Graph-based heuristics for recognition of machined features from a 3D solid model

S Joshi and T C Chang*

The internal representation of the solid modeller provides a description of parts which when used directly is useful for automation of the process planning function. So that the CAD model can be used to provide the information required for manufacturing, techniques to improve machine understanding of the part as required for manufacturing are needed. This paper presents the development of the concept attributed adjacency graph (AAG) for the recognition of machined features from a 3D boundary representation of a solid. Current implementation of the feature recogniser is limited to polyhedral features such as pockets, slots, steps, blind steps, blind slots, and polyhedral holes. Sample results that show the capabilities of the system are presented.

process planning, part description, attributed adjacency graph, feature recognition, polyhedral features

Computers are being used extensively to assist in the development of generative process planning systems. One such area is the development of a CAD interface for process planning[1]. The internal representation of the solid modeller provides a form of part description which when used directly is useful for automation of the process planning function. Part description in 3D CAD models is in a form of basic geometry and topology that is unsuitable for direct application to process planning. Process planning needs information in the form of features, which need to be extracted from this CAD model. Current generation of process planning systems require human input to interpret a part drawing or model and translate it into a set of machinable features suitable for process planning. The importance of feature recognition stems from the fact that each feature can be associated with knowledge about how the feature can be manufactured, and this information can be collectively used to generate a process plan. Hence feature extraction forms a major component of the CAD interface for process planning. Extraction of features can also be viewed as a fundamental aspect of the more general problem of machine understanding of designed parts. This understanding can also be extended to other applications besides manufacturing.

The problem of extracting geometric features involves recognising high level entities from the set of lower level entities in a geometric model. Depending on the application and the type of part representation scheme used, geometric entities such as faces, edges, vertices, primitive solids, etc. may be considered lower level entities. The definition of a feature is dependent on context. Features that are required for machining may differ considerably from those required for forging, even though they may be based on the same lower level entities. Features of a geometric model are relevant within a given context and highly dependent on what the model is used for. In this paper features are considered to be 'regions of a part having some manufacturing significance in the context of machining'. Examples of such features are Hole, Slot, Pocket, etc., each of which needs to be defined uniquely for the recognition procedure.

The development of a feature recogniser entails the following problems[2]:

- a representation scheme for the features that is suitable for automatic recognition
- unique definition of the feature based on the representation scheme
- inference procedure capable of recognition in a complete and consistent manner

REVIEW OF RELEVANT WORK

To increase the usefulness of solid models for manufacturing, enhanced machine understanding of the part is needed. The low level description has to be converted to the higher level conceptual information required for manufacturing. Several approaches have been used in the past. Grayer[3] used the boundary representation of the part, compared it to the initial workpiece, and divided the material to be machined into luminae to compute numerical control (NC) tool paths. Woo[4] focused on cavity recognition to transform volumetric designs of parts into descriptions for NC machining. Using a constructive solid geometry (CSG) tree, the cavities are recognised from the spatial relationships between the primitive volumes. These methods do not provide any semantic information that could be associated with the machined volumes and are based on local information. To improve global understanding of the part, Woo[5] proposed using the alternating sum of volumes. Here the raw material to be removed is decomposed into a set of volumes, using the alternating sum technique. No semantic information is associated with the volumes, and the volumes produced may be difficult to machine using a conventional machining process.

Another approach that was used was based on the syntactic pattern recognition methods. This approach was used by Jakubowski[6] to characterize overall part shapes. Choi[7,8] used a similar approach to recognise and classify various types of holes and generate process planning information based on it. Staley *et al.*[9] also used syntactic pattern recognition to classify and recognise various types of holes. Kyprianou[10] used a feature grammar to recognise features characterized by protrusions and depressions from a 3D

Department of Industrial and Management Systems Engineering, Pennsylvania State University, University Park, PA 16802, USA
*School of Industrial Engineering, Purdue University, West Lafayette, IN 47907, USA

 0010–4485/88/020058–09 $03.00

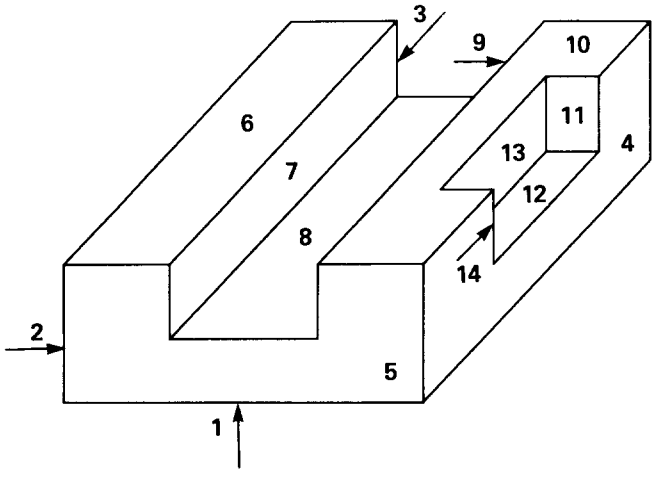
CAD database and used it to generate group technology codes for classification of parts.

Henderson[11] and Kung[12] used expert system rules and techniques to extract features from a 3D solid model using the internal boundary representation of the part. The features recognised are based on a table of existing features that need to be searched for in the part description. Lee and Fu[13] used a CSG to extract manufacturing features from a CSG representation automatically by unification of the CSG scheme. The features considered are fillet, round, chamfer, neck, and bore (features created by adding or subtracting cylindrical primitive solids).

This paper presents another approach to solving the feature recognition problem. The approach used is explained with respect to the representation scheme used, feature definitions based on the representation scheme, and the inference procedures used. Finally, some examples are shown to demonstrate the application of the technique.

PART REPRESENTATION SCHEME

Several part representation schemes exist for representing 3D solids internally in the computer[14]. A boundary representation (B-rep) scheme is used in our research, and the recognition algorithms are based on it. The topology of a part described by a boundary model can be represented in the form of a relational model, in which the primitive topological entities such as faces, edges, and vertices are represented. Associated with the faces, edges, and vertices are the corresponding geometrical definitions. A boundary model of an object can be defined as a triple B = (V, E, F) in which V = {set of vertices}, E = {set of edges}, and F = {set of faces}. The B-rep in its raw form provides the low level information that is not directly usable for feature recognition. Additional information regarding the type of face adjacencies and relationships between sets of faces needs to be explicitly available to assist in feature recognition. To facilitate the recognition process, the concept of an attributed adjacency graph built on the underlying B-rep is proposed. The low level B-rep is converted into features that form a higher level structural entity whose primitives are the elements of a B-rep.



ATTRIBUTED ADJACENCY GRAPH

The atrributed adjacency graph (AAG) can be defined as a graph G = (N, A, T) where N is the set of nodes, A is the set of arcs, and T is the set of attributes to arcs in A, such that

- for every face f in F, there exists a unique node n in N
- for every edge e in E, there exists a unique arc a in A, connecting the nodes n_i and n_j, corresponding to face f_i and face f_j, which share the common edge e
- every arc a in A, is assigned an attribute t, where $t = 0$, if the faces sharing the edge form a concave angle and $t = 1$, if the faces sharing the edge form a convex angle

Figure 1 shows an example of the AAG for a part. The AAG is represented in the computer in the form of a matrix.

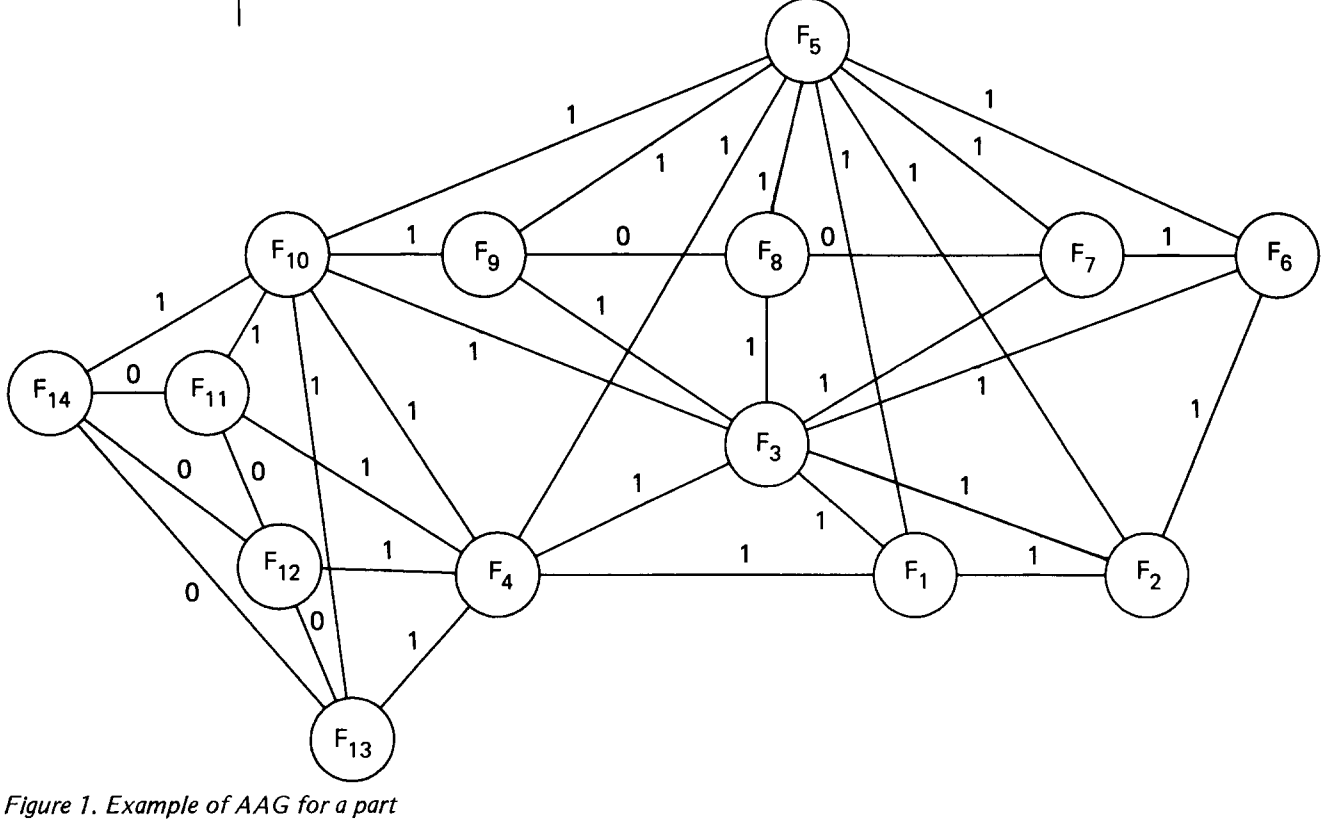


Figure 1. Example of AAG for a part

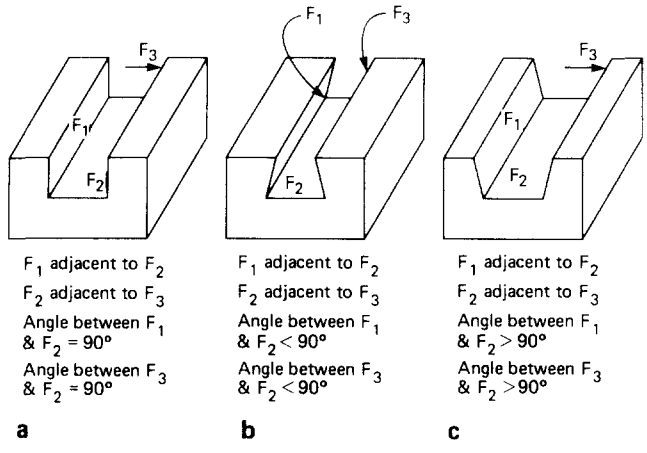


Figure 2. Different Slot types and their recognition rules

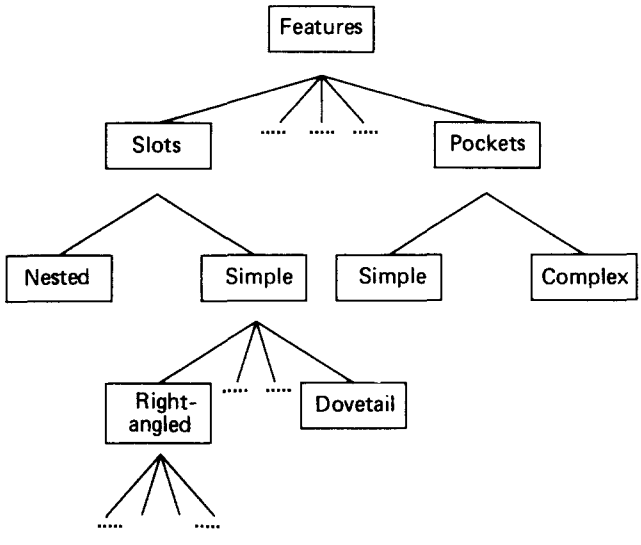


Figure 3. Hierarchical organization of features

Each node stores the pointer to the corresponding face information, thus providing a link to the B-rep if desired. Currently, the use of AAGs for extracting features is limited to polyhedral parts with polyhedral features. Cylindrical holes are recognised separately.

FEATURE DEFINITIONS

For the features to be recognised, they need to be precisely defined. Each instance of the feature must be identified, and incorrect instances should not be recognised. The definition of features involves determining the minimal set of necessary conditions that classify a feature uniquely.

Consider, for example, the different types of Slots shown in Figure 2. The slots can be defined based on the topological and geometric relationships of the faces that comprise the slot. The difference between the definitions of the three slots is the value of the angle formed by the faces with the base of the slot. The topological information is identical. A generic definition for the class of slots could be based on the following properties:

- F_1 is adjacent to F_2
- F_2 is adjacent to F_3
- F_1 forms a concave angle with F_2
- F_3 forms a concave angle with F_2

This approach of defining features based on the concept of generalized classification provides a framework for representing the features in an hierarchical manner. The specific information for further classification can be obtained from the geometric information. The degree of detailed classification desired controls the depth of the hierarchy tree (Figure 3). A major advantage of the hierarchical organization is the reduction in time required to recognise a feature. Another advantage arises from the notion of inheritance. As the instance of a particular subclass is also an instance of its superclass, the properties of the superclass can be inherited by the subclass without explicitly being repeated. This provides a more compact representation.

This paper deals with the problem of recognition of features based at the generic level, and the feature rules are based on recognising the most general instance of a feature type. Specific subclasses can be obtained through more specialized rules. The recognition rules are based on the properties of the AAG unique to each feature. Figure 4 shows some specific instances of generic feature types and their AAGs.

The properties of the AAG used to define a pocket are as follows:

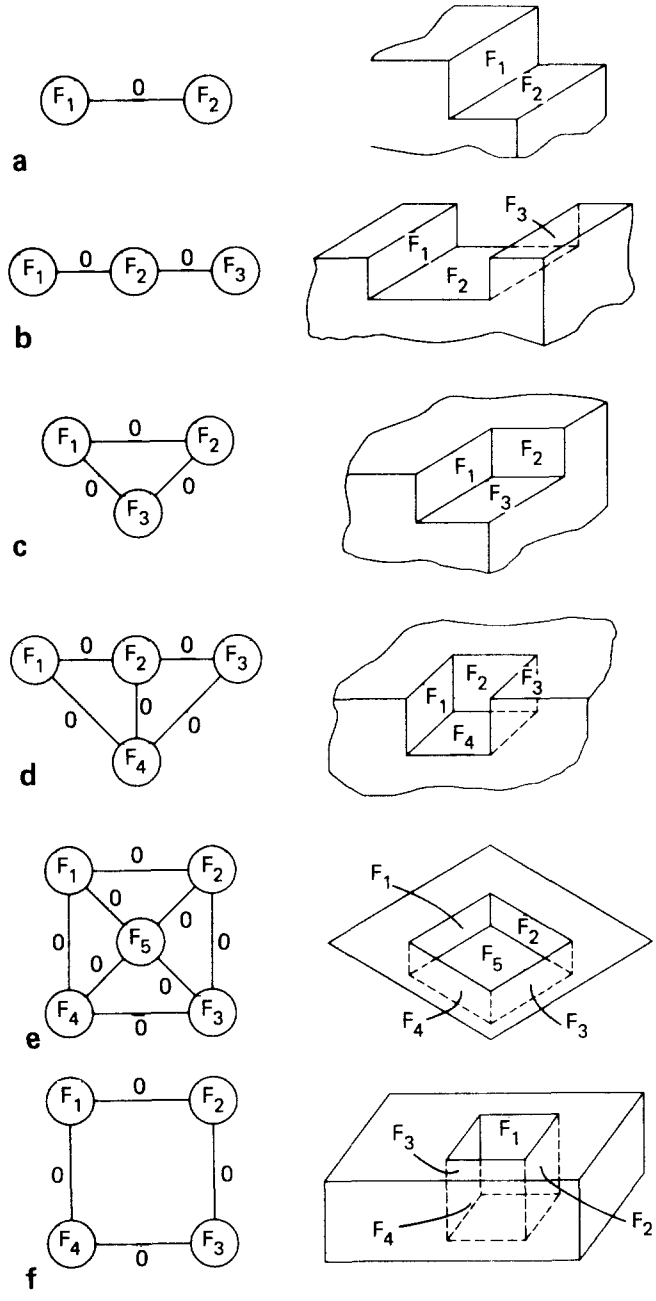


Figure 4. AAG of feature instances

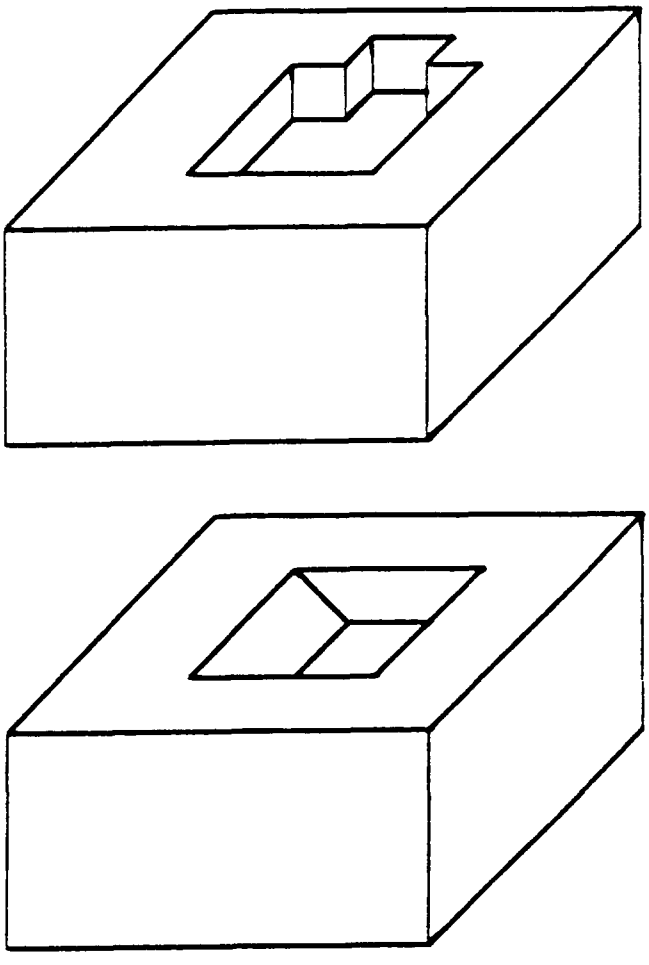


Figure 5. Different pockets recognised by rules for Pocket

- the graph is cyclic
- it has exactly one node *n* with number of incident 0 arcs equal to the total number of nodes - 1
- all other nodes have degree = 3
- the number of 0 arcs is greater than the number of 1 arcs (after deleting node *n*)

These properties of the Pocket are general enough to identify a wide range of pockets formed. Some examples of the types of pockets that can be recognised by the rule based on these properties are shown in Figure 5. Similarly, the other rules are general enough to recognise a wide range of feature instances. For example, the rule for Slot allows the recognition of nested as well as simple slots, since it has a varying number of faces.

Similar rules are written for each feature. The rules have to be unique for each feature. In the case of the simple Blind step, the AAG is not unique and can correspond to a triangular hole (hole formed by three faces); hence in this case an additional test needs to be performed to distinguish it from the triangular hole. All other features have unique properties of the AAG.

FEATURE RECOGNITION PROCEDURE

The features in the part are subgraphs of the complete AAG, and recognition of the features involves identification of the subgraphs that correspond to the features. The search for subgraphs in a larger graph is a subgraph isomorphism problem and is computationally exhaustive. A heuristic method is proposed to identify components of the graph that could form a feature. The heuristic is based on the following observation: a face that is adjacent to all its neighbouring faces with a convex angle does not form part of a feature. This is used as the basis to separate out the subgraphs that could correspond to features from the original graph. These subgraphs are analysed by the Recognizer to determine the feature types. The procedure to recognise the features is outlined below.

```
Procedure Recognize_Features
  create AAG
  delete nodes (and the incident arcs at the nodes) such that
    for each node deleted, all incident arcs have attribute '1'
  form components of the graph
  for each component
    Call Recognizer
    if recognized then
      return (feature_type, comprising_faces)
    else
      call Split_Edges
        form subcomponents of the graph
        for each subcomponent
          Call Recognizer
          if recognized then
            return (feature_type, comprising_faces)
          else
            call Split_Nodes
            form sub_components
            for each sub_subcomponent
              Call Recognizer
              if recognized then
                return (feature_type, comprising_faces)
              else
                Call Virtual_Pocket
                if recognized
                  return (feature_type, comprising_faces)
                else
                  return (Not_recognized)
                endif
              endif
            next sub_subcomponent
          endif
        next subcomponent
      endif
    next component
  Call Join_Features
End
```

After the initial AAG is created from the boundary model, subsequent operations such as delete node are performed on the graph. The effect of deleting a node from the graph only corresponds to eliminating a face from being considered as part of a feature. It does not correspond to deleting a face from the boundary model.

The algorithm is capable of recognising disjoint features and the types of intersecting features described in the next section. The Recognizer incorporates all the feature recognition rules. Each graph component is analysed to determine the feature type it corresponds to.

```
Procedure Recognizer (graph, feature_type)
  if (graph is acyclic) then
    check if linear or can be made linear by merging split faces
    if (linear) then
      check if Slot or Step
      if (recognized) then
        return (feature_type)
      else
        return (not_recognized)
      endif
    endif
```

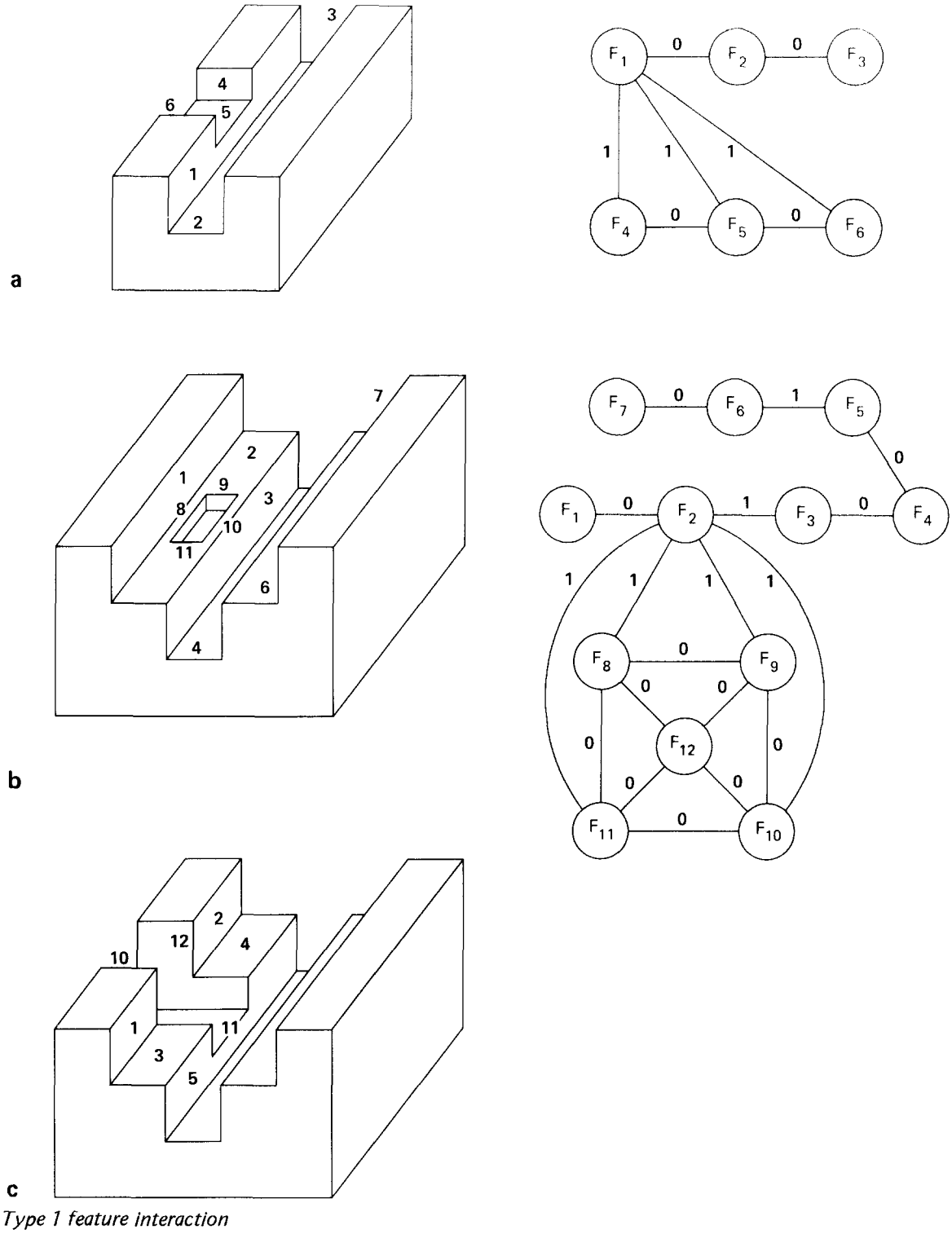


Figure 6. Type 1 feature interaction

```
    else (graph is cyclic) then
        check if Pocket, Blind Step, Blind Slot, or Hole
        if (recognized) then
            return (feature_type)
            else
            return (not_recognized)
        endif
    endif
End
```

INTERACTING FEATURES

Sometimes primitive features may not exist by themselves but may intersect with each other. Several interactions may be possible between the features, and all the possible interactions cannot be predefined using rules. Two types of feature interactions are considered.

Type 1

Features intersect such that they only have common edges between them. Examples of such cases are shown in Figure 6. Interaction of features may cause faces of one feature to be split up. In Figure 6(c) faces $\{F_1, F_2\}$ and $\{F_3, F_4\}$ (referred to as split faces) are split by a feature formed by faces $\{F_{10}, F_{11}, F_{12}\}$. Faces that cause the split are faces F_{10} and F_{12}. During the recognition process the split faces may have to be merged into a single node. The recognition procedure requires that the split face pairs satisfy two conditions

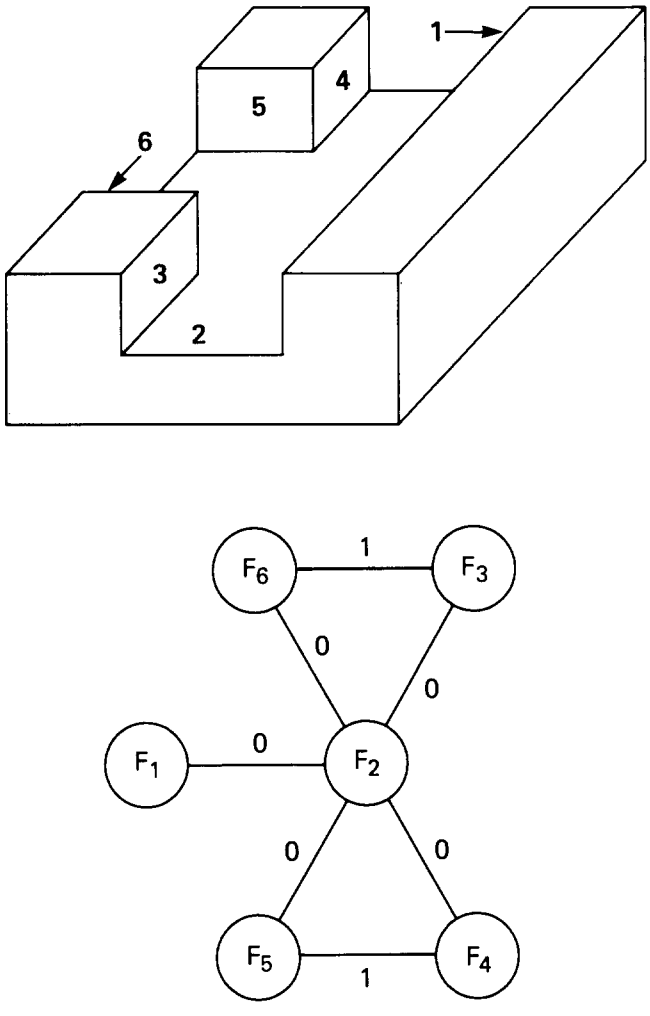


Figure 7. Type 2 feature interaction

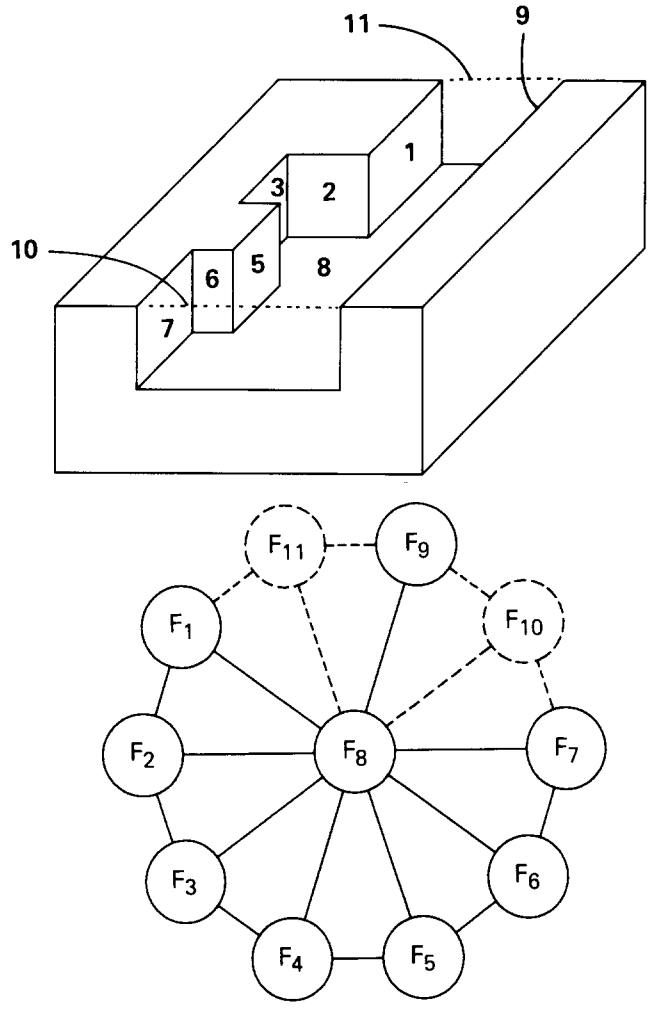


Figure 8. Virtual pocket

- the two faces have the same equation and
- the two faces have one face to which both are adjacent

In such cases the subgraph obtained after deleting the nodes does not correspond to a single feature. To recognise the constituent features, the subgraph needs to be decomposed into its constituent primitive elements. The procedure Split_Edges is used to separate the features. This procedure comprises

- identify all nodes with number of 1 incident arcs > 1
- delete all 1 arcs that emanate from such nodes
- form components of the graph
- add back 1 arcs deleted within each component

A problem with this heuristic procedure is that it may cause separation of features where no separation is desired. For example, in the part shown in Figure 6(b) the desired recognition would be the two features Slot $\{F_1, F_2, F_3, F_4, F_5, F_6, F_7\}$ and Pocket $\{F_8, F_9, F_{10}, F_{11}, F_{12}\}$. Applying the heuristic to the feature subgraph results in the formation of three components, recognised as follows

- Step $\{F_1, F_2\}$
- Slot $\{F_3, F_4, F_5, F_6, F_7\}$
- Pocket $\{F_8, F_9, F_{10}, F_{11}, F_{12}\}$

Such problems are corrected at the end by the procedure Join_Features.

Procedure Join_Features evaluates pairs of features to determine if they can be merged into a single feature, thereby refining the results of the feature recognition module. Only local considerations such as adjacencies and split face information is used to merge the features. Global relationships are not included. The process merges features by merging the list of faces that comprise the features, reconstructing the adjacencies for the merged list, and applies the Recognizer to determine if the merged features can be recognised as a single feature.

Type 2

Features intersect such that they share a common face, and interaction between the features splits a face of the feature. An example is shown in Figure 7, where the two slots S_1, S_2 share a common face F_2 and the split faces F_3, F_4 belong to feature S_1. This class of features can be separated by the procedure Split_Nodes. This procedure comprises the following steps.

- Find the nodes that correspond to the pair of split faces in the graph and assign to set A. This is done by checking the face equations from the B-rep information. If set A is empty: Stop.
- Find the nodes adjacent to A with 0 arcs and assign to set B. If set B is empty goto 8.
- Find the nodes that correspond to faces that cause the split and assign to set C. These nodes are adjacent to A

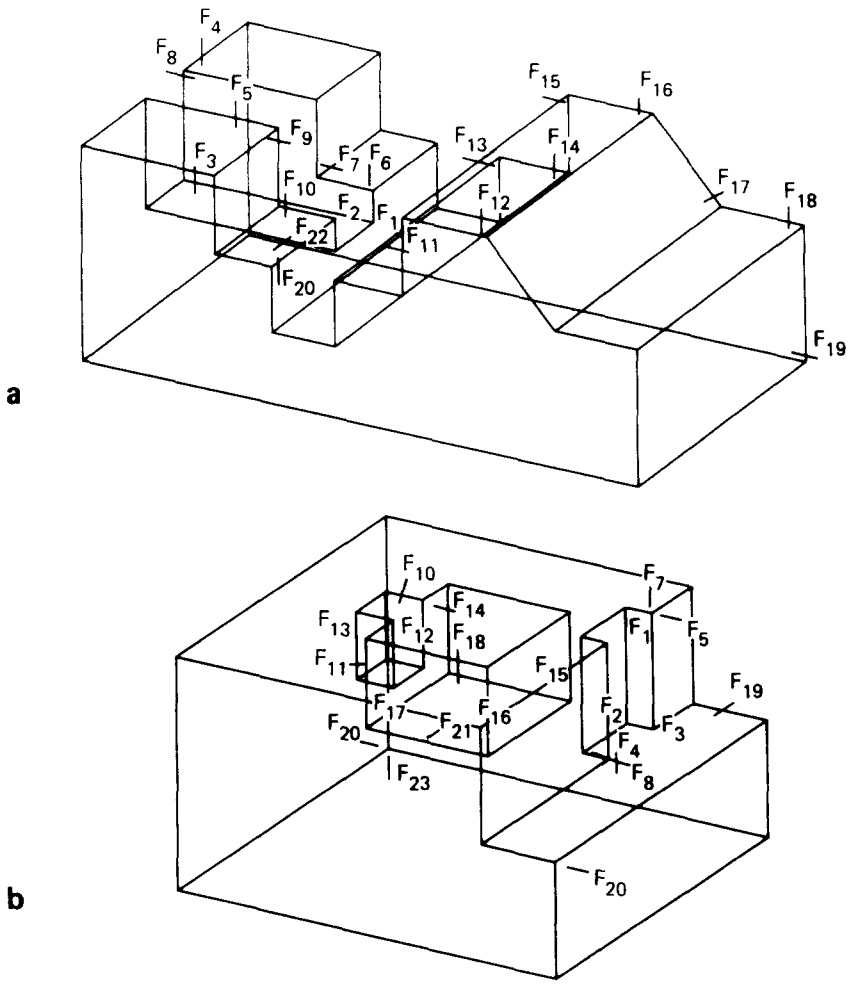


Figure 9. (a) Test part 1 (b) test part 2

with 1 arcs and to B with 0 arcs.

- The node n_k to be split is adjacent to (A ∪ C) with 0 arcs.
- Create set D as follows:

  D ← C
  for each element in D
  find node n_j adjacent to it such that n_j does not belong to (A ∪ n_k)
  assign n_j to set D
  next element in D.

- Connect the nodes in D to new node n'_k if the nodes are adjacent to n_k and delete 1 arcs that connect nodes in D to other nodes not in D.
- Combine the split face node pairs in set A into a single node and update graph.
- Reinitialize variables and repeat all steps.

This procedure is further explained with the help of the example shown in Figure 7. Faces F_3 and F_4 are identified as split faces and assigned to set A. Node F_2 is adjacent to A with 0 arcs and assigned to set B. At step 3, the nodes that cause the split are identified as F_5 and F_6 and assigned to set C. The node n_k to be split is F_2. Set D is created as $\{F_5, F_6\}$. The nodes in D are connected to F'_2 if they are adjacent to F_2, and the 1 arcs between F_6 and F_3 and between F_5 and F_4 are deleted in step 6. Nodes F_3 and F_4 are merged into one node. The components of the split graph are then recognised as slots $S_1 = F_1, F_2, F_3, F_4$ and $S_2 = F_5, F_6, F'_2$.

VIRTUAL POCKETS

The recognition procedure is further complicated by feature interactions that result in the loss of some information critical to the recognition of features. When several features interact in such a manner it may be difficult to make a unique classification, even for the human recognition process. The features in the part (Figure 8) can be recognised in several different ways, using different combinations of faces, all having face F_8 as the common face. Also, when machining such a combination of features it may be desirable to machine the features together as a pocket. Hence, such features are recognised as a virtual pocket, where faces F_{10} and F_{11} form the virtual faces. The virtual faces do not actually exist in the part but are only used to assist in classifying as virtual pockets some ambiguous features that can be machined as a pocket. The virtual faces are machined out when planning the cutting operations.

The recognition of virtual pockets is performed at the end after all features, both individual and intersecting, have been identified. The unidentified components are checked to see if they can be recognised as virtual pockets. Thus features that can be identified uniquely will not be identified as virtual pockets.

IMPLEMENTATION AND TEST RESULTS

The feature recogniser is interfaced with the Romulus solid modeller. Romulus stores the internal part description in

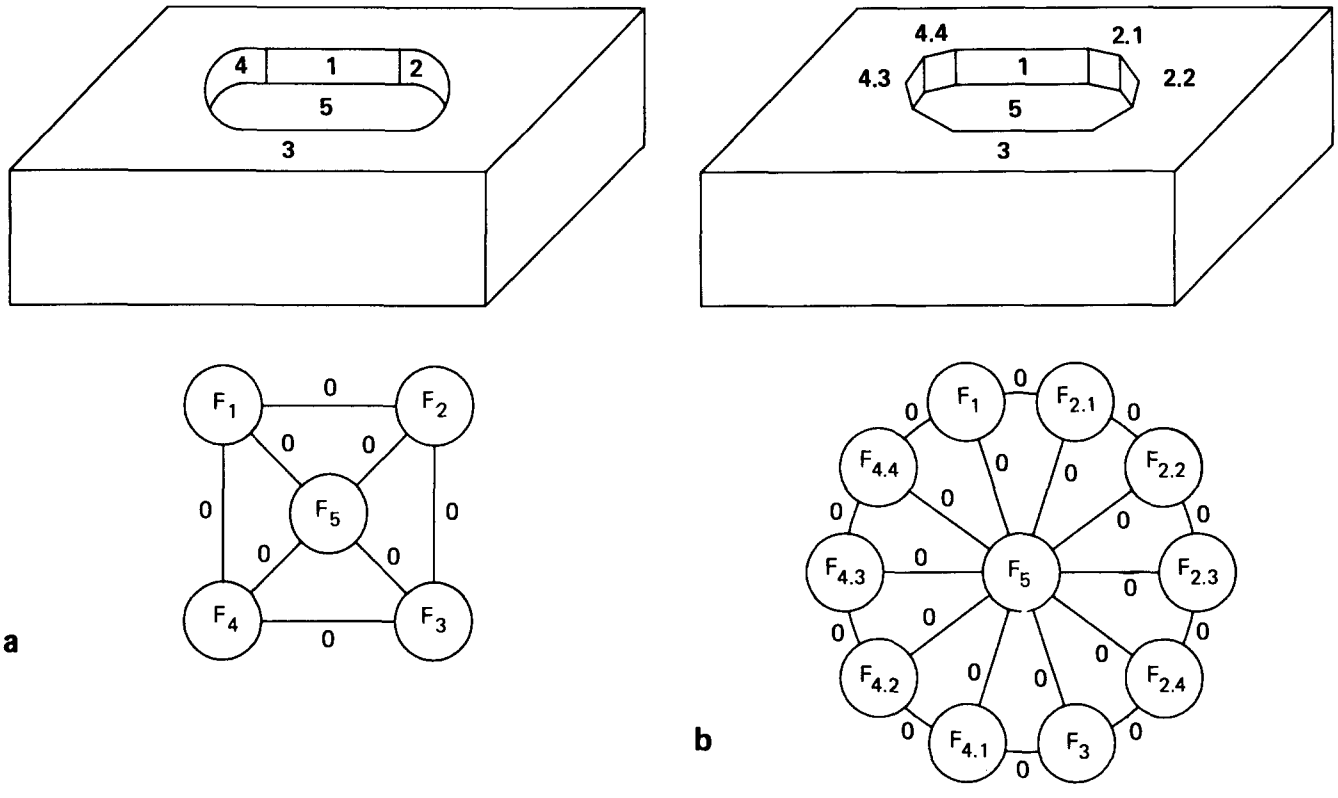


Figure 10. Feature with combination of cylindrical and planar faces

the form of a B-rep. Any other solid modeller capable of providing a B-rep can be used instead. The feature recognition module is coded in Fortran 77 programming language and implemented on a Sun 3/50 workstation.

Several parts with a varying number of features, with and without interactions, were tested. Two sample results are shown in Figure 9. Figure 9(a) shows a test part that comprises 22 faces, 56 edges, and 40 vertices. The part has three features, a Step $FT_1 = \{F_{18}, F_{17}\}$, a nested Slot $FT_2 = \{F_{15}, F_{14}, F_{13}, F_{12}, F_{10}, F_9, F_7, F_6\}$, and a Slot $FT_3 = \{F_3, F_2, F_1\}$. The features FT_1 and FT_2 exhibit a type 1 feature interaction. The execution time to recognise the features was 2.5 s. As shown in the results, the nested slot is recognised as a generic feature Slot. Inclined faces can also be handled, as shown in the recognition of the Step.

Figure 9(b) shows another sample part that has 21 faces, 51 edges, and 36 vertices. The part has four elements, a Pocket $FT_1 = \{F_{18}, F_{17}, F_{16}, F_{15}, F_{14}\}$, a Blind Slot $FT_2 = \{F_{13}, F_{12}, F_{11}, F_{10}\}$, a Step $FT_3 = \{F_5, F_4, F_8\}$, and another Blind Slot $FT_4 = \{F_3, F_2, F_1, F_4\}$. Features FT_1 and FT_2 exhibit a type 1 interaction, and features FT_3 and F_4 have a type 2 interaction. Execution time was 2.0 s.

COMMENTS AND CONCLUSIONS

The AAG-based heuristics discussed provide an effective and efficient technique for recognising machined features in a part. The expert system-based techniques use a backward chaining procedure, where execution is performed by invoking feature primitives one-by-one and performing an exhaustive search for the presence of the feature. Computational time grows exponentially with the number of features. The AAG approach, on the other hand, is a data-driven procedure and avoids the exhaustive search, hence resulting in drastic reduction in execution times for recognising the features. The range of features recognised and the types of feature interactions permitted far exceed the capabilities of the current existing systems.

The implementation of the AAG technique is currently limited to polyhedral features and parts, but the concept can be extended to features formed by combination of planar and cylindrical faces. Extensions of the approach to other types of faces used in solid modellers, such as cone, sphere, and torus, needs further research. Two approaches can be used to adapt the technique for features formed by combination of cylindrical and planar faces.

- Modify the concept of concave and convex adjacencies at edges common to planar and cylindrical faces. The part shown in Figure 10 can be recognised as a Pocket if the adjacencies between the cylindrical and planar faces are identified as concave.
- Approximate the cylindrical faces by planar facets or strips and use the same concept for concave and convex adjacencies for each strip. Figure 10(b) shows the effect of using this approach. The cylindrical faces F_2 and F_4 are approximated by four planar faces $F_{2,1}$, $F_{2,2}$, $F_{2,3}$, $F_{2,4}$ and $F_{4,1}$, $F_{4,2}$, $F_{4,3}$, $F_{4,4}$ respectively. The AAG created using this approximation will identify the feature as a Pocket.

Further research needs to be conducted to incorporate compound features such as T-slots and other types of intersecting features not discussed here. The feature recogniser fails to recognise all types of holes and is limited to all convex holes and some nonconvex holes. A hole is convex if all cross-sections of the hole are convex. Nonconvex holes that have a hole face adjacent to other hole faces with only convex angles cannot be recognised. The feature recogniser is useful in a variety of manufacturing applications. One such area is automated process planning,

where the feature information can be used to determine tool approach directions for machining, develop machining sequences based on feature relationships[15], and generation of tool paths for machining. Other applications of feature extraction include use in design for manufacturability analysis, parts classification, and general machine understanding of designed parts.

ACKNOWLEDGEMENT

The work was partially supported by the NSF Engineering Research Center for Intelligent Manufacturing Systems at Purdue University and a NSF Presidential Young Investigator award to Dr Chang with matching funds from Rockwell International and Xerox Corp.

REFERENCES

1 **Joshi, S and Chang, T C** 'CAD interface for automated process planning' in *Proc. 19th Int. CIRP Seminar on Manufacturing Systems* Pennsylvania State University (1987) pp 39–45

2 **Hirschtick, J K and Gossard, D C** 'Geometric reasoning for design advisory systems' in *Proc. ASME International Computers in Engineering Conference* (Chicago, July 20–24, 1986) Vol 1 (1986) pp 263–70

3 **Grayer, A R** 'The automatic production of machined components starting from a stored geometric description' in **McPherson, D (ed)** *Advances in computer-aided manufacturing* North-Holland Publishing Co. (1977) pp 137–151

4 **Woo, T C** 'Computer aided recognition of volumetric designs' in **McPherson, D (ed)** *Advances in computer-aided manufacturing* North-Holland Publishing Co. (1977) pp 121–135

5 **Woo, T C** 'Interfacing solid modeling to CAD and CAM: data structures and algorithms for decomposing a solid' *Technical Report 83-6* Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, USA (1983)

6 **Jakubowski, R** 'Syntactic characterization of machine-parts shapes' *Cybernetics and Systems Int. J.* Vol 13 (1982) pp 1–24

7 **Choi, B K** 'CAD/CAM compatible tool oriented process planning for machining centers' *PhD Thesis* Purdue University, West Lafayette, IN, USA (1982)

8 **Choi, B K, Barash, M M and Anderson, D C** 'Automatic recognition of machined surfaces from a 3-D solid model' *Comput.-Aided Des.* Vol 16 No 2 (March 1984) pp 81–86

9 **Staley, S M, Henderson, M R and Anderson, D C** 'Using syntactic pattern recognition to extract feature information from a solid geometric data base' *Comput. Mech. Eng.* Vol 2 No 2 (1983) pp 61–66

10 **Kyprianou, L K** 'Shape classification in computer aided design' *PhD Thesis* Christ College, University of Cambridge, Cambridge, UK (1983)

11 **Henderson, M R** 'Extraction of feature information from three dimensional CAD data' *PhD Thesis* Purdue University, West Lafayette, IN, USA (1984)

12 **Kung, H** 'An investigation into development of process plans from solid geometric modeling representation' *PhD Thesis* Oklahoma State University, OK, USA (1984)

13 **Lee, Y C and Fu, K S** 'Machine understanding of CSG: extraction and unification of manufacturing features' *IEEE Comput. Graph. Appl.* (1987) pp 20–32

14 **Requicha, A A G** 'Representations for rigid solids: theory, methods and systems' *Comput. Surveys* Vol 12 No 4 (December 1980) pp 437–464

15 **Joshi, S** 'CAD interface for automated process planning' *PhD Thesis* Purdue University, West Lafayette, IN, USA (1987)