사. 1991년 미국 씬씨내티대학교 전산학과 석사(Computer Science, University of Cincinnati). 1996년 미국 USC 전산학과 박사(Computer Science, University of Southern California). 1996년 ~ 1997년 미국 상무성 표준기술연구원(National Institute of Standards and Technology). 1997년 ~ 현재 성균관대학교 전기전자 및 컴퓨터공학부 조교수. 관심분야는 Solid Modeling, Computer Graphics, Computer Vision.

한 인 호

1998년 성균관대학교 공과대학 정보공학과 학사. 1998년 ~ 현재 성균관대학교 전기전자 및 컴퓨터공학부 대학원 재학중. 관심분야는 Computer Graphics, Computer Vision, Solid Modeling, CAD, Visualization.