Pergamon

PII: S0360-8352(96)00295-1

Computers ind. Engng Vol. 32, No. 2, pp. 383–397, 1997

0360-8352/97 $17.00 + 0.00

A NEW FEATURE-BASED DESIGN SYSTEM WITH DYNAMIC EDITING

DER-BAAU PERNG and CHAO-FAN CHANG

Institute of Industrial Engineering, National Chiao Tung University, Hsin-chu, Taiwan 30010, Republic of China

(Received 28 August 1996)

Abstract—This paper proposes a feature-based design system for prismatic parts in which predefined 3D features, instead of low-level 2D lines and points, are used as entities for representation and manipulation. Two special sets of functions, a new construction function and a dynamic editing function, enable designers to freely construct and modify parts. The parts are saved as feature-based files in CSG form, while a B-rep form is also created. The feature-based file allows the proposed 3D part-design system to be easily integrated into CAPP/CAM systems without complex feature-recognition and -extraction processes. Object-oriented concepts are used in implementing the proposed system, and examples are given to show its feasibility and effectiveness. © 1997 Elsevier Science Ltd. All rights reserved

1. INTRODUCTION

Rapid progress in computer technologies has made computer-integrated manufacturing (CIM) increasingly more feasible. Integration of computer-aided design (CAD) and computer-aided process planning/computer-aided manufacturing (CAPP/CAM) systems has become more important within the CIM environment. The key issue is to take part-manufacturing processes and requirements into account earlier in the design phase to produce a tightly coupled integration environment.

Over the last decade, using features for part modelling has come to be thought of as a key to integrating design and manufacturing [1–8]. The methodologies proposed for creating high-level models have all taken one of three approaches: interactive identification [9, 10], automatic recognition of features [11–18], or design-by-feature (feature-based design) [19–31]. Details of these approaches can be found in [32].

To eliminate the complexities of feature-recognition and -extraction, and provide designers with a unique, consistent set of features for 3D part construction, we have used the design-by-feature approach in constructing 3D object models. Most surveyed feature-based design systems focus on assisting designers to construct part-feature models. For example, in [19, 31], a volume-based approach for part construction in an interactive graphical environment was used. Our research, however, stresses not only the construction aspects, but also the issue of enabling designers to modify feature models of existing parts conveniently.

We therefore combine a dynamic editing function with dynamic picking and positioning. Designers can now get real-time feature-variation visualizations while creating or modifying a part. A selected feature or a modification can be dragged dynamically to a desired position in a user-friendly design environment. Designed parts are saved as feature-based files in CSG (Constructive Solid Geometry) form; and a B-rep (Boundary REPresentation) form is also created. Using the feature-based file, designers can easily integrate our proposed 3D part-design system with CAPP/CAM systems, while avoiding the complex processes of feature-recognition and -extraction. The system schematic structure is shown in Fig. 1.

The next section briefly reviews the feature-based design system literature. In Section 3, we describe the object-oriented concepts and propose a hybrid B-rep/CSG representation scheme for feature-based part description. Details of the feature-based part design of graphical-user-interface design environment and the dynamic editing function are described in Sections 4 and 5. Simulation

examples illustrating use of the proposed system are given in Section 6. Finally, in Section 7, we summarize conclusions and suggestions.

2. LITERATURE REVIEW OF FEATURE-BASED DESIGN SYSTEMS

Table 1 is a review of papers that discuss feature-based design systems. Their characteristics can be assigned to two major categories: capacity/property, and design environment. Capacity/property refers to the basic construction and editing functions a system provides; design environment refers to additional functions provided for ease of design and edit. Below, we comment on a certain element in Table 1.

Feature-based design systems generally handle rotational parts more easily [33], and only 2D geometric description is needed, such as in [22, 24]. Although Shah and Rogers [29, 30] proposed a 3D rotational-part design system, the means for extending it to 3D prismatic-part design are not clear.

Representation schemes required for 3D prismatic parts are usually more complex than those required for rotational parts. A hybrid B-rep/CSG representation scheme is a better choice [28, 34]. Some studies have been done on 3D prismatic-part design [19, 27, 28, 31], as shown in Table 1. However, few part-design systems provide convenient design environments; [19, 31] are exceptions. None of these systems provide all dynamic positioning/editing, feature-picking, graphic menu, or interactive dialog-box input functions.

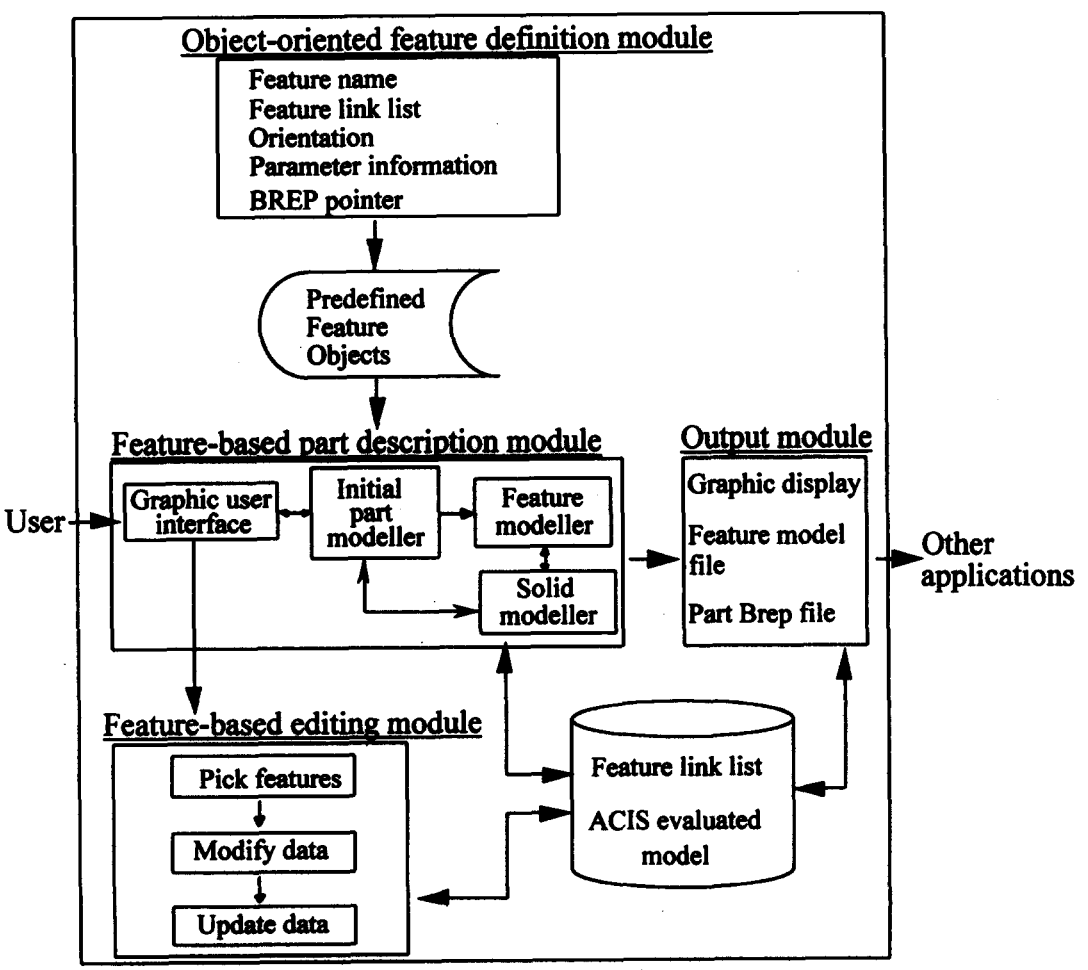

Fig. 1. Feature-based part design system.

Table 1. Review of feature-based design systems

System name	QTC [19, 31]	I-DEAS [20]	I-DEAS [20]	GFM [22]
Capability/properties				
Part	Prismatic	Casting	Sheet metal	Rotational
Geometric capability	3D	3D	3D	2D
Solid modeller	TWIN	I-DEAS	I-DEAS	No
Programming language	C	Lisp	C	C
Output	Geometric data Tolerance	Unclear	Solid modeller command Mid-surface model Features file	Manufacturable features Feature connectivity
Modelling	Volume-based	Surface-based	Surface-based	Unclear
Editing				
Delete	Unclear	Not mentioned	Not mentioned	Unclear
Modify	Yes	Not mentioned	Not mentioned	Not mentioned
Move	Unclear	Not mentioned	Not mentioned	Not mentioned
Undo/redo	Unclear	Not mentioned	Not mentioned	Unclear
Feature validation				
Geometrical feasibility	Not mentioned	Yes	Yes	Yes
Functional feasibility	Not mentioned	Yes	Yes	Unclear
Design environment				
User interface	Multi-window, highly graphical, interactive environment	Unclear	Unclear	Interactive graphical menu
Positioning	Keyboard Mouse	Keyboard	Keyboard	Unclear
Quick display	Not mentioned	Not mentioned	Not mentioned	Not mentioned
Querying	Not mentioned	Yes	Not mentioned	Not mentioned
Picking	Yes	Not mentioned	Not mentioned	Yes
Dragging	Not mentioned	Not mentioned	Not mentioned	Not mentioned

System name	TURBO-MODEL [24]	FEDDS [27]	None [28]	ASU [29–30]
Capability/properties				
Part	Rotational	Prismatic	Prismatic	Rotational
Geometric capability	2D	3D Planar face only	3D	3D
Solid modeller	No	MIT CADLAB	ROMULUS	SOLIDESIGN
Programming language	Pascal	Modula-2	Unclear	C
Output	Geometrical data Technological data Global data		Unclear	Feature relationship graph
Modelling	Unclear	Volume-based	Volume-based	Volume-based
Editing				
Delete	Yes	Yes	Yes	Yes
Modify	Yes	Yes	Unclear	Yes
Move	Unclear	Unclear	Not mentioned	Unclear
Undo/redo	Yes	Not mentioned	Not mentioned	Not mentioned
Feature validation				
Geometrical feasibility	Yes	Yes	Unclear	Yes
Functional feasibility	Yes	Not mentioned	Unclear	Not mentioned
Design environment				
User interface	Text menu Drop-down screen tablet	Non-interactive	Non-interactive	Non-interactive
Positioning	Unclear	Mouse	Unclear	Keyboard
Quick display	Not mentioned	Yes	Not mentioned	Yes
Querying	Not mentioned	Not mentioned	Not mentioned	Yes
Pick	Not mentioned	Yes	Not mentioned	Unclear
Dragging	Not mentioned	Not mentioned	Not mentioned	Not mentioned

The issues listed below have not been clearly dealt with in these feature-based design systems:

- 3D feature-based picking functions were not provided;
- feature dynamic-positioning and editing functions were not described;
- editing capabilities for "feature interactions" [28] were not clearly described.

The previous feature-based design systems describe only editing functions and interactive user-interfaces for constructing and modifying 3D parts. Therefore, our aim is to develop new functions for each construction and editing of 3D prismatic parts through a user-friendly graphical interface.

3. OBJECT-ORIENTED CONCEPTS AND FEATURE-BASED DESIGN

For easy designing of the features, we propose a feature-definition module using object-oriented concepts. The module provides a set of prismatic-part features as basic part-construction elements

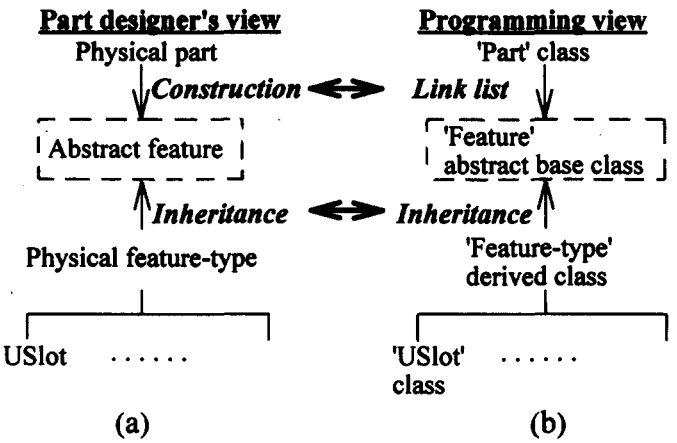

Fig. 2. The mapping relationships between "object" and "class" using object-oriented concepts.

using the object-oriented concepts. We also propose a hybrid B-rep/CSG representation scheme in this module for feature-based part description. The object-oriented concepts, as applied for part and feature descriptions, are outlined in Fig. 2 and explained later.

3.1. Object-oriented part/feature descriptions from the designer's view

In this section, we will explain the construction relationships between physical parts and abstract features as well as the "inheritance" relationships between abstract features and physical feature-types, as shown in Fig. 2(a), using the object-oriented concepts. Part designers usually think of features as volumes to be removed by machining operations, and describe a feature-based part as an object by subtracting features from the raw material. Thus, we represent features as volumetric objects, that are somewhat different from CAM-I's constructive features [35], and the physical parts can then be constructed by the subtracted volumes of abstract features. The advantages of using volume-based models can be found in [36].

Machining parts are generally divided into two classes, rotational and prismatic, according to their external shapes. From the manufacturing viewpoint, we focus on the class of prismatic parts that have depressive features including: arches, fillets, holes, T-slots, U-slots, V-slots, and wedges [13]. All feature orientations are assumed to be parallel or perpendicular to the coordinate axes. Such properties of abstract features are inherited by the previous seven physical feature types. The structure diagram of the part–feature relationship can be represented as in Fig. 3.

Figure 4 shows an example of a prismatic part described by features. Each feature is treated as an object in the part designer's view. We represent the sample prismatic part in Fig. 4 as the physical part described in Fig. 2(a), and it consists of eight features in abstract form. The associated physical feature-type objects are Arch1, Arch2, Arch3, Arch4, Hole1, Hole2, USlot1 and USlot2.

3.2. Object-oriented part/feature descriptions from the programming view

Taking the object-oriented concepts of "inheritance" and "polymorphism", we adopt the C++ programming language [37] to implement the objects relationships in the proposed system.

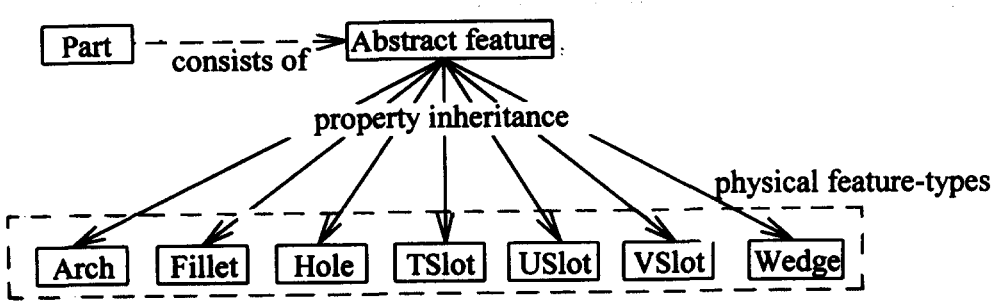

Fig. 3. The structure diagram of the part–feature relationship.

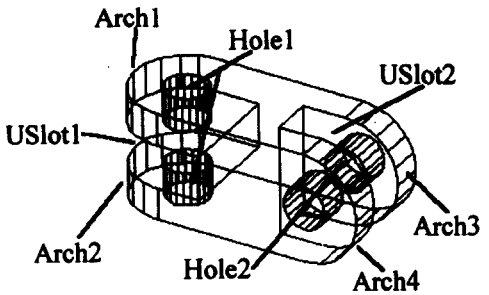

Fig. 4. Example of a prismatic part described by the features of arch, hole, and Uslot.

Inheritance enables the extension of class hierarchies from physical part, to abstract feature, and to physical feature-type. C++ uses the mechanism of "class derivation" to implement the inheritance property. A new "derived class" can be built upon the basis of some existing "base class". That is, a new feature-type class can be created by refining the members of the abstract base class of features. On the other hand, polymorphism is applied to the object-oriented program design. It means one operation or function can be implemented in different ways for different classes. In C++, the "dynamic biding" property of polymorphism is implemented through the mechanism of "virtual function". Using virtual functions, the implementation of the proposed system can be simplified [37].

Thus, the C++ language strongly supports implementation of the construction relationship between physical parts and abstract features, and the inheritance relationship between abstract features and physical feature-types. For example, the construction relationship can be implemented by defining the physical part as a class named "Part", which is composed of a linked list of abstract features as shown in Fig. 2(b). On the other hand, the abstract base class is defined as "Feature", as in Fig. 5(a). A derived physical feature-type of the sample "USlot" is given in Fig. 5(b) to explain the inheritance-relationship implementation. The base class is implemented as an object composed of common attributes and virtual functions, while the derived physical feature-type of the sample "USlot" is implemented as an object composed of private attributes and virtual functions. The derived class will inherit the common attributes of the base class. Other types of physical features can be similarly implemented. And the operation of inheriting the property of the abstract base class "Feature" by the sample feature-type "USlot" and virtual functions are explained as follows.

class Feature /*an abstract base class which contains common attributes of physical-feature-type object*/	class USlot: public Feature /*a derived physical-feature-type class which inherits common attributes of the base class 'Feature'*/
{	{
Common attribute	Private attribute
feature name reference point orientation Boolean operator feature B-rep BODY pointer part B-rep BODY pointer Feature link list pointer	Width Height Depth order
Virtual function	Virtual function
input feature parameters generate B-rep data output feature file	input feature parameters generate B-rep data output feature file
};	};

(a)The definition of abstract base class 'Feature' (b)The definition of derived class 'USlot'

Fig. 5. Example definitions for the abstract base class "Feature" and the derived class "Uslot".

(1) C++ keywords

In Fig. 5(a), the keyword "class" is used to define a new data type named "Feature". A class is a collection of static properties (common attributes) and dynamic behaviors (virtual functions) of an object. Another keyword, "public", appearing in Fig. 5(b) indicates that the "USlot" is a derived class and that it inherits the common attributes of the base class "Feature". That is, the data of class "Feature" can be inherently shared by the derived class "USlot".

(2) Common attributes [see Fig. 5(a)]

Analysing the features, we find that each physical feature-type has several common attributes, such as feature-type name, reference point, orientation, and associative Boolean operator. These common attributes are collected into an abstract base class called "Feature" in our system. Such implementation facilitates the manipulation of feature-attribute inheritance [37].

(3) Private attributes [see Fig. 5(b)]

Besides the common attributes, each physical feature-type may have its own private attributes. For example, the "USlot" feature has the private attributes of width, height, and depth; while the "Hole" feature has those of diameter and depth. The common attributes mentioned above can be inherited and re-used by different feature-type objects, but the private attributes can only be accessed by the feature-type object that defines these private attributes. In addition to easy maintenance, extension of the feature library can easily be accomplished by adding additional private attributes of new feature-types.

(4) Virtual functions

A virtual function has the advantage of dynamic-binding [37]. When some virtual function of a defined feature class is called at run-time, the corresponding function to be executed next will depend on the object's actual bounded feature-type. The virtual function defined in the physical feature-type class includes input feature parameters, output feature parameters, 3D B-rep data generation functions, and so on. Using virtual functions, simplifies feature-description program design and helps make the program more versatile [37].

In this paper, the geometric representation of part description is in a hybrid B-rep/CSG form. The B-rep data, including parts and features, is represented in ACIS format [38] while the CSG parameters include associative common/private attributes for each feature described. Using the sample part in Fig. 4, Fig. 6 shows its corresponding hybrid B-rep/CSG representation scheme in the programming view. The physical parts and feature-types thus implemented using object-oriented programming concepts pave the way for design of an easy-to-use, maintainable environment for the proposed system described in the next sections.

4. FEATURE-BASED PART DESCRIPTION

To provide designers with a friendly interface, the set of feature icons and associated dialog boxes are developed in the feature-based part description module. Instead of low-level 2D line/point entities, this module uses the 3D features defined in Section 3 as basic design entities for representation and manipulation. Both B-rep and CSG data are saved after finishing the design task. The characteristics of this module are described below.

4.1. Graphic–user interface

A principle characteristic of our design system is its graphic–user-interface design environment. Each type of feature has a corresponding graphic icon (Fig. 7). The icon menu shows each graphic entity along with its name, key parameters, and a reference point. The raw material and different features have their own corresponding dialog boxes. When a desired feature is selected, an associated dialog box will pop up (Fig. 8). The designer can then specify dimensions in the dialog box needed to design or modify a feature. Besides data input, the dialog box also provides other related information on features.

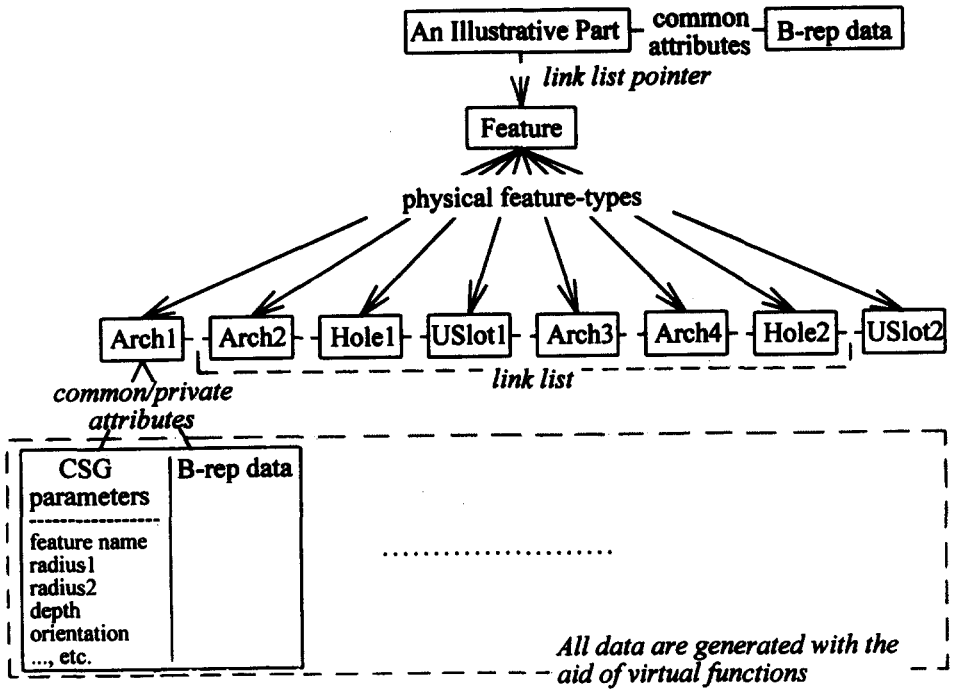

Fig. 6. A hybrid CSG/B-rep representation scheme for the sample part in Fig. 4.

4.2. *Feature-based part description*

With 3D-feature entities, the designer can describe a part by specifying the geometric dimensions of raw material and features as well as geometric Boolean operations. For example, to design the part illustrated in Fig. 9(c), the designer simply has to specify the parametric raw material values as given in Fig. 9(a), and the parametric "USlot" values including Reference Point, H, W, D, and Orientation in Fig. 9(b). The desired result in Fig. 9(c) can then be obtained by specifying the geometric Boolean operation of "subtracting" Fig. 9(b) from Fig. 9(a). Also included is checking for cases of infeasible/inconsistent input data, and appropriate prompt messages to the designer.

The detailed procedures of the feature-based part description module are shown in Fig. 10. The designer only has to point out where on the raw material or on the semi-finished product the features are to be specified. This task is accomplished by the functions described in the next section.

4.3. *Dynamic positioning*

When pointing out where on the raw material or on the semi-finished product the features are to be placed, the designer can use one of two approaches the system provides: (a) the designer is asked to input a precise operation point, at which the specified feature is to be subtracted from the raw material or from the semi-finished product; (b) the designer can dynamically specify where, on the raw material or semi-finished product, to cut out the features using a mouse. Using the first method, the designer can quickly finish the input task when the actual data is available in advance. Or, using the second method, the designer can use a mouse to dynamically drag 3D features as entities to desired positions.

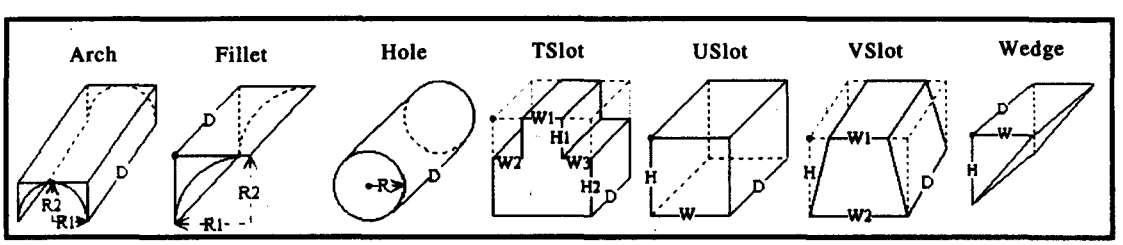

Fig. 7. Icon menu: a property of the design environment of the proposed system.

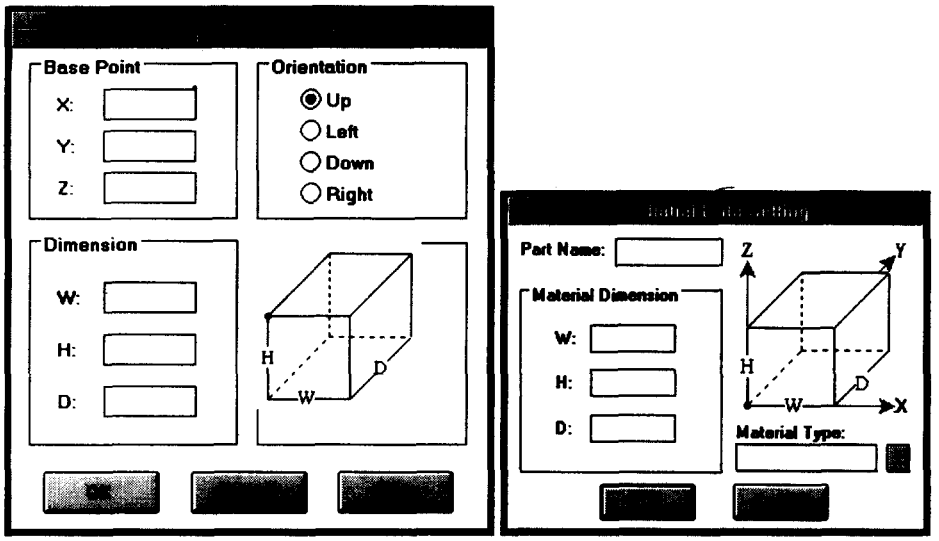

Fig. 8. Examples of the dialog box. (a) Dialog box for a "USlot"; (b) dialog box for stock-type raw material.

The critical issue involved in dynamic-positioning is transforming the 2D coordinates picked by the mouse on the monitor into 3D coordinates. This is accomplished by applying the concepts of "Action Face" and "Working Depth" to fix one 3D coordinate. The "Action Face" [see Fig. 9(a)] may be a boundary surface of the raw material/semi-finished product or a plane cut through the raw material/semi-finished product. A valid operation point can be dynamically specified on the action face. The default action face is one of the outermost boundary faces of the raw material/semi-finished product. The action face, here, is assumed to be parallel/perpendicular to the coordinate axes of the world coordinate system. On the other hand, the "Working Depth" is the depth between the current action face and the default one (see Fig. 11). Using the concepts of "Action Face" and "Working Depth", designers can specify any position inside the part and can directly place their design features at the desired positions on the object without inputting further coordinates.

When the feature is dynamically positioned at the desired location, the ACIS solid modelling system is called to update the 3D B-rep data of the designed part and display the design result. If the design result is not satisfactory, the designer can undo it or redo the design process. A linked list of features is updated following the modification. The geometric operations upon the raw material/semi-finished product and features performed by the ACIS solid modelling system can be the Boolean operations of union, subtraction, and intersection. More about the feature-based editing functions is described in Section 5.

The proposed system, though useful, still has the following feature-validation design constraints:
(1) Point constraints:

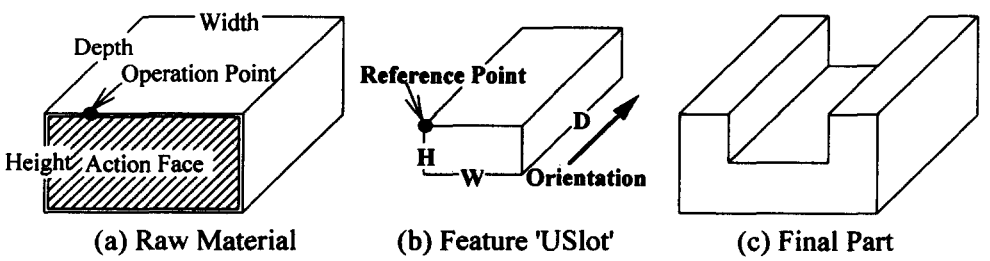

Fig. 9. Example of designing the feature-based part in (c) by specifying the parametric raw material values in (a); the associative values of the "USlot" feature in (b); and the geometric operation of subtracting the "USlot" from the raw material.

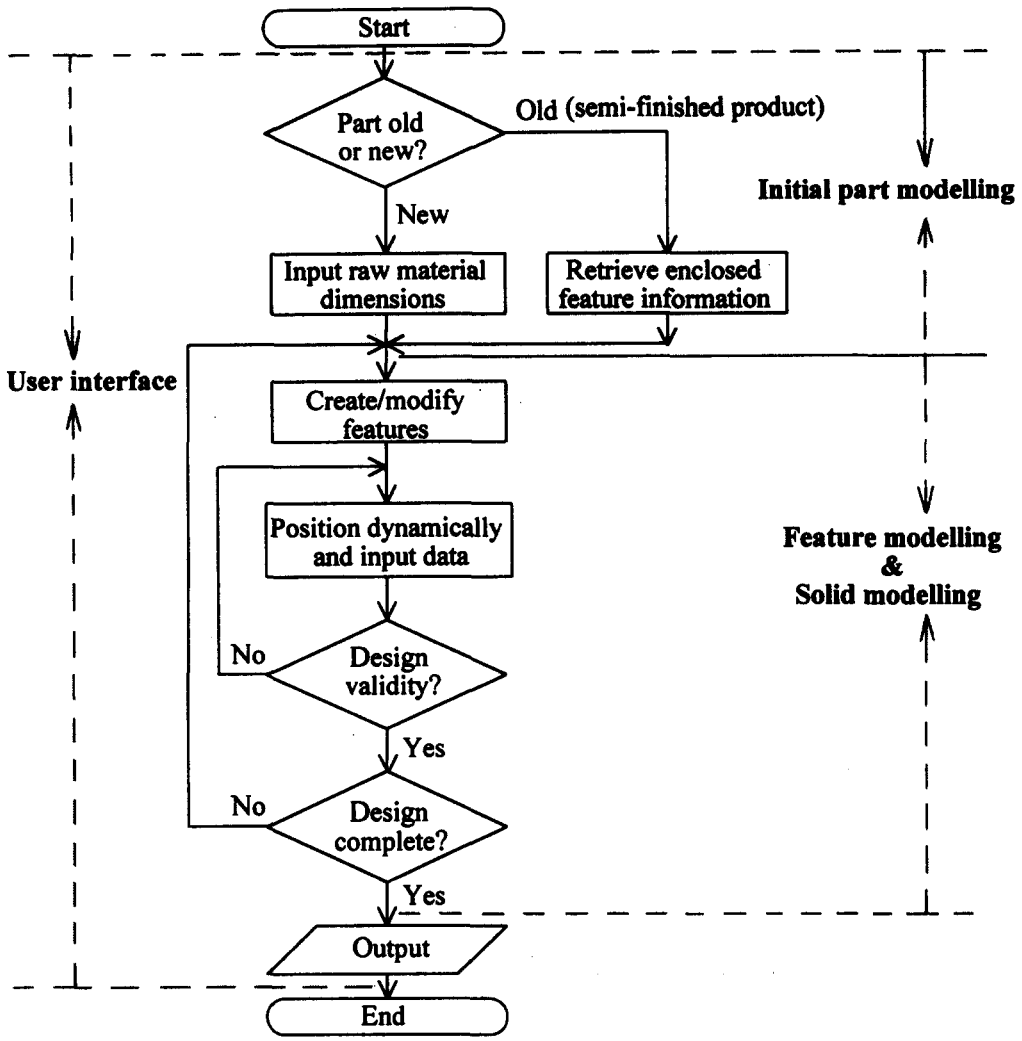

Fig. 10. Flowchart of designing a feature-based part.

The feature operation point must lie on the surface of the raw material or semi-finished product.

(2) Volume constraints:

The volume of the specified feature can not exceed the volume of the raw material or semi-finished product.

(3) Feature-interaction constraints:

- If any existing feature is enclosed by a newly added feature, then the enclosed feature must be removed from the designed part expression list.
- A modified feature may not be enclosed in any existing feature.

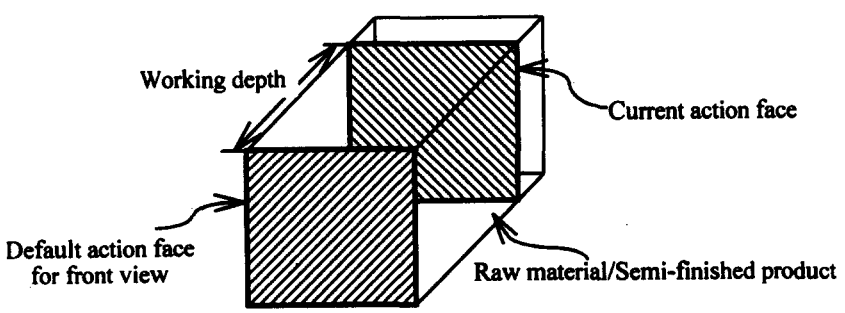

Fig. 11. Illustration of "Action Face" and "Working Depth" definitions.

Table 2. The output feature-file format of the designed part

Raw material data				
Feature name	Coordinates of reference point	Dimension values	Action face	Feature orientation

4.4. *Graphic and text-file output module*

In the output module, a function is provided that can display the designed object in evaluated or unevaluated form. Using the evaluated display image, the designer can get the physical view of a designed part that has been processed by Boolean operations among features and the raw material/semi-finished product. The unevaluated display image skips the above geometric operations, allowing the designer to quickly preview the designed part.

After finishing a part design, the part is saved as a feature-based text file in a CSG-like structure according to the part representation scheme proposed in Section 3. Each feature of a part is stored in ASCII file format, thus greatly reducing the file sizes. More important, through such feature files we can easily integrate the proposed part design system into the CAPP/CAM systems without the complex processes of feature-recognition and -extraction. The general feature-based file format is given in Table 2. Another output of the designed part is saved in a 3D B-rep form.

5. FEATURE-BASED DYNAMIC EDITING

To help designers save time and effort in editing/modelling parts, the proposed system allows retrieval of designed parts. Designers can also seek information on existing features for further modification. With the aid of a mouse for dynamic picking and positioning, they are able to perform editing and querying functions more conveniently.

5.1. *3D feature-based picking function*

When designing a part, designers usually need to keep the information about features in mind. This can be both inconvenient and can result in errors. A practical CAD system should provide designers with a handy way to obtain desired features and associated information. Most previous approaches do not support this type of function in 3D environments. We propose a 3D feature-based picking approach that allows designers to obtain desired features by picking any point in the feature volume. If the point picked is contained in more than one feature, the system highlights each of the candidate features one by one.

When the designer confirms the desired feature, the system pops up a dialog box that provides such information as feature name, reference point, dimension values, orientation, action face, and so on. The designer can then edit the feature directly by modifying the associated parameters, or just note the information for other design work. The examples given in the next section illustrate these operations of feature picking and confirmation.

5.2. *Dynamic editing function*

Editing function include: moving, modifying, deleting, undoing, and redoing. The dynamic editing function applies the dynamic positioning and picking functions to these editing functions. The dynamic moving and modifying functions are illustrated below. The functions of undoing, redoing, and deleting can be similarly implemented.

5.2.1. Moving. Designers may change the original operation point of a feature by means of (a) inputting a new operation point from the keyboard; or (b) dynamic positioning to a desired location using the mouse. The feature-dragging effect is implemented to help this moving function (see Fig. 12). When the feature is being moved, the corresponding coordinate values of the feature's reference point are simultaneously shown on the monitor.

5.2.2. Modifying. The designer can modify any data shown in the feature dialog box except its type. An illustration of modifying is given in Figs 13(a) and (b). Dialog boxes similar to those shown for Figs 8(a) and 12(c) also appear prior to modification.

The examples in Figs 12 and 13 show how easy using the dynamic editing function is, and how feature intersection is handled. The key to accomplish feature-based editing successfully is

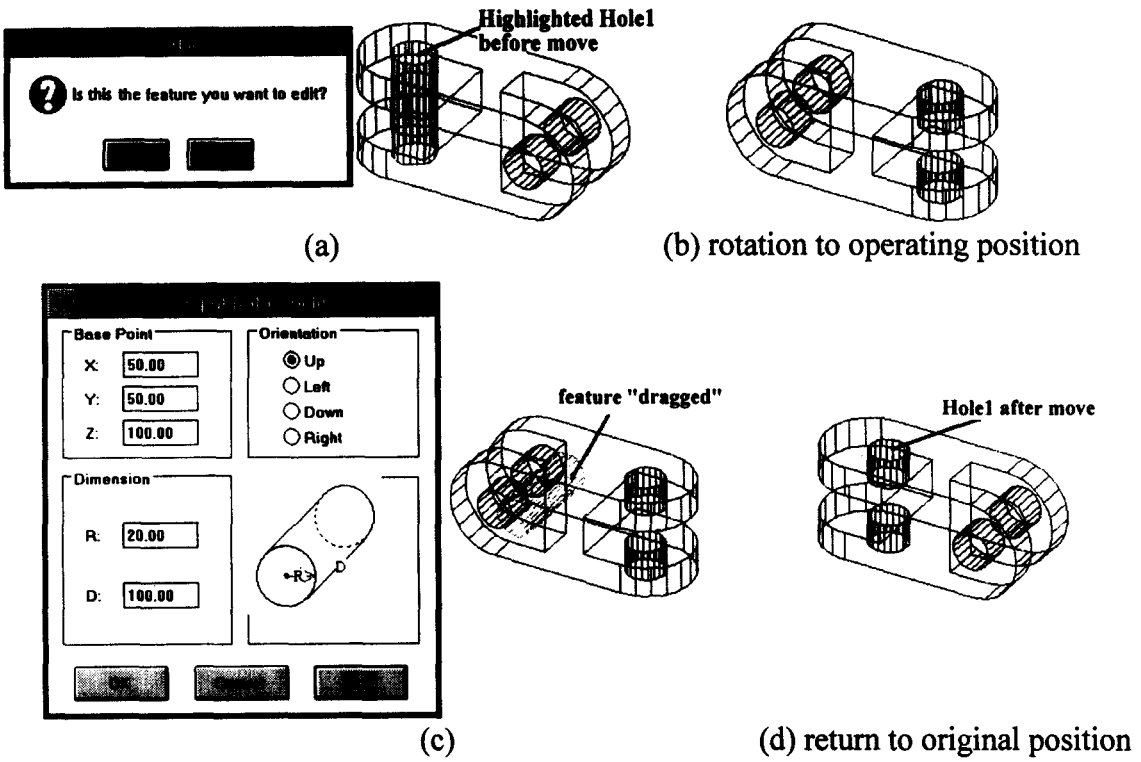

Fig. 12. An illustrative example of moving a feature as outlined in Fig. 4. (a) Hole1 feature selected; (b) rotation of the "Action Force" of the feature into position; (c) feature "dragged" to the desired location; (d) the final result, displayed in its original view, after Hole1 has been removed.

processing feature-interaction problems, such as enclosure and intersection relationships among features using a two-stage approach. In stage 1, the enclosure relationship between the "original feature" (the one picked for editing) and the "modified feature" (the one modified from the original), as well as the intersection relationship between the original feature and other existing features, are considered and the interacted B-rep data is updated. In stage 2, the enclosure relationship between the original feature, the modified feature, and other existing features are dealt with. To maintain a compact data structure, the CSG representation or the feature-list of an edited part is also updated [39].

6. SYSTEM SIMULATION

The authors take advantage of the ACIS solid modelling system to assist in creating and manipulating 3D-prismatic-part features. To ensure a user-friendly interface, PC/MS-Windows [40] is chosen as the development platform. Designers can select features, position them dynamically,

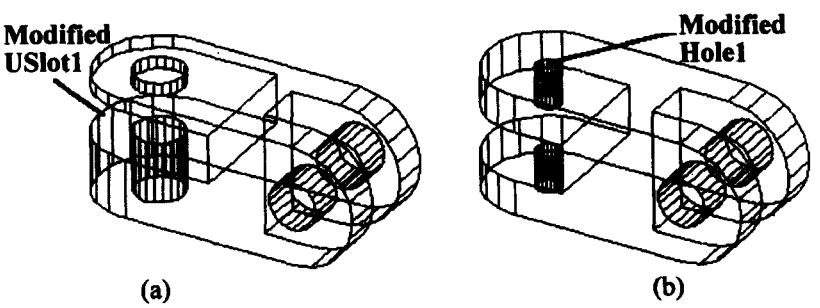

Fig. 13. An illustration of modifying a feature. (a) Moving the USlot1 feature shown in Fig. 4 up, and enlarging the width of USlot1; (b) shrinking the diameter of Hole1.

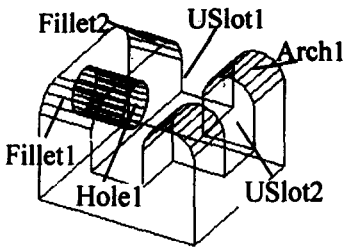

Fig. 14. A sample part to be constructed.

and design parts quickly using the part description system. During the design process, a powerful and efficient feature-based editing module can also be used.

Two examples of the system's functions follow:

Example 1. A sample part (Fig. 14) constructed using the following steps:

Step 1. Select the function "New" from the "File" submenu of the system menu. The part name, material type and geometric dimension of the raw material are then input in the associated dialog box [Fig. 8(b)].

Step 2. Pick the "USlot" icon from the system icon menu as shown in Fig. 15.

Step 3. Input dimension values and relative orientation of the feature "USlot" in the associated dialog box (see Fig. 15). Dynamically drag "USlot" to the desired location on the "Action Face", or input the raw material operation point in the associated dialog box.

Step 4. Use the current designed object as a new raw material. Go to Step 2, and continue to input other features on the proper "Action Face" until the design work is done.

A total of six features are present in this example. Completion of the part-design process normally takes less than 8 minutes.

The feature file of this example is shown in Fig. 16 below. The file format is described in Section 4 and in Perng [13].

Example 2. Using similar procedures, we can design other parts such as those shown in Figs 17(a) and (b). There are 19 features in Fig. 17(a) and 14 in Fig. 17(b). It would normally take only about 20 minutes to complete the design process for each.

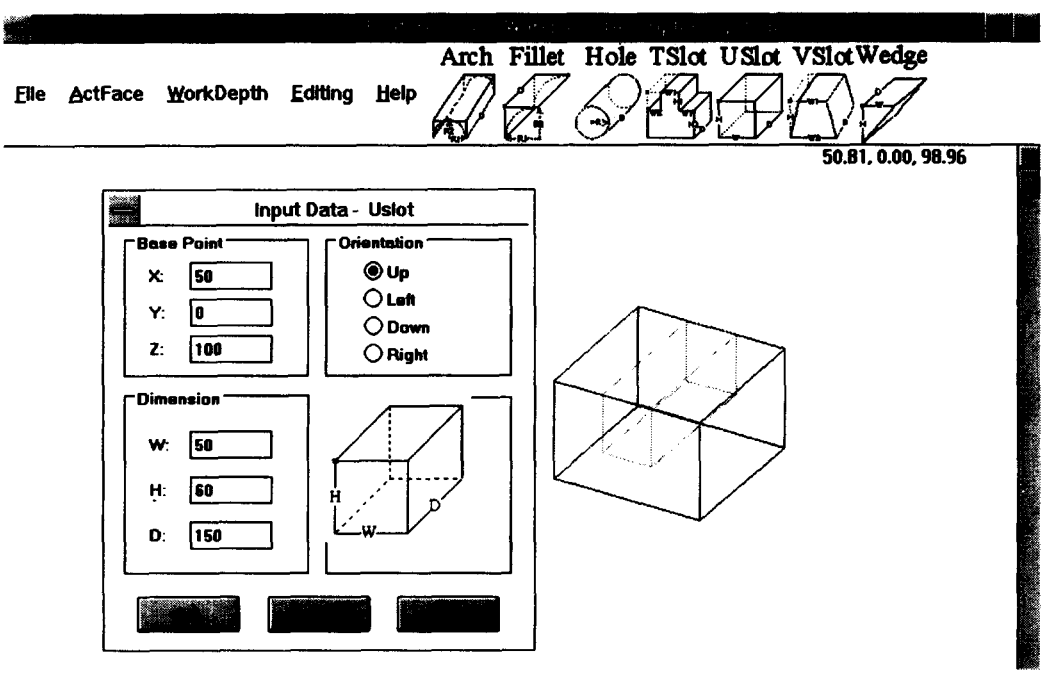

Fig. 15. Illustration for beginning to create the USlot1 feature of the sample part shown in Fig. 14.

```
demo3 LowCarbonSteel 150.00 100.00 150.00
Uslot 50.00 0.00 100.00 50.00 60.00 150.00 0 0
Uslot 150.00 50.00 100.00 50.00 60.00 50.00 3 0
Hole 0.00 75.00 65.00 20.00 50.00 1 0
Fillet 0.00 150.00 100.00 30.00 30.00 50.00 1 0
Fillet 0.00 0.00 100.00 30.00 30.00 50.00 1 3
Arch 125.00 0.00 100.00 25.00 25.00 150.00 0 0
```

Fig. 16. The feature file of the finished part corresponding to Fig. 14.

7. CONCLUSIONS

This paper proposes and implements a new 3D part construction system. The system uses high-level 3D features as basic design entities to facilitate design work. This approach has considerable advantages over conventional 2D-point/line-entity-based CAD systems, allowing designers to concentrate more on their design-concept representations than on point/line drawing operations. Other advantages and contributions of the system are:

1. Features are defined using the object-oriented concepts. It is easy to define and extend feature types. The system is also easy to implement and maintain.
2. CAD/CAPP/CAM system integration can be strongly supported by the proposed part description system. Designers can consider more manufacturing properties earlier in the part-design phase. The feature-file output of a part can be passed directly to the downstream CAPP/CAM systems. The feature-recognition and -extraction processes are completely eliminated.
3. Parts can be designed and modified using high-level features with user-friendly graphic interfaces containing feature icons, dialog boxes, and by dynamic editing functions.
4. Designers can quickly model parts in an unevaluated display mode. The time-consuming Boolean operation between features and raw material can be avoided. Designers can also realize relative positions among features. Furthermore, they can quickly build several part models to choose among.

The proposed 3D part description system can be used to design most commonly used prismatic parts; the class of parts that can be described, though, is limited to seven feature types. Some further research is suggested:

1. Extending the part description range by the inclusion of more features, especially protrusive ones.
2. Binding more manufacturing properties to feature definition, such as tolerance, surface finish, among others, so that model information is sufficient to realize complete integration of CAD and CAPP/CAM.

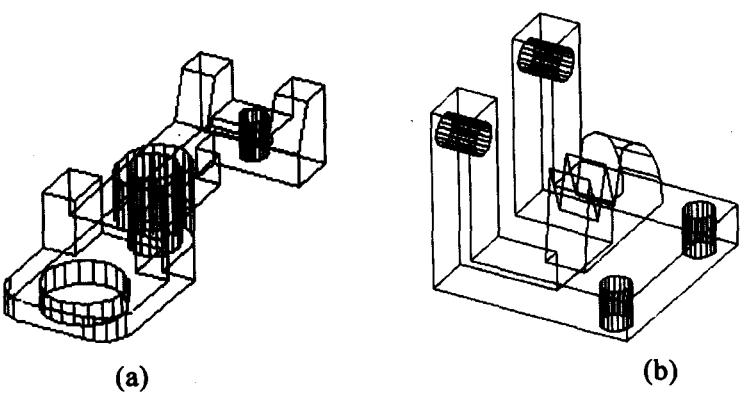

Fig. 17. Examples of prismatic parts that can be described by the proposed system.

3. Application of the current system to automatic part coding and classification systems.
4. Extending to the assembly design.

Acknowledgement—This work is supported by the National Science Council of the Republic of China under Contract NSC 83-0415-E-009-004.

REFERENCES

1. Case, K. and Gao, J., Feature technology: an overview. *Int. J. Computer Integrated Manufacturing*, 1993, **6,** 2.
2. Dixon, J. R., Cunningham, J. J. and Simmons, M. K., Research in designing with features. *Proceeding IFIP TC 5/WG 5.2 Workshop on Intelligent CAD*, ed. H. Yoshikawa and D. Gossard. Elsevier Science Publishers B.V., p. 137.
3. Gindy, N. N. Z., Huang, X. and Ratchev, T. M., Feature-based component model for computer-aided process planning systems. *Int. J. Computer Integrated Manufacturing*, 1993, **6,** 20.
4. Gu, P., A feature representation scheme for supporting integrated manufacturing. *Computers & Industrial Engineering*, 1994, **26,** 55.
5. Lenau, T. and Mu, L., Features in integrated modelling of products and their production. *Int. J. Computer Integrated Manufacturing*, 1993, **6,** 65.
6. Salomons, O. W., van Houten, F. J. A. M. and Kals, H. J. J., Review of research in feature-based design. *Journal of Manufacturing Systems*, 1993, **12,** 113.
7. Shah, J., Sreevalsan, P. and Mathew, A., Survey of CAD feature-based process planning and NC programming techniques. *Computer-Aided Engineering Journal*, 1991, **8,** 25.
8. Unger, M. B. and Ray, S. R., Feature-based process planning in the AMRF. *Computers in Engineering Conference*, ASME, San Francisco, July/Aug. 1988, 563–569.
9. Chang, T. C. and Wysk, R. A., Integrating CAD and CAM through automated process planning. *Int. J. Prod. Res.*, 1984, **22,** 877.
10. Hummel, K. E. and Brooks, S. L., Symbolic representations of manufacturing features for an automated process planning system. *Proceedings of the Winter Annual Meeting of the ASME*, Anaheim, CA, Dec. 1986, 263–270.
11. Choi, B. K. and Barash, M. M., STOPP: an approach to CAD–CAM integration. *Computer-Aided Design*, 1985, **17,** 162.
12. Lee, I. B. H., Lim, B. S. and Nee, A. Y. C., Feature based modelling and process planning in CIM. *Proceedings of the Int. Conf. on Computer Integrated Manufacturing*, Singapore, Oct. 1991, 509–512.
13. Perng, D. B., Chen, Z. and Li, R. K., Automatic 3D machining feature extraction from 3D CSG solid input. *Computer-Aided Design*, 1990, **22,** 285.
14. Perng, D. B. and Cheng, C. T., Feature-based process plan generation from 3D DSG inputs. *Computers & Industrial Engineering*, 1994, **26,** 423.
15. Tornincasa, S. and Zompi, A., Feature-based object oriented approach in automated manufacturing. *Proceedings of the Int. Conf. on Computer Integrated Manufacturing*, Singapore, Oct. 1991, 515–518.
16. van't Erve, A. H. and Kals, H. J. J., XPLANE, a knowledge-base driven process planning expert system. *Proceedings of the Int. Conf. on Computer Aided Production Engineering*, Edinburgh, Apr. 1986, 41–46.
17. Wang, M. T., A geometric reasoning methodology for manufacturing feature extraction from 3-D CAD model. Ph.D. thesis, School of Industrial Engineering, Purdue University, U.S.A., 1990.
18. Wang, M. T. and Chang, T. C., Feature-based recognition for automated process planning. *Proceedings of Manufacturing International '90*, Vol. II, Advances in Manufacturing Systems. ASME, New York, 1990, p. 49.
19. Anderson, D. C. and Chang, T. C., Automated process planning using objected-oriented feature-based design. *Advanced Geometric Modelling for Engineering Applications*, ed. F.-L. Krause and H. Jansen. Elsevier Science Publishers B.V., 1990, p. 247.
20. Chung, J. C. H., Patel, D. R., Cook, R. L. and Simmons, M. K., Feature-based modeling for mechanical design. *Computer & Graphics*, 1990, **14,** 189.
21. Clark, A. L. and South, N. E., Feature-based design of mechanical parts. *AUTOFACT*, 1-69–1-76, Nov. 1987.
22. Desai, V. S. and Pande, S. S., GFM—an interactive feature modeller for CAPP of rotational components. *Computer-Aided Engineering Journal*, 1991, **8,** 217.
23. Dixon, J. R., Artificial intelligence and design: a mechanical engineering view. *Proceedings of AAAI-86 Fifth National Conference on Artificial Intelligence*, **2,** Philadelphia, PA, Aug. 1986, 872–877.
24. Jasthi, S. R. K., Prasad, A. V. S. R. K., Manidhar, G., Rao, P. N., Rao, U. R. K. and Tewari, N. K., A feature-based part description system for computer-aided process planning. *Journal of Design and Manufacturing*, 1994, **4,** 67.
25. Luby, S. C., Dixon, J. R. and Simmons, M. K., Designing with features: creating and using a features data base for evaluation of manufacturability of castings. *Proceedings of ASME International Computers in Engineering Conference and Exhibition*, Chicago, IL, 20–24 July, 1986, 285–292.
26. Luby, S. C., Dixon, J. R. and Simmons, M. K., Creating and using a features data base. *Computers in Mechanical Engineering*, 1986, **5,** 25.
27. Miner, R. H., A method for the representation and manipulation of geometric features in a solid model. Master's thesis, Dept of Mechanical Engineering, Massachusetts Institute of Technology, U.S.A., May 1985.
28. Pratt, M. J., A hybrid feature-based modelling system. *Advanced Geometric Modelling for Engineering Applications*, ed. F.-L. Krause and H. Jansen. Elsevier Science Publishers B.V., 1990, p. 189.
29. Shah, J. J. and Rogers, M. T., Expert form feature modelling shell. *Computer-Aided Design*, 1988, **20,** 515.
30. Shah, J. J. and Rogers, M. T., Feature based modelling shell: design and implementation. *Computers in Engineering Conference*, ASME, San Francisco, July/Aug. 1988, 255–261.
31. Turner, G. P. and Anderson, D. C., An object-oriented approach to interactive, feature-based design for quick turnaround manufacturing. *Computers in Engineering Conference*. ASME, San Francisco, July/Aug. 1988, 551–555.
32. Shah, J. J., Assessment of features technology. *Computer-Aided Design*, 1991, **23,** 331.
33. Li, R. K., Tu, Y. M. and Yang, T. H., Composite feature and variational design concepts in a feature-based design system. *Int. J. Prod. Res.*, 1993, **31,** 1521.

34. Gomes, A. J. P. and Teixeira, J. C. G., Form feature modelling in a hybrid CSG/B-rep scheme. *Computer Graphics*, 1991, **15**, 217.
35. Butterfield, W. B., Green, M. K., Scott, D. C. and Stoker, W. J., Part features for process planning. Report R-86-PPP-01, CAM-I, Incorporated, 611 Ryan Plaza Drive, Suite 1107, Arlington, TX 76011, Nov. 1986.
36. Pratt, M. J., Synthesis of an optimal approach to form feature modelling. *Computers in Engineering Conference*, ASME, San Francisco, July/Aug. 1988, 263–274.
37. Lippman, S. B., *C++ Primer*, 2nd edn. Addison-Wesley, Reading, MA, 1991.
38. Spatial Technology Inc., *ACIS Geometric Modeller Interface Guide*, 2425. 55th Street, Building A, Boulder, Colorado 80301, U.S.A., 1992.
39. Perng, D. B. and Chang, C. F., A new feature interaction solving approach for 3D feature-based part editing problems. *Computer-Aided Design*, May 1996 (in revision).
40. Microsoft Corporation, *Microsoft Windows 3.1 Guide to Programming*. Washington Microsoft Press, Redmond, 1992.