Manufacturing Feature Recognition Toward Integration with Process Planning

Jung Hyun Han, *Member, IEEE*, Inho Han, Eunseok Lee, and Juneho Yi, *Member, IEEE*

***Abstract*—Process planning plays a key role by linking CAD and CAM. Its front-end is feature recognition, but feature recognition research has not been in accord with the requirements of process planning. This paper presents an effort for integrating the two activities: feature-based machining sequence generation primarily based on tool capabilities. The system recognizes only manufacturable features by consulting the tool database, and simultaneously constructs dependencies among the features. Then, the A^* algorithm is used to search for an optimal machining sequence by the aid of the feature dependencies and a manufacturing cost function.**

***Index Terms*—Feature recognition, machining sequence, process planning.**

I. Introduction

COMPUTER aided process planning (CAPP) links computer aided design (CAD) and computer aided manufacturing (CAM). Given CAD data of a part, CAPP generates a machining sequence to create the specified part. In order to do that, CAPP has to recognize *features* of the part such as holes, slots, and pockets. Therefore, feature recognition acts as a front-end of CAPP.

However, feature recognition research has not been in accord with the requirements of process planning. Let us take an example in pocket recognition. This paper is limited primarily to 2.5D pocket recognition. A 2.5D pocket is defined by an arbitrary-shaped *floor (profile)* and a sweeping vector perpendicular to the floor, as depicted in Fig. 1. The boundary curves of the floor are swept along the floor normal to make pocket walls. In fact, the pocket in Fig. 1 is made by a flat-end mill. However, ball-end milling is also allowed in our system such that there exist blends between the walls and the floor. Despite small variation in geometry, such a ball-end milled pocket can also be classified into a 2.5D pocket from the manufacturing viewpoint.

The part in Fig. 2(a) can be decomposed by either $pocket_1$ with the tool axis direction-y, as shown in Fig. 2(b), or $pocket_2$ with the tool axis direction $\pm z$, as shown in Fig. 2(c). Note that $pocket_1$ and $pocket_2$ are different features because they have different tool axis directions. In fact, this shows an example of

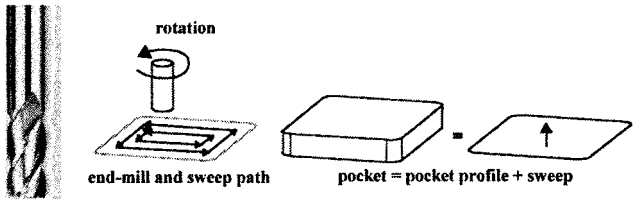


Fig. 1. Pocket definition.

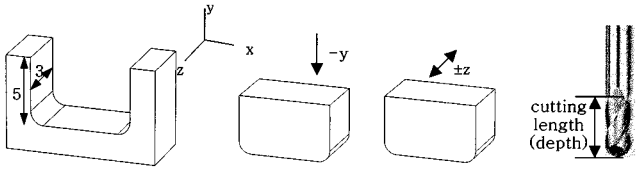


Fig. 2. Manufacturability. (a) Part. (b) $Pocket_1$. (c) $Pocket_2$. (d) Ball end mill.

multiple interpretations[1] of a part. Suppose that the heights of $pocket_1$ and $pocket_2$ (along their tool axis directions) are five and three, respectively. Suppose also that we have a single ball end mill shown in Fig. 2(d) and its cutting length (depth) is four. Then, $pocket_2$ is manufacturable with the ball end mill, whereas $pocket_1$ is not. Therefore, the part should be decomposed into $pocket_2$ and not into $pocket_1$. However, many of existing systems generate features primarily based on geometric information of the part solid model, and do not care about its manufacturability. This often results in inefficient or infeasible planning.

This paper proposes to integrate feature recognition with process planning, and presents efforts toward the integration: features are recognized with manufacturability guaranteed, dependencies among features are analyzed, and finally, an optimal machining sequence is generated by the aid of feature dependencies.

II. Related Work

The literature on feature recognition is huge, and currently active approaches include graph-based techniques, volume decomposition techniques, hint-based reasoning, etc. Surveying the huge literature on feature recognition is out of the scope of this paper; see [15] for a thorough survey. This section will briefly review the graph-based and volume decomposition

Manuscript received July 11, 1999; revised January 9, 2001. This work was supported by Grant 1999-2-515-001-5 from the Basic Research Program of the Korea Science and Engineering Foundation. This paper was recommended by Associate Editor Y. Narahari.

J. H. Han, E. Lee, and J. Yi are with the School of Electrical and Computer Engineering, Sung Kyun Kwan University, Suwon, Korea (e-mail: han@simsan.skku.ac.kr).

I. Han is with Samsung Electronics Corporation, Suwon, Korea.

Publisher Item Identifier S 1083-4419(01)05215-3.

[1]A unique representation of a part in terms of features is often called an *interpretation* of the part. A part can be represented by more than one interpretation, and *multiple interpretations* of a part roughly correspond to different ways to machine the part. Unfortunately, the nature of multiple interpretations is combinatorial. Many approaches proposed to handle multiple interpretations, such as [30], suffer from the combinatorial nature. See [11] for more discussions on this topic.

techniques. The hint-based reasoning will be discussed in Section III.

The graph-based technique was first formalized by Joshi [17], [18], where the part is represented by a graph and searched for the subgraphs that match the feature template graphs. More recently, the UK Heriot–Watt University group led by Corney has advocated graph searching algorithms for recognizing 2.5D features [4], [23].

The volume decomposition techniques decompose the input object into a set of intermediate volumes and then manipulate them to produce features. The intermediate volumes may correspond to alternating sum of volumes organized in a CSG tree [32], a set of cells generated by extending and intersecting all the surfaces/halfspaces of the part [28], or domain-independent form features to be mapped into each specific task through geometric reasoning [25].

It is widely accepted that generic process planning research is saturated, but research based on feature technologies is required to enhance the state-of-the-art [24]. An integrated system for feature recognition and process planning was developed at the University of Maryland [10]. Feature recognition focused on manufacturability has been exploited by Gaines [7], [8]. 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 [5]. In [22], feature precedence relations are generated using face dependency information and the features' associated machining process information. Reference [21] presented a process planning system based on feature reasoning.

The system presented in this paper generates an optimal machining sequence based on feature dependencies. However, such an idea is not novel and there have been similar efforts reported in [6], [9], [10], [25], [26], and [29].

In the work at the University of Maryland [9], [10], [26], multiple interpretations are generated through *feature algebra* [19] or a *set-covering* procedure [3], and then multiple machining sequences are produced from them using feature dependencies based on accessibility constraints. Unfortunately, feature algebra does not necessarily generate the interpretations leading to optimal machining sequences as the features and the volumes associated with them are largely predefined. In contrast, the set-covering procedure is subject to combinatorial explosion, as pointed out by [11]. Unlike these approaches, our system first constructs a dependency network among all recognized features, and then searches for an optimal interpretation which corresponds to a machining sequence.

Sormaz [29] manually defined all possible feature dependencies: one for a pair of features. However, this kind of enumeration is vulnerable in that the enumeration should be manually extended when a set of new features is taken into account.

Easwanah, in his Ph.D. dissertation [6], introduced the concept of layers to provide a way to represent feature dependencies such that the features accessible/machinable at the start of machining are placed at layer 1, all features which are accessible after the features in layer 1 have been machined are placed at layer 2, and so on. Unlike our approach, his work is focused on attaching tolerances and surface-finish information to a feature-based model and using it subsequently for automating tasks such as process selection.

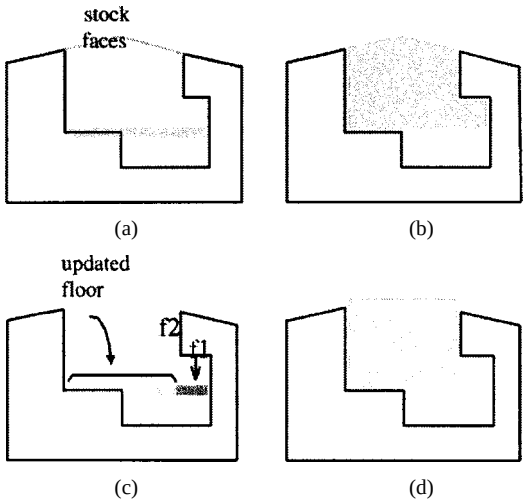


Fig. 3. Floor-based pocket recognition algorithm.

In the work by Narayan [25], domain-independent form (DIF) features are recognized from the input model and mapped into each specific task such as machining. A visibility-based approach is used to find the accessibilities of features. A globally accessible feature can be created independently of other features. In contrast, a globally inaccessible feature can become machinable after processing another. This leads to dependency relations among features. All possible machining sequences can be enumerated from the dependencies. The enumerations can then be pruned using some constraints such as tool availability and/or tool-machine combination. Unlike Narayan's, the approach presented in this paper recognizes globally-accessible features from scratch as discussed in Section III-A, constructs dependencies among them based on tool availabilities, and then generates an optimal machining sequence.

III. Geometric Reasoning for Feature Recognition

This section briefly overviews the feature recognition kernel of our system, which has already been published in literature [12], [13].

A. Hint-Based Reasoning

In feature recognition, the most important issue has been handling *intersecting features*. For this purpose, Vandenbrande and Requicha [31] proposed *hint-based reasoning*. A hint implies a feature's potential existence. For example, the hint for a pocket is defined as its floor, and the pocket is recognized from the floor through geometric reasoning. This paper presents integrated feature finder ($\mathbf{IF}^2$) built upon a hint-based reasoning kernel [13].

Fig. 3 illustrates the steps in pocket recognition. The reasoning steps follow the *generate-test-repair* paradigm. The material to be removed by machining, called *delta volume*, is obtained by subtracting the part from the stock. The *generate* step maximally extends a floor within the delta volume [Fig. 3(a)], infinitely sweeps the extended floor along the floor's normal

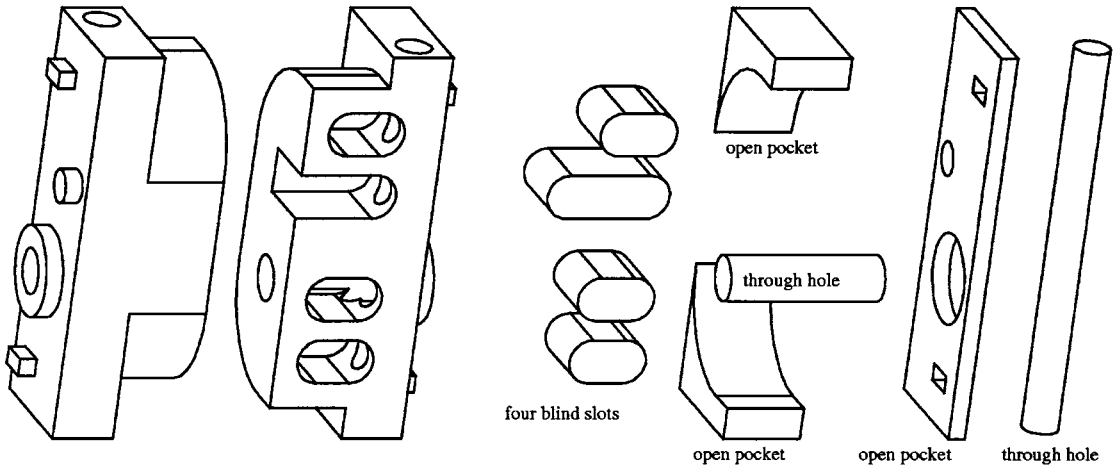


Fig. 4. Feature recognition example.

vector, and then intersects it with the delta volume to produce a tentative removal volume V^*, as shown in Fig. 3(b). Then, the boundary of V^* is *tested* to collect a set of faces that makes V^* globally inaccessible. If such faces are found, they are used to *repair* V^*. They are projected on the extended floor, as shown in Fig. 3(c), and subtracted from the floor. Again, the updated floor is swept along the normal and intersected with the delta volume to get a volume V. A local coordinate system is associated with V 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, to instantiate a pocket in Fig. 3(d).

Fig. 4 shows an industrial part with features heavily intersected. $\mathbf{IF}^2$ decomposes the part into nine 2.5D features such as holes, slots, and pockets.

B. Control Mechanism

From the part solid model, $\mathbf{IF}^2$ detects all hints at a time and ranks them by assigning a heuristic strength to each hint [12]. (See the Appendix for the hint-ranking mechanism.) The ranked hints constitute a priority queue.

Initially, the total volume to be removed is the entire delta volume. $\mathbf{IF}^2$ 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, $\mathbf{IF}^2$ 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, $\mathbf{IF}^2$ stops because the delta volume is fully decomposed. Otherwise, $\mathbf{IF}^2$ takes the new top-ranked hint and repeats the same process. We call the repeated process *recognize-test cycle*.

IV. Dependencies Among Manufacturable Features

Section III briefly overviewed $\mathbf{IF}^2$s geometric reasoning kernel and control mechanism. Sections IV and V will present recent research results, implemented in $\mathbf{IF}^2$, toward the integration with process planning.

A. 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. Then, such a direction leads to a *required setup* in three-axis machining. For example, in Fig. 5(b), the closed pocket's floor f_E determines a required setup -y.

$\mathbf{IF}^2$ tries to generate features which require as small number of setups as possible because setup operations are costly. When collecting hints $\mathbf{IF}^2$ pays special attention to the floor hints for closed pockets, [2] and obtains the required setups determined by their floor normals. Such a required setup direction $\mathbf{d}$ increases the ranks of the pocket hints (floors) with a normal $-\mathbf{d}$, through the process discussed in the Appendix, and the recognize-test cycle is done with the newly ranked hints.

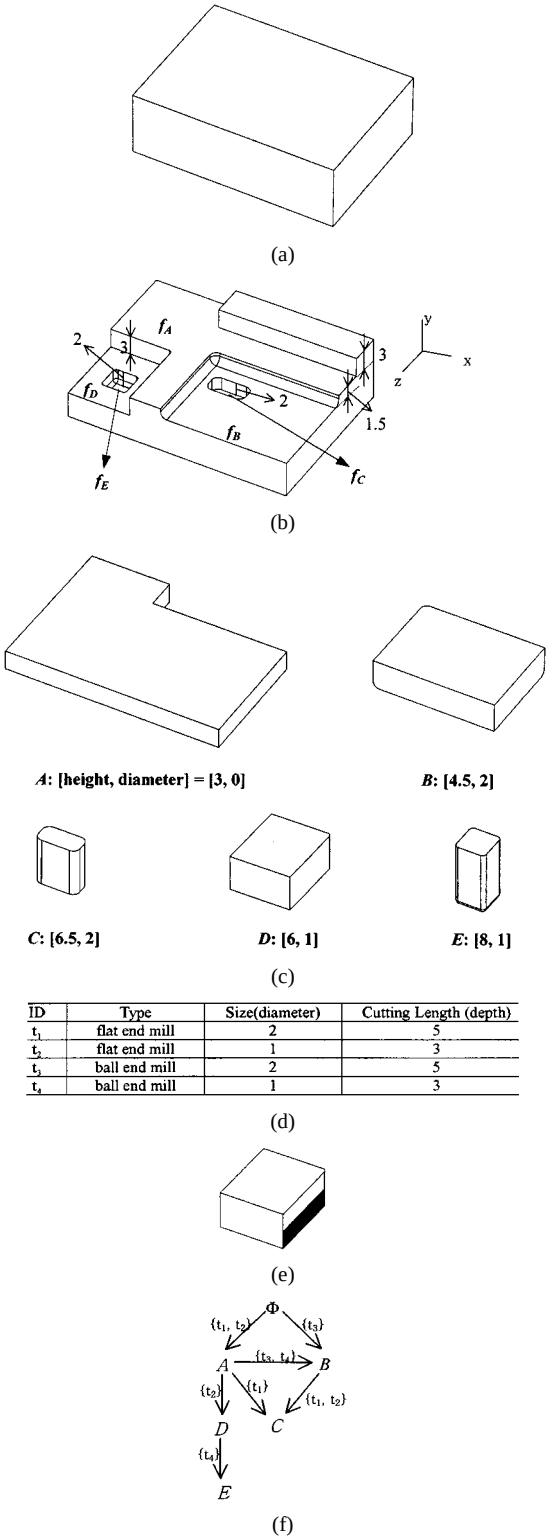
With this mechanism, the part in Fig. 5 is decomposed by A, B, C, D, E, which are all accessible in a single setup -y. However, if a part is not completely decomposed even after all hints reinforced by a required setup are processed, the recognize-test cycle will take hints at a different setup. Then, the part needs multiple setups.

B. Manufacturability and Feature Dependency

Recall that $\mathbf{IF}^2$ recognizes a maximally extended feature volume. For example, given the stock and part in Fig. 5, the pocket hint f_D leads to D, shown in Fig. 5(c), which is extended vertically up to the stock boundary. Its height is six and cylindrical corner's diameter is one, as denoted by D: [6,1].

At every iteration of the recognize-test cycle, a feature is tested if it is manufacturable with the available tool set. Suppose that $\mathbf{IF}^2$ is linked with a tool database such as the table shown in Fig. 5(d). We can see that D should be machined by a flat end mill. $\mathbf{IF}^2$ finds two flat end mills t_1 and t_2 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 might be good for machining D. However, t_2's cutting length (depth) is three, which is too short to machine D

[2]Floor hints for closed pockets can be easily found. If every edge of a floor is shared by another face at a concave angle, the floor is a hint for a closed pocket.



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

Fig. 5. Pocket recognition example.

of height six. Then, D would have to be determined nonmanufacturable with the available tool set. Suppose that, however, A is machined prior to D. Then, D turns out to be manufacturable because its height is reduced to three as a result of machining A. This relation leads to *feature dependency*.

The portions of D's wall faces which contact the part are illustrated in a dark color in Fig. 5(e). These portions lead to a required volume of a feature. It is required to be machined even after A is machined prior to D. Such a required volume's range along the pocket floor normal direction can be obtained by computing an enclosing box along the normal direction, as discussed in Section III-A.

In $\mathbf{IF}^2$, feature dependency construction is tightly coupled with manufacturability test. 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.) $\mathbf{IF}^2$ then collects every appropriate mill 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. (For D and t_2, A is collected.) Check if $\boldsymbol{Q}$'s floor contains $\boldsymbol{P}$'s floor when they are projected along the tool axis direction. (It is the case for D and A.) If so, $\mathbf{IF}^2$ 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 dependencies result in a partially ordered graph, as shown in Fig. 5(f). It is interesting to see which features C depends on: $A \rightarrow C$ with respect to $\{t_1\}$ and $B \rightarrow C$ 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.

In the graph, Φ represents no prerequisite, and therefore, A and B are taken as manufacturable with no dependency on other features. Even for such features, however, $\mathbf{IF}^2$ constructs dependencies. The graph shows that B can be machined with t_3 without considering any other features (as denoted by $\Phi \rightarrow B$) ***or*** can be machined with either t_3 or t_4 if A is removed first (as denoted by $A \rightarrow B$). Dependency construction for the immediately manufacturable feature B allows us to use one more tool (t_4) and therefore helps us achieve optimization for process planning.

V. Machining Sequence Generation

A. Manufacturing Cost Function

In order to determine an optimal machining sequence, we have to consider setup cost, tool change cost, machining cost, etc. There have been a lot of efforts to devise manufacturing cost functions with them. See [20] for an example.

In our work, all recognized features are associated with specific setups, and we pursue an optimal machining sequence in each setup. For the sake of simplicity, we use the following manufacturing cost function in the current implementation.

$$C_{total} = C_{tc} + \sum_{i=1}^{n} C_{fi} = \alpha N_{tc} + \sum_{i=1}^{n} \beta V_{fi}$$

where

C_{tc} total cost for tool changes;
C_{fi} machining cost for each feature f_i;
α cost per a tool change;
N_{tc} number of tool changes;
β cost coefficient per a unit volume;
V_{fi} feature f_i's volume.

Obviously, this function is overly-simplified. For example, the machining cost C_{fi} is made proportional to f_i's volume, and therefore remains constant regardless of the selected tool's dimensions. In reality, we should select appropriate cutting parameters such as feed given the part material, the feature type/dimensions and the tool material/dimensions, and then compute the machining cost with the parameters. However, the goal of this paper is to present a framework for generating an optimal machining sequence, and this framework allows $C_{fi} = \beta V_{fi}$ to be immediately replaced by any well-developed machining cost function.

We will show how the above cost function and the feature dependency graph can be used for generating a machining sequence. For simplicity of discussion, we focus on tool change cost in this paper. Typically, tool change cost is negligible compared to machining cost, but focusing on the tool change cost enhances the clarity of the algorithm.

B. State Space Search

An *optimal* machining sequence can be formalized as the optimal path from the start state (stock) to a goal state (part), and can be found by A^* algorithm which was first presented by [16]. In A^* algorithm, we need a heuristic function $\boldsymbol{f'}$ that evaluates each state we generate. The prime on $\boldsymbol{f}$ indicates that it is an approximation to a function $\boldsymbol{f}$ that gives the true evaluation of the state. The function $\boldsymbol{f'}$ is defined as the sum of $\boldsymbol{g}$ and $\boldsymbol{h'}$ where $\boldsymbol{g}$ is a measure of the cost of getting from the start state to the current state and $\boldsymbol{h'}$ is an evaluation of the additional cost of getting from the current state to a goal state. In our application, $\boldsymbol{g}$ is a measure of "how many tool changes have occurred," and $\boldsymbol{h'}$ is a guess of "how many tool changes will occur."

Fig. 6 shows the search trees spanned until a goal is found. Initially, there is only one state: the start state. In the dependency graph computed in Fig. 5, A and B are pointed by Φ and often called *maximal* elements. A can be machined by either t_1 or t_2, whereas B can be only by t_3. Therefore, as shown in Fig. 6(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.

Let us compute $\boldsymbol{g}$ and $\boldsymbol{h'}$ for the state $A(t_1)$. Installation of t_1 is counted as a tool change, and therefore $\boldsymbol{g}$ is set to 1. For computing $\boldsymbol{h'}$, we repeatedly use a *greedy strategy* [3]. The greedy strategy proposes that, as t_1 is already selected for machining A, all remaining features that can be machined by t_1 be machined by it. For this purpose, we use a table created from the feature dependency graph, shown in Fig. 6(a), where all possible tools are listed for each feature. The table shows that C is in t_1's column. Then, A and C are assumed to be machined out by t_1, but B, D, and E remain. Computing $\boldsymbol{h'}$ 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

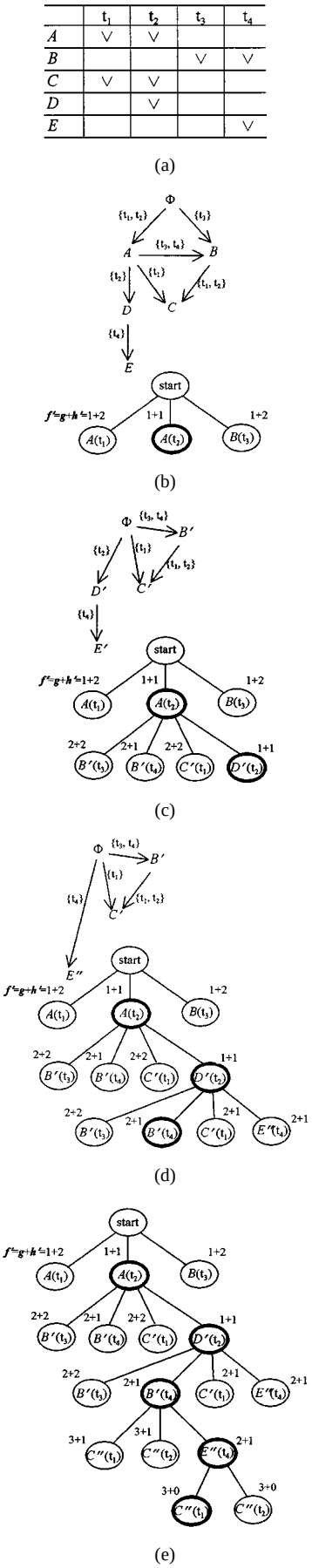


Fig. 6. Application of A* algorithm.

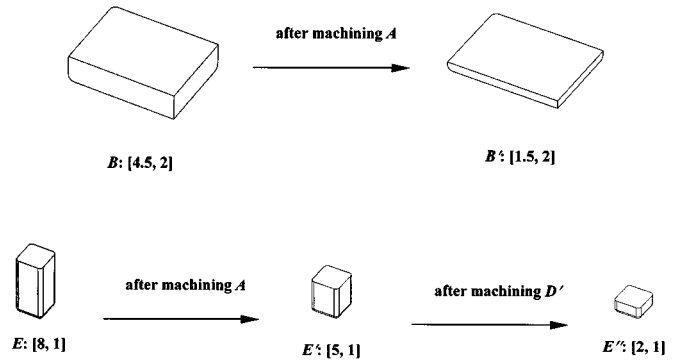


Fig. 7. Feature volume update.

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 $\boldsymbol{h}'$ to two: i.e., from t_1 to t_4, and then to t_2. Therefore, $\boldsymbol{f}'$ is set to three, which is the sum of $\boldsymbol{g}$ and $\boldsymbol{h}'$.

The same procedure sets $\boldsymbol{f}'$ of $A(t_2)$ to two, and $\boldsymbol{f}'$ of $B(t_3)$ to three. 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. 6(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. 7. For the tool path generation at CNC machines, we have to use the reduced feature volume B' instead of B so as possibly reduce the number of passes, etc. In the current implementation, considering B' will reduce machining cost because the machining cost is made proportional to feature volume, as discussion in Section V-A.

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. 6(c), state $A(t_2)$ has four branch states: $B'(t_3)$, $B'(t_4)$, $C'(t_1)$, and $D'(t_2)$. Their $\boldsymbol{f}'$ values are set to four, three, four, and two, respectively. Among all six terminal nodes in the search tree, $D'(t_2)$ has the smallest $\boldsymbol{f}'$ value, and so is expanded at the next stage. When D' is machined out, B', C', and E'' become new maximal elements as shown in Fig. 6(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. 7. Fig. 6(d) shows their $\boldsymbol{f}'$ values. $B'(t_4)$, $C'(t_1)$, and $E''(t_4)$ are assigned three. Note that their $\boldsymbol{f}'$ 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 break this tie in favor of the path we are currently following such that we can avoid *backtracking*. 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 Fig. 6(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 always finds an optimal path to a goal, if one exists [27]. 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 less than or equal to $\boldsymbol{h}$, i.e., $\boldsymbol{h}'$ can never be an overestimate. Consequently, the A^* algorithm always generates an optimal machining sequence with the minimum number of tool changes.

VI. Discussion

So far, feature recognition research has largely focused on finding all possible features, and all other important tasks such as manufacturability analysis are shifted to process planners. This scheme 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 and feature dependency construction. The feature dependencies, together with the manufacturing cost functions, guide the state space search for an optimal machining sequence.

Manufacturability analysis should consider a number of constraints. For example, when the entry face of a pocket is not perpendicular to the tool axis direction, its machining cost will increase. The analysis module may then use the increased cost to calculate the total manufacturing cost. However, this paper discusses only tool availability in order to determine a feature's manufacturability. Note that the goal of this paper is to provide a framework where an optimal machining sequence is generated through the state space search guided by feature dependencies and manufacturing cost function. We believe that this framework can be extended to incorporate other constraints such as that mentioned above.

We restrict discussions to a set of features accessible in a single setup. This restriction leads to an important implication: The resulting machining sequence will be a local optimum. Given a part which requires multiple setups or tool orientations, we do not necessarily achieve a global optimum simply by collecting the locally optimal machining sequences. Achieving a global optimum is an intriguing possibility, and $\mathbf{IF}^2$ is currently being extended toward this goal.

VII. Implementation

The proof-of-concept implementation of the algorithms discussed in this paper was completely done. $\mathbf{IF}^2$ is written in C++ and runs at Windows NT on PC. $\mathbf{IF}^2$ obtains geometric services from the Parasolid modeler, and the interface between $\mathbf{IF}^2$ and Parasolid is discussed in [14]. $\mathbf{IF}^2$ cooperates with the tool database through Microsoft ODBC APIs.

VIII. Conclusion

Features play a key role in integrating CAD and process planning, but feature recognition research has not been guided by the requirements of process planning. For example, much of the manufacturing knowledge, which is typically used in process planning, is rarely incorporated into feature recognition. This paper proposes to integrate the two activities, and presents a framework for feature recognition for manufacturability and setup minimization, feature dependency construction, generation of an optimal feature-based machining sequence, etc. Analysis of more constraints including tolerance and setup/fixturing will enhance the framework.

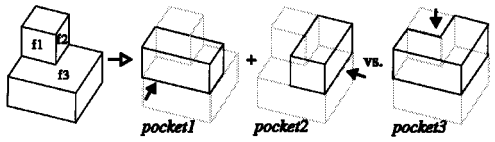


Fig. 8. Hint ranking.

APPENDIX

IF^2 assigns a heuristic strength to each hint. Consider the example part in Fig. 8. We have three pocket hints (floors): f_1, f_2, and f_3, which will lead to $pocket_1$, $pocket_2$, and $pocket_3$, respectively. The part has two interpretations: $\{pocket_3\}$ requiring a single setup and $\{pocket_1, pocket_2\}$ requiring two setups. A smaller number of setups is preferred because setup operations are costly and adversely affect precision. In general, the fewer 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) hints with more edges. This heuristic gives priority to f_3 over f_1 or f_2 in Fig. 8.

The ranked hints constitute a priority queue, but the queue can be updated with incoming pieces of evidence. For this purpose, AI uncertain reasoning techniques are useful, and our current implementation uses the well-known certainty factor (CF) model [1]. Although the CF model has been shown not to correspond to a more principled probabilistic reasoning [2], it is easy to implement and has a good experimental track record for our purposes. Heckerman [2] defined CF(H, E) as follows:

$$\mathrm{CF}(H, E) = \begin{cases} \dfrac{p(H|E) - p(H)}{p(H|E)(1 - p(H))}, & \text{if } p(H|E) > p(H) \\ \dfrac{p(H|E) - p(H)}{p(H)(1 - p(H|E))}, & \text{if } p(H|E) < p(H) \end{cases}$$

where $p(H)$ is the *prior* probability of a hypothesis H and $p(H|E)$ is the *posterior* probability of H given evidence E. CF(H, E) normally represents the change in belief in H based on E. Given two pieces of evidence E_1 and E_2, which contribute to H with positive CF values X and Y, respectively, the CF for the combined evidence is given by [1] as $X+Y(1-X)$.

IF^2 assigns a range of CF values to a group of hints-of-a-type, e.g., a group of pocket hints. Individual hints in the group are ranked according to the above heuristics discussed above. As mentioned in Section IV-A, if a closed pocket hint is found, it is taken as an additional piece of evidence for every pocket hint of the same normal as the closed pocket hint. Contribution of such evidence is assessed using the rule of [1]. For more detailed discussions on hint ranking, see [12].

REFERENCES

[1] B. G. Buchanan and E. H. Shortliffe, *Rule-Based Expert Systems*. Reading, MA: Addison–Wesley, 1984.
[2] D. Heckerman, "Probabilistic interpretation for MYCIN's certainty factors," in *Uncertainty in AI*, Kanal and Lemmer, Eds. Amsterdam, The Netherlands: North-Holland, 1986, pp. 167–196.
[3] T. H. Cormen *et al.*, *Introduction to Algorithms*. New York: McGraw–Hill, 1990.
[4] J. Corney, "Graph-based feature recognition," Ph.D. dissertation, Heriot Watt Univ., Edinburgh, U.K., 1993.
[5] J. Dong and S. Vijayan, "Manufacturing feature determination and extraction—Part 1: Optimal volume segmentation," *Computer-Aided Design*, vol. 29, no. 6, pp. 427–440, June 1997.
[6] P. Easwanah, "Some studies on CAPP for prismatic components," Ph.D. dissertation, Indian Inst. Technol., Delhi, India, 1997.
[7] D. M. Gains and C. C. Hayes, "CUSTOM-CUT: A customizable feature recognizer," *Computer-Aided Design*, vol. 31, no. 2, pp. 85–100, Feb. 1999.
[8] D. M. Gains *et al.*, "MEDIATOR: Reconfigurable feature recognition for a maintainable, extensible CAD/CAPP integration," *ASME J. Mech. Des.*, vol. 121, no. 1, pp. 145–158, 1999.
[9] S. Gupta and D. Nau, "Generation of alternative feature-based models and precedence ordering for machining applications," in *Proc. Third ACM SIGGRAPH Symp. Solid Modeling Applicat.*, Montreal, QC, Canada, 1993, pp. 465–466.
[10] S. K. Gupta, "Automated manufacturability analysis of machined parts," Ph.D. dissertation, Univ. Maryland, College Park, Dept. Mech. Eng., 1994.
[11] J. Han, "On multiple interpretations," in *Proc. Fourth ACM SIGGRAPH Symp. Solid Modeling Applicat.*, Atlanta, GA, May 14–16, 1997, pp. 311–321.
[12] J. Han and A. Requicha, "Integration of feature based design and feature recognition," *Computer-Aided Design*, vol. 29, no. 5, pp. 393–403, May 1997.
[13] J. Han and A. A. G. Requicha, "Feature recognition from CAD models," *IEEE Comput. Graph. Applicat.*, vol. 18, pp. 80–94, Mar./Apr. 1998.
[14] J. Han and A. Requicha, "Modeler independent feature recognition in a distributed environment," *Computer-Aided Design (Network-Centric CAD: Special Issue)*, vol. 30, no. 5, pp. 453–463, May 1998.
[15] J. Han *et al.*, "Manufacturing feature recognition from solid models: A status report," *IEEE Trans. Robot. Automat.*, vol. 16, pp. 782–796, Dec. 2000.
[16] P. E. Hart *et al.*, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Solid-State Circuits*, vol. SSC-4, pp. 100–107, Jan. 1965.
[17] S. Joshi and T. Chang, "Graph-based heuristics for recognition of machined features from a 3D solid model," *Computer-Aided Design*, vol. 20, no. 1, pp. 58–66, Jan. 1988.
[18] S. Joshi *et al.*, "Expert process planning system with solid model interface," *Int. J. Prod. Res.*, vol. 26, pp. 863–885.
[19] R. Karinthi and D. Nau, "An algebraic approach to feature interactions," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, pp. 469–484, Apr. 1992.
[20] B. Khoshnevis *et al.*, "A cost-based system for concurrent part and process design," *Eng. Econ.*, vol. 40, no. 1, pp. 101–124.
[21] ——, "An integrated process planning system using feature reasoning and space search-based optimization," *Proc. Inst. Elect. Eng.*, vol. 31, pp. 597–616, July 1999.
[22] Y. Kim *et al.*, "Feature-based machining precedence reasoning and sequence planning," in *Proc. ASME Design Eng. Tech. Conf.*, 1998.
[23] G. Little *et al.*, "Delta-volume decomposition for multi-sided components," *Computer-Aided Design*, vol. 30, pp. 695–705, 1998.
[24] P. G. Maropoulos, "Review of research in tooling technology, process modeling and process planning—Part II: Process planning," *Comput. Integr. Manufact. Syst.*, vol. 8, no. 1, pp. 13–20, 1995.
[25] A. Narayan, "Feature-based geometric reasoning for process planning," *Sadhana*, vol. 22, no. 2, pp. 217–240, 1997.
[26] D. Nau *et al.*, "Generation and evaluation of alternative operation sequences, quality assurance through integration of manufacturing process and systems," in *Proc. ASME Winter Annu. Meet.*, vol. PED-56, 1992, pp. 93–108.
[27] N. J. Nilsson, *Principles of Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann, 1980.
[28] H. Sakurai and P. Dave, "Volume decomposition and feature recognition—Part II: Curved objects," *Computer-Aided Design*, vol. 28, no. 6–7, pp. 519–537, June/July 1996.
[29] D. Sormaz, "Knowledge-based integrative process planning system using feature reasoning and cost-based optimization," Ph.D. dissertation, Ind. Syst. Eng. Dept., Univ. Southern California, Los Angeles, 1994.
[30] Y. J. Tseng and S. B. Joshi, "Recognizing multiple interpretations of interacting machining features," *Computer-Aided Design*, vol. 26, no. 9, pp. 667–688, Sept. 1994.

[31] J. H. Vandenbrande and A. A. G. Requicha, "Spatial reasoning for the automatic recognition of machinable features in solid models," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, pp. 1–17, Dec. 1993.
[32] D. Waco and Y. Kim, "Geometric reasoning for machining features using convex decomposition," *Computer-Aided Design*, vol. 26, no. 6, pp. 477–489, June 1994.

Jung Hyun Han (S'96–M'97) received the B.S. degree in computer engineering from Seoul National University, Korea, in 1998, the M.S. degree in computer science from the University of Cincinnati, Cincinnati, OH, in 1991, and the Ph.D. degree in computer science from the University of Southern California, Los Angeles, in 1996.

Since 1997, he has been an Assistant Professor with the School of Electrical and Computer Engineering, Sung Kyun Kwan University, Suwon, Korea, and directs the Computer Graphics Laboratory. Prior to joining the University, he worked at the National Institute of Standards and Technology. His research interests include CAD/CAM, computer graphics, and computer vision.

Inho Han received the B.S. and M.S. degrees, both in information engineering, from Sung Kyun Kwan University, Suwon, Korea, in 1998 and 2000, respectively.

He is a Researcher at Samsung Electronics Corporation, Suwon. His research interests include computer graphics, computer vision, CAD/CAM, and MPEG.

Eunseok Lee received the B.S. degree in electronic engineering from Sung Kyun Kwan University Suwon, in 1985 and the M.S. and Ph.D. degrees, both in information engineering, from Tohoku University, Sendai, Japan, in 1998 and 1992, respectively.

Since 1995, he has been an Associate Professor with the School of Electrical and Computer Engineering at Sung Kyun Kwan University and directs the Software Engineering Laboratory. Previously, he was a Faculty Member at Tohoku University and a Research Scientist at Mitsubishi Electronics. His research interests include software/usability engineering, agent-based intelligent systems, and electronic commerce.

Juneho Yi (S'93–M'95) received the B.S. degree from Seoul National University, Seoul, Korea, in 1985, the M.S. degree from Pennsylvania State University, University Park, in 1987, and the Ph.D. degree from Purdue University, West Lafayette, IN, in 1994, all in electrical engineering.

He is currently an Assistant Professor with the School of Electrical and Computer Engineering, Sung Kyun Kwan University, Suwon, Korea. From 1995 to 1996, he was a Senior Research Scientist at the Korea Institute of Science and Technology, Seoul. From 1994 to 1995, he was a Postdoctoral Research Scientist at the University of California, Riverside. In 1989, he was a Research Scientist at the Samsung Advanced Institute of Technology. His research interests include computer vision and vision-based computer interaction.