144 KSME Journal, Vol. 8, No. 2, pp. 144~151, 1994

A System for Recognition and Modification of Machining Features on Prismatic Components

M. W. Park*, H. Ko*, Y. Sohn* and H. Kang*

(Received July 14, 1993)

A recognition process of the features in a part model resorts to the knowledge of an application engineer. The knowledge is encoded as rules of the recognition procedure at the beginning that are applied to the part model during the recognition process. Such a human interaction is difficult to control in extracting the intended features because the intended features by the application engineer may change from one engineer to the other and the external situations. Instead, we treat the result of the recognition process as a rough extraction and allow the user interactively modify the result. In this paper, we present a feature recognition system where the user can inspect the result of the recognition and delete recognized features interactively.

Key Words: Feature, CAD, CAPP

1. Introduction

Most feature recognition(FR) methods developed so far are effective with nonintersecting features in a polyhedral geometries and experience difficulties with a part model containing a free-form surfaces or interacting features although some limited remedies have been suggested : face extension method(Dong, 1991 ; Sakurai, 1990), joining(Kim, 1991) and others. Furthrmore, a tolerance, surface finish, and other techonological information must be taken into account in order to extract the intended features.

Most FR methods extract the features from a part model based on a fixed set of recognition rules and the resulting feature data of the recognition process is not easily modified. That is, the only channel of user interaction in the recognition process is by encoding the user's preference in the recognition rules and let the recognition rules take over the entire recognition process, a batch style. This form of user interaction is indirect and inconvenient.

* CAD/CAM Laboratry, Korea Institute of Science and Technology, Cheong Ryang PO Box 131, Seoul, Korea

This paper presents an altenative user interaction paradigm where the user is allowed to modify the result of the recognition process by inspecting the result of the recognition and interactively deleting the recognized features.

A feature deletion has been used in recognizing compound features(Dong, 1988 ; Park, 1990), an add-material operation to enclose the lower level features. The operation was applied using a fixed set of criteria during the recognition process.

A feature deletion operation is more commonly found in a feature-based modeling(FBM) system where a user specifies a feature to insert and both the feature and its associated geometric information are created by the system. A feature deletion is performed by retracing the history of feature insertion operations because the feature information is encoded implicitly by a sequence of procedures calls without any explicit links to the underlying geometric information(Luby, 1986 ; Shah, 1988). This feature deletion methodology is incompatible to use for modifying the result of FR process. There are $n!$ possible feature insertion sequences corresponding to n recognized features and each of them must be compared to the original whether the resulting part model of each sequence correspond to the part model from

which the features are recognized. Here, we developed a feature deletion method based on the add-material operation. The deletion is applied interactively to the result of the recognition process rather than being applied during the recognition process using add-material type operations that are defined a priori.

As a part of the development of an integrated CAD/CAM system for mould die manufacturing, a software system is developed, which recognizes features on prismatic components designed on a core solid modeler-ACIS geometric modeling kernel, and is able to interactively modify the part model by inspecting and deleting recognized features. The recognition methodology is explained in the next section. The inspection and the deletion of features will be described in the Secs. 3 and 4 respectively. Application of the developed system to an example case is given in the Sec. 5. The example is a real part of the injection mould die set which comprises of 2.5 dimension features.

2. Feature Recognition

2.1 Feature library

The feature libray contains a predefined generic features that are volumetric ; it consists of depression, protrusion, and through features that are commonly found in mould manufacturing application. The system currently handles 21 elementary features on prismatic components but the feature set can be expanded. 6types of pockets, 4types of holes, 4types of slots, 3types of steps and 4type of islands are shown in Fig. 1 where the top and side view are shown for each feature.

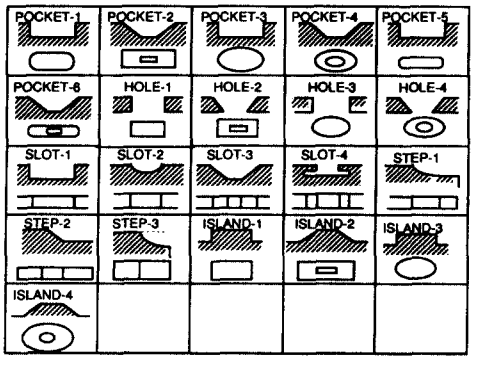


Fig. 1 Geometry of elementary features

The shape of an elementary feature is defined by size dimensions and it is positioned in the part model by positioning parameters. The parameters associated with POCKEY-1 and HOLE-3 in Fig. 1, are shown in Fig. 2.

The elementary features are defined as classes of C++language in two levels. A base class contains field pointers that are commonly needed across all elementary features like parent/child features, brother features, shown/closure faces, and the common parameters as shown in Fig. 3. Their usage will be described shortly.

A derived class of an elementary feature contains parameters specific to the feature. For example, a derived class of the base feature class for POCKET-1 in Fig. 2 includes the direction vector

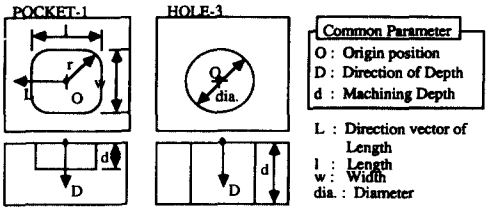


Fig. 2 Feature parameters

```
class  Feature

    protected:
    char        *feature_name;        // Points to name of atomic feature
    Feature     *parent_feature;      // Points to parent feature
    Feature     *child_feature;       // Points to child feature
    Feature     *next_feature;        // Points to next brother
    Feature     *previous_feature;    // Points to previous brother
    List_of     *shown_faces;         // Points to list of feature's faces
    List_of     *closure_faces;       // Points to list of closure faces
    vector      *Origin;              // Points to position vector of origin
    vector      *Direction_D;         // Points to direction vector D
    double      Depth;                // Machining depth or height
};
```

Fig. 3 Base class for feature

```
class  Pocket_1 : public Feature
{
    protected:
    vector      *Direction_L;         // Points to direction vector L
    double      Width;                // Width of feature
    double      Length;               // Length of feature
    double      Radius_Of_Corner;     // Corner radius
};
class  Hole_3 : public Feature
{
    protected:
    double      Diameter;             // Diameter of hole
};
```

Fig. 4 Derived classes for elementary features

of length(L), width(w), length(l), and corner radius(r); that of HOLE-3 includes the diameter(dia). Their class definitions are given in Fig. 4.

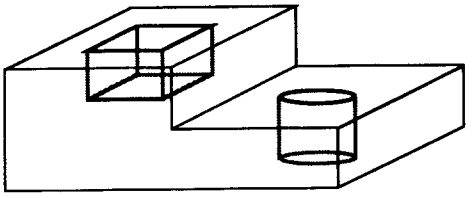


Fig. 5 Part model

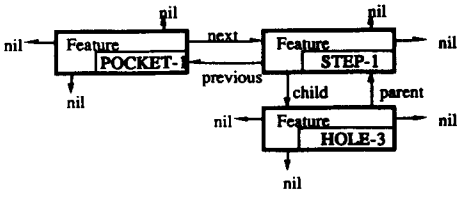


Fig. 6 Feature hierachy

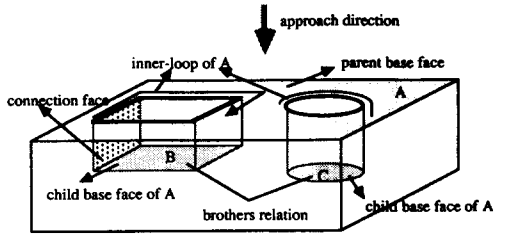
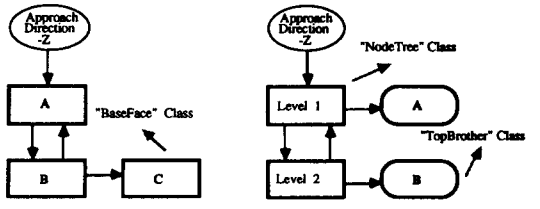


Fig. 7 Hierarchical relations between base faces



(a) Hierarchy of base faces
(b) Nood tree of the base faces hierachy

Fig. 8 Base face hierachy

```
PATTERN POCKET-1:    // Recognition rule of POCKET-1
  Equal(Bounding(loop_BaseFace), Bounding(loop_parent_BaseFace))
  Num_Planar_ConFace(loop_BaseFace) == 4
  Exist(Cyl_ConFace)
    Straight_Edge_Parallel(Cyl_ConFace)
    Adjacent_Planar_Cyl_Planar(Cyl_ConFace)
  RECOGNIZED POCKET-1:
```

Fig. 9 Rule for "POCKET-1"

2.2 Feature hierachy

A part model with multiple features are organized into a hierachy where the nodes represent instances of the elementary features defined above and the links represent the parent-child and sibling relations. A part model with instances of "POCKET-1", "STEP-1", and "HOLE-3" features are shown in Fig. 5 and the constructed feature hierachy is schematically shown in Fig. 6.

2.3 Recognition algorithm

It is assumed that the base of a feature is formed by an inner loop in a certain face, called a base face, and the axes of the global coordinate are coaxial to the approach directions. The 'base faces', where features start and end, are extracted from the part model along the approach direction, and they are hierachically related. Definition of the base faces and their hierachical relations are dipicted in Fig. 7.

Then, the hierachical structure for the base faces are constructed as shown in Fig. 8. Then, the type of feature - depression, protrusion or through hole - is decided by considering the geometric and topological information of the faces adjacent to the base face. After the feature type is decided, recognition rule is applied and feature data conforming the prearranged structure is produced. One example of recognition rule is given at Fig. 9. The rule contains the way how the feature "POCKET-1" is defined, *i.e.* identical profiles on different planes are linked by four perpendicular planes and cylindrical surfaces exist among each perpendicular plane.

The whole procedures of feature recognition explained up to now can be detaileded as below :

algorithm Feature_Recognition
  input *part model*
  select one of the orthgonal axes as an *approach direction*
  construct a *base faces hierarchy* of the *part model*
  construct a *node tree* of the *base faces hierarchy*
  find the *highest node level* in *node tree*
  for-each *node level* (from the *highest node level* until *node level* becomes 1)
    in *node tree*
    for-each *Topsibling* class in the seleted *node level*

for-each *base face* in *Topsibling* class
find its *parent base face*
find all the *connection faces* between *base face* and its *parent base face*
determine the feature type(depression, protrusion, through hole) according to the geometry of the *base face* and its *parent base face*
applying *recognition rules*
append the recognized feature to feature data structure
if (number of the *base faces* of the selected *Topsibling* class >= 2)
find all the *connection faces* between *base faces*
determine the feature type
applying *recognition rules*
append the recognized feature to feature data structure
end of algorithm

Feature finding is carried out axis by axis. In each axis, features are extracted form the highest node level of base face to the lowest one, which is reverse of machining sequence. In other word, the feature located in the deepest position is extracted first.

3. Feature Inspection

The resulting feature hierachy of a recognition process above is visualized by a grapher that displays the result in a tree structure. There is one-to-one correspondence between individual nodes in the tree and the features in the hierachy. Figure 10 shows the class definition of a grapher node.

The feature datastructure is read in to construct the grapher nodes. Each grapher node is associated with a box representing the node in the grapher window, labelled as EF-Hierarchy, that displays the tree structure as boxes. The parent and child relations between features are displayed as lines between boxes. Depending on the depth and width of the tree structure, the size of the boxes and the length of the lines between boxes are determined by the system.

```
class GrapherNode
{
  public:
      GrapherNode *next;          // next point
      GrapherNode *prev;          // previous point
      GrapherNode *parent;        // parent point
      GrapherNode *child;         // child point
      GrapherNode *brother; // brother point
      Feature  *Fpoint;           // Feature data base point
      int      label;             // node label
      int      path;              // does pass ?
      int      depth;             // depth of node
      int      xcoord;      // x coordinate to display
      int      ycoord;      // y coordinate to display
};
```

Fig. 10 Grapher node structure

When a box is picked, the grapher displays the associated parameters of the corresponding feature such as feature name, origin, direction, width, depth, and others in the information window labelled as EF-parameters. Furthemore, the component feature faces are highlighted in the viewport. Alternatively, a feature may be selected by picking a component feature face in the viewport.

4. Feature Deletion

4.1 Datastructure of a part model

A feature model consists of a feature hierachy and a part model. The feature hierachy is constructed by a feature recognition method from the part model. A part model is represented in B-rep using an extension of the half-edge datastructure(Mantyla, 1988) where a body consists of faces, a face consists of loops, and a loop consists of half-edge. A half-edge is an edge with a loop direction. In the part model, an edge can have more than two half-edges, called coedges. Here, an edge of a part model may have more than two coedges.

An individual feature is associated with the faces in the boundary model of a part, called shown faces. In addition, when a feature is created, a portion of a shown face may be hidden. The hidden protion of a feature is called a closure face(Pratt, 1988). Furthemore, the closure face is tagged with imposing features so that when an imposing feature is deleted, those imposed closure faces can be exposed. These closure faces with imposing features represent the feature interactions explicitly in the feature model.

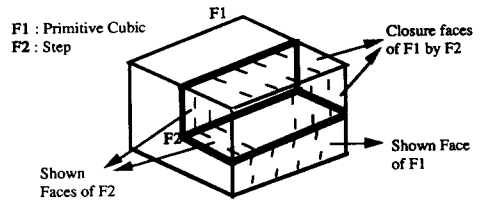


Fig. 11 Closure/shown faces

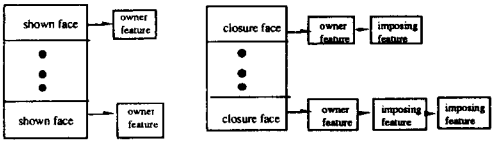


Fig. 12 Data structure for shown/closure faces

Figure 11 shows a part model with closure faces. In order to record the feature interactions, the closure faces are marked with the imposing features. The shown faces of F1 are partially hidden by F2. Those closure faces of F1 are closure faces with an imposing feature, F2. In general, there can be more than one imposing features.

The faces of the body structure is linked to the feature information as shown in Fig. 12. A shown face is associated with an owner feature if it is a part of the shown face list of the feature : a closure face is associated with an owner feature and the imposing features. These feature information is created and maintained by the feature modeling operations, feature insertion, recognition, and deletion. In the next section, we describe the use of closure faces in the feature deletion.

4.2 Adding closure faces to the recognized features

Once the recognition process finishes, each recognized feature is associated with only shown faces. Then, the closure faces are added with an imposing feature. A closure face is created using a face extension method[Dong 2]. Currently, the implementation is limited to cases where no new topology is created by the face extension.

Given a feature, collect the boundary edges of the feature. A boundary edge of a feature contains a coedge that belongs to a shown face of another feature. Group the coedges of the boundary edges

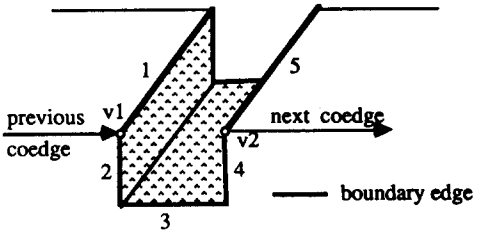


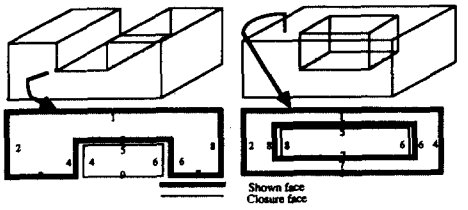
Fig. 13 Slot feature with boundary edges

with the same surface equation. For example, in Fig. 13, two groupings are shown : (1,5), (2, 3, 4). Construct a loop for each grouping. The loop construction requires adding new edges. For example, the vertices, $v1$ and $v2$, are connected by a new edge with the curve equation of the previous coedge of $v1$ if the previous coedge of $v1$ and the next coedge of $v2$ are of the same curve ; otherwise, a straight line is created for the new edge.

4.3 Deletion algorithm

Deletion of a feature involves exposing those closure face whose sole imposing feature is the feature being deleted and removing the feature from the imposing feature list of other closure faces by the feature. Then, the remaining faces are repaired with the exposed closure faces by concatenating the closure face with the shown faces. The concatenating process is explained using the external and internal features in Fig. 14.

Here, name the shown face of Fig. 14-(a) as A-loop and the closure face as B-loop where the edges are numbered. That is, A-loop is (1, 2, 3, 4, 5, 6, 7, 8) and B-loop is (5, 4, 9, 6). Select an edge in A-loop not in B-loop (assume edge 3) and store it in a queue. Using the next coedge, the next edge in the loop(edge 4)is selected and becomes the current edge. If this edge exists in the other loop(B-loop), the next edge in B-loop(edge 9) is chosen and becomes the current edge. Then, the current edge is stored in the queue(in this state, the contents of the queue is 3-9). The next edge of edge 9 is edge 6 and becomes the current edge. Edge 6 exists in the other loop(A-loop), then the current edge becomes edge 7. Now, the contents of the queue is 3-9-7. This process is continued until the current edge is the same as the first one in the



(a) External feature (Slot)
Shown face : (1 2 3 4 5 6 7 8)
Closure face : (5 4 9 6)
(b) Internal feature (Pocket)
Shown face : (1 2 3 5) (5 6 7 8)
Closure face : (5 8 7 6)

Fig. 14 Exposing dosure faces

queue. In this case, the resulting loop becomes 3-9-7-8-1-2.

If there is a remaining edge in A-loop, the above process is restarted. In Fig. 14(b), the shown face consist of two loops, the outer loop(1 2 3 4) and the inner loop(5 6 7 8) as A-loop and the closure face as B-loop, (5, 8, 7, 6). After going through the steps above with starting edge 1, the edges of the inner loop remains in A-loop. However, none of the edges remaining in A-loop is an edge in A-loop but not in B-loop. So, the process terminates only producing the loop,(1, 2, 3, 4).

After all the loops are produced with the exposed closure faces, the topology of the part must be modified with the new shown face. The overall deletion process is given below :

```
algorithm Delete_feature
  input feature
  for-each surface in feature
    if the surface has a closure face whose
        imposing feature is the input feature
    then concatenate the closure face with other
        loops din the surface
  modify topology
end of algorithm
```

The above algorithm does not support interacting features, however. Two additional steps are required : removing the deleted feature from the

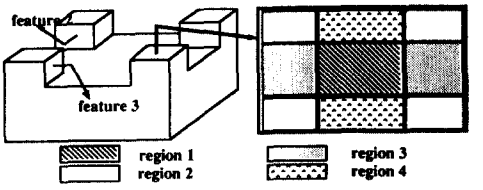


Fig. 15 Closure faces imposed by interacting features

imposing feature list of the closure faces and concatenating the adjacent closure faces with the same imposing feature list. In Fig. 15, there are two features, feature 2 and 3, that are crossing each other. The closure faces on the top surface of the block are labelled by regions : region 1 is a closure face whose imposing features are features 2 and 3, region 3 is imposed by feature 2, region 4 by feature 3, and region 2 shows the shown faces on the top surface of the base part. If feature 2 is removed, region 3 must be exposed. Then, region 1 is concatenated with region 4, and feature 3 remains as the only imposing feature of region 1. The modified algorithm for interacting feature deletion is shown in the following. Modified/ added steps are indicated with italic fonts.

```
algorithm Delete_interacting_feature
  input feature
  for-each feature in part
  for-each surface in feature
    if surface has a closure face whose imposing
        feature is the input feature
      if the closure face has only one
          imposing feature
        then concatenate the closure face with
            other shown faces in surface
        else
            remove the input feature from the
              imposing feature list
            concatenate the closure face with
              adjacent closure faces with the
              same imposing feature list
  modify topology
end of algorithm
```

5. Case Study

We have used a 'divided core plate' as an

example case. The part model of the divided core plate show at Fig. 16 was created by using a conventional solid modeling operations. Using a restore function, it loads the part model from the archive. The figures shows the feature model after the FR method has been applied to the part model. There are no recognized features from the approach directions, X, $-X$, and $-Y$. From $-Z$ approach direction, there are features related by the parent-child relationships. The feature hierachy is displayed in an EF-Hierarchy window by a grapher which displays the feature nodes in boxes and the parent-child relations as lines where the root node represents the base feature of the blank body. Using the grapher, one can inspect the parameters of the individual features in the hierarchy interactively.

Node 6 was picked and its boundary in the part model is high-lighted and its parameters are displayed in EF-Parameters window. Figure 17 shows holes deleted from the part model and the EF-Hierarchy.

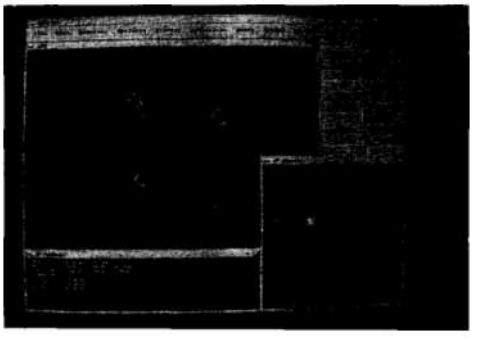


Fig. 16 Result of feature recognition

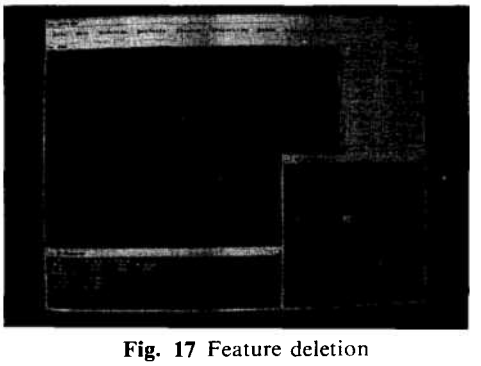


Fig. 17 Feature deletion

6. Conclusion

An interactive feature recognition method presented by Sakurai(Sakurai, 1990) introduces the interactivity by introducing the component feature faces in the part model. Alternatively, in this paper, we introduce interactivity after the recognition assuming that the result of the recognition is only a first approximation and the user must edit the final result.

Currently, almost every part of the system needs significant improvement, especially the closure face construction by the face extension method and the editing capability(only deletion is implemented). The face extension method can be improved by modeling the geometry and topology of the closure face interactively by introducing a geometric modeling functionality into the system. Additional editing capability of the result of FR process would increase the fondational range of the system to other domains : changing the parent and child relations between features, inserting a new feature into the hierachy, and more.

Acknowledgement

The research has been supported by the Ministry of Science and Technology under the Grants NO8791 and NO8783.

References

Dong, X., 1988, "Geometric Feature Extration for Computer-aided Process Planning," PhD thesis, Resselaer Polytechnic Institute.

Dong, X. and Wozny, M., 1991, "A Method for Generating Volumetric Features from Sufrace Features," Proceedings of Symposium on Solid Modeling and Foundations and CAD/CAM Applications, J. Rossignac and J. Turner(eds), Austin, Texas, pp. 185~194.

Kim, Y. S., 1991, "Form Feature Recognition by Convex Decomposition," ASME Computers in Engineering, Santa Clara, pp. 1~9.

Luby, S, Dixon, J. and Simmons, M., 1986, "Creating and Using a Feature Data Base,"

Computers in Mechanical Engineering, pp. 25 ~33.

Mantyal, M., 1988, *An Introduction to Solid Modeling,* Computer Science Press.

Park, M. W., Vosniakos, G. C. and Davies, B. J., 1990, "Automatic Feature Extraction from Wireframe Models of 2-1/2 *D* Prismatic Components," *Proc. 28th MTDR Conference,* Manchester, U. K., pp. 83~90.

Pratt, M. I., 1988, "Synthesis of an Optimal Approach to Form Feature Modeling," ASME Computers in Engineering 1988, V. A. Tipnis and E. M. Patton(eds), San Francisco, pp. 263~274.

Sakurai, H. and Gossard, D. C., 1990, "Recognition Shape Feature in Solid Models," IEEE Computer Graphics and Applications, pp. 22 ~32.

Shah, J. J. and Rogers, M. T., 1988, "Expert Form Feature Modeling Shell," Computer-aided Design, pp. 515~524.