Chapter 2

IGES Standard Protocol for Feature Recognition CAD System

Emad Abouel Nasr, PhD Candaite[1] and Ali Kamrani, PhD[2]
Industrial Engineering Department, Faculty of Engineering, University of Houston, 213 Engineering Building 2, Houston, TX 77204-4008,[1]emadsamir60@yahoo.com, [2]akamarni@uh.edu

Abstract:

Automatic feature recognition from CAD solid systems highly impacts the level of integration. CAD files contain detailed geometric information of a part, which are not suitable for using in the downstream applications such as process planning. Different CAD or geometric modeling packages store the information related to the design in their own databases. Structures of these databases are different from each other. As a result no common or standard structure has been developed so far, that can be used by all CAD packages. For that reason this chapter proposes an intelligent feature recognition methodology (IFRM) to develop a feature recognition system which has the ability to communicate with various CAD/CAM systems. The proposed methodology is developed for 3D prismatic parts that are created by using solid modeling package by using constructive solid geometry (CSG) technique as a drawing tool. The system takes a neutral file in Initial Graphics Exchange Specification (IGES) format as input and translates the information in the file to manufacturing information. The boundary (B-rep) geometrical information of the part design is then analyzed by a feature

recognition program that is created specifically to extract the features from the geometrical information based on a geometric reasoning approach by using object oriented design software which is included in C++ language. A feature recognition algorithm is used to recognize different features of the part such as step, holes, etc. Finally, a sample application description for a workpiece is presented for demonstration purposes.

Key words:

CAD, CAM, CAPP, CIM, Feature Recognition, IGES

2.1. History & Overview

The origins of solid modeling are traced back to the key technological inventions of the 1950s which was Computer Graphics and NC machining. These spawned the developments of computer based geometric systems to aid in the description of object's geometry, which is the main activity to design and manufacture of mechanical parts. This resulted research into the development of Computer Aided Design and Computer Aided Manufacturing (CAD/CAM). Preliminary systems used electronic drafting and wireframe models to represent the shape of three dimensional objects. Subsequent systems developed in the 1960s used polygonal and surface based models which were utilized for a variety of applications in aerospace, marine and automotive industries[4].

Until the 1970s, these models were used in a broad manner. They were merely a collection of lower dimensional entities (polygons, surfaces, lines, curves and points) put together in an unstructured manner to represent a real object. The developments in CAD/CAM led to the crucial questions about the uniqueness and the validity of these models, issues which until then were unimportant from the point of view of computer graphics and its applications. In the late 1970s, these issues were resolved by the Production Automation Project at the University of Rochester, where the term "solid modeling" was coined[11]. This group developed new mathematical models for representing solids and identified the relevant properties of an informationally complete representation. They also identified the mathematical operations that could be used to manipulate these models. After that several other models have been proposed along with different representation schemes[14].

In the 1980s, several solid modeling systems were developed and used in the commercial CAD/CAM world, including the automobile, aerospace

and manufacturing industries. Moreover, many advanced CAD/CAM applications of solid modeling have emerged such as feature and constraint based modeling, automatic mesh generation for finite element analysis, assembly planning including interference checking, higher dimensional modeling for robotics and collision avoidance, tolerance modeling, automation of process planning tasks, etc. Currently, solid modeling techniques have gained importance in the industry and are also actively pursued as a research field in academic institutions[12]. Figure 2-1 lists a summary of solid modeling history evolution technology.

To summarize, solid modeling provides a framework to model and represents an object's shape in the computer, and to perform operations. In addition to, a group of application independent geometric tools and algorithms is provided which can be used to query/analyze the model to obtain unambiguous results. These tools can be used or combined with other application specific tools to perform the required task. The issues related to data structures and geometric algorithms, their efficiency, reliability and robustness also form an important aspect of solid modeling[15].

The academic effort in solid modeling utilizes several disciplines in many applications. That is including algebraic geometry and topology, differential geometry and topology, combinatorial topology, computer science, and numerical analysis. Another well established and closely related field is ***Computer Aided Geometric Design*** (CAGD) which concentrates on developing techniques for freeform surface design used to model curves and surfaces[10].

2.2. Standard Data Format

This section presents discussions related to a standard product data format of an object which considered as the most important tool towards the standardization of product data and at the same time towards the compatible exchange of information among various CAD and CAM systems. IGES format is addressed in details as one of the popular standard format.

2.2.1 Data Transfer in CAD/CAM Systems

Field of data transfer between different CAD/CAM systems is a well established one for a number of years and the paramount importance of CAD/CAM/CAE data transfer between manufacturers and their suppliers and subcontractors has become more apparent. In the early years of CAD/CAM industry, software packages were developed which were employed as direct translators between different systems. Obviously, these packages were used with great success. However, as the number of

CAD/CAM system vendors was increased, the impracticality of using direct translators becomes more apparent. Hence, a few number of neutral format translators developed by various organizations in different countries were introduced into the industrial market[9].

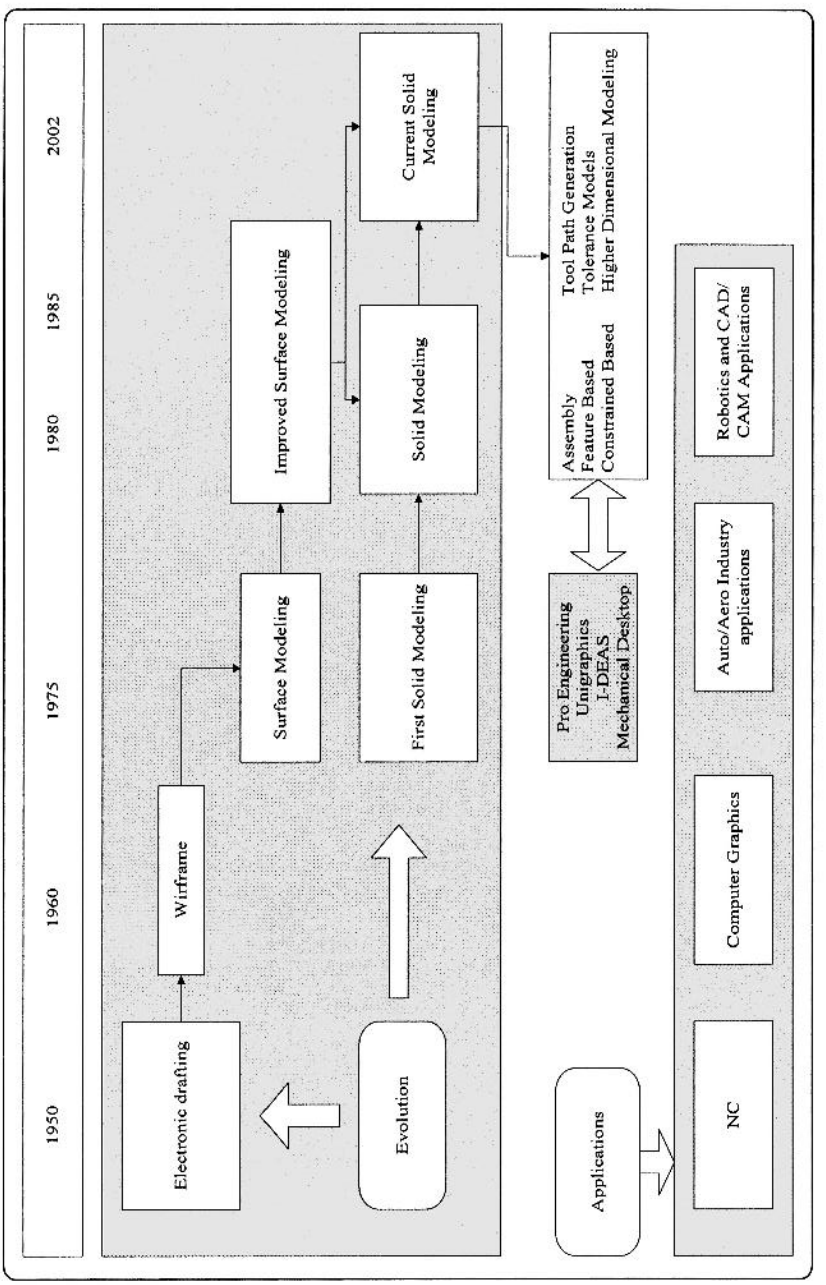


Figure 2-1. Solid Modeling technology evolutions

Some of these translators were tailor made for specific industries and others were accepted as standard tools by various authorized standard organizations. Some of these standards such as Standard for Exchange of Product data (STEP), Data eXchange File (DXF), Product Data Exchange Specifications (PDES) and Initial Graphics Exchange Specifications (IGES) have proved more popular with CAD/CAM system vendors and users[1, 8, 13]. IGES was first developed by National Aeronautical and Space Administration and National Bureau of Standards in 1979. Soon after it was adapted and recognized by American National Standard Institute (ANSI) as a standard tool format. Consequently, IGES has become an acceptable and widely used neutral format translator by many CAD/CAM system vendors[2, 3]. Even though some translators are more broadly used than IGES, this neutral format translator has been through many revisions and has proved reasonably a comprehensive tool in transferring data for parts designed by wireframe, surface or solid models. For this book, IGES version 5.3 documentation was closely studied and adopted[16].

2.2.2 Initial Graphics Exchange Specifications (IGES)

Initial Graphics Exchange Specification (IGES) was developed as a neutral data format for the transmission of CAD data between dissimilar CAD/CAM systems. Although the IGES format does not provide a suitable data format for downstream manufacturing applications, it can be considered as the major driving force to achieve the international standard of product data and the data exchange format. Therefore it is described in details in this section. In order to transfer information, translation is done from one native format to the neutral file and then to another native format[17]. As shown in Figure 2-2, the number of processors needed to transfer data among N different CAD systems using a neutral file is 2 * N.

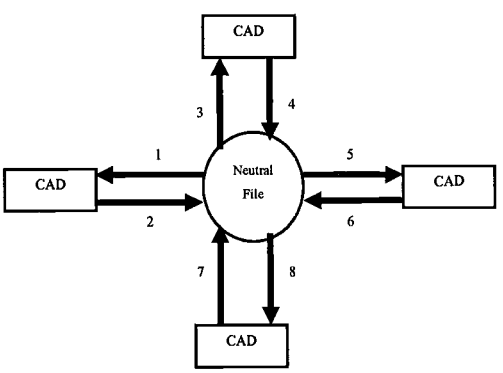


Figure 2-2. Translation using a Neutral File

IGES file is just a document that specifies what should go into a data file. Programmers should write software to translate from their system to the IGES format or vice versa. The program that translates from a native CAD format to IGES is called preprocessor. The program that translates from IGES to another target format is called postprocessor as shown in Figure 2-3.

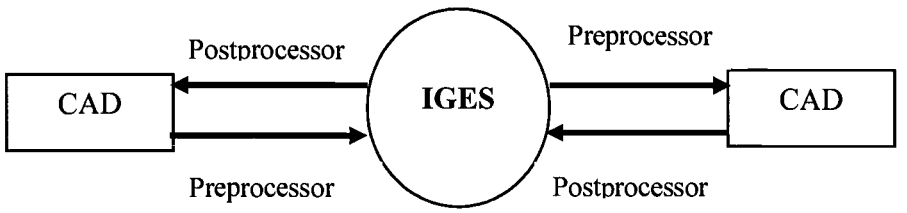


Figure 2-3. IGES translators

2.2.2.1 Structure of IGES file

Similar to the most CAD systems, IGES is based on the concept of entities. Entities could range from simple geometric objects, such as points, lines, plane, and arcs, to more sophisticated entities, such as subfigures and dimensions. Entities in IGES are divided in three categories:

1. Geometric entities: such as arcs, lines, and points that define the object.
2. Annotation entities: such as dimensions and notes that aid in the documentation and visualization of the object.
3. Structure entities: Those define the associations between other entities in IGES file.

An IGES file is a sequential file consisting of a sequence of records. The file formats treat the product definition to be exchanged as a file of entities, each entity being represented in a standard format, to and from which the native representation of a specific CAD/CAM system can be mapped. IGES file is written in terms of ASCII characters as a sequence of 80 character records.

An IGES file consists of five sections which must appear in the following order: Start section, Global section, Directory Entry (DE) section, Parameter Data (PD) section, and Terminate section, as shown in Figure 2-4. The role of these sections is summarized in the following subsections.

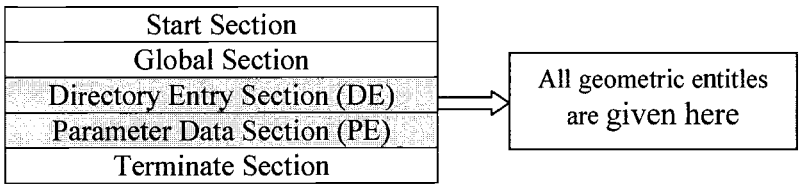


Figure 2-4. IGES file structure

2.2.2.1.1 Start Section

The Start section is a human readable introduction to the file. It is commonly described as a "prologue" to the IGES file. This section contains information such as the names of the sending (source) and receiving (target) CAD/CAM systems, and a brief description of the product being converted.

2.2.2.1.2 Global Section

The Global section includes information that describe the preprocessor and information needed by the postprocessor to interpret the file. Some of the parameters that are specified in this section are:

1. Characters used as delimiters between individual entries and between records (usually commas and semicolons respectively),
2. The name of the IGES file itself,
3. Vendor and software version of sending (source) system,
4. Number of significant digits in the representation of integers and single and double precision floating point numbers on the sending systems,
5. Date and time of file generation,
6. Model space scale,
7. Model units,
8. Minimum resolution and maximum coordinate values,
9. Name of the author of IGES file.

2.2.2.1.3 Directory Entry Section (DE)

The DE section is a list of all the entities defined in the IGES file together with certain attributes associated with them. The entry for each entity occupies two 80-character records which are divided into a total of twenty 8-character fields as shown in Figure 2-5. The first and the eleventh

(beginning of the second record of any given entity) fields contain the entity type number such as 100 for circle, 110 for lines, etc. The second field contains a pointer to the parameter data entry for the entity in the PD section. The pointer of an entity is simply its sequence number in the DE section. Some of the entity attributes specified in this section are line font, layer number, transformation matrix, line weight, and color.

Column	1-8	9-16	.	49-56		65-72	73-80
Line 1	Entity Type	Parameter Entry Pointer		Transformat -ion Matrix		Visible Entity Switch	Sequence Number
Line 2	Entity Type						Sequence Number

Figure 2-5. Structure of Directory Section

2.2.2.1.4 Parameter Data Section (PD)

The PD section contains the actual data defining each entity listed in the DE section as shown in Figure 2-6. For example, a straight line entity is defined by the six coordinates of its two endpoints. While each entity has always two records in the DE section, the number of records required for each entity in the PD section varies from one entity to another (the minimum is one record) and depends on the amount of data. Parameter data are placed in free format in columns 1 through 64. The parameter delimiter (usually a comma) is used to separate parameters and the record delimiter (usually a semicolon) is used to terminate the list of parameters. Both delimiters are specified in the Global section of the IGES file. Column 65 is left blank. Columns 66 through 72 on all PD records contain the entity pointer specified in the first record of the entity in the DE section.

2.2.2.1.5 Terminate Section

The Terminate section contains a single record which specifies the number of records in each of the four preceding sections for checking purposes.

Field	1	2 3 4	5 6	7 8	..	73-80
Circle	100	Z X Y (center of circle)	X_1 Y_1 (start point)	X_2 Y_2 (end point)		Sequence Number
Line	110	X_1 Y_1 Z_1 (start point)	X_2 Y_2 Z_2 (end point)			Sequence Number

Figure 2-6. Structure of Parameter Data Section

2.3. The Feature Extraction Methodology

In this section, a methodology for feature analysis and extraction of prismatic parts for CAM applications is developed and presented. This approach aims to achieve the integration between CAD and CAM. Different CAD or geometric modeling packages store the information related to the design in their own databases. Structures of these databases are different from each other. As a result no common or standard structure has so far developed yet that can be used by all CAD packages. For that reason this research will try to develop an Intelligent Feature Recognition Methodology (IFRM) which has the ability to communicate with the different CAD/CAM systems.

The part design is introduced through CAD software and it is represented as a solid model by using CSG technique as a design tool. The solid model of the part design consists of small and different solid primitives combined together to form the required part design. The CAD software generates and provides the geometrical information of the part design in the form of an ASCII file (IGES) that is used as standard format which provides the proposed methodology the ability to communicate with the different CAD/CAM systems. The boundary (B-rep) geometrical information of the part design is analyzed by a feature recognition program that is created specifically to extract the features from the geometrical information based on the geometric reasoning and object oriented approaches. The feature recognition program is able to recognize these features: slots (through, blind, and round corners), pockets (through, blind, and round corners), inclined surfaces, holes (blind and through) and steps (through, blind, and round corners), etc. These features are called manufacturing information that are mapped to process planning as an application for CAM. Figure 2-7 shows the structure of the proposed methodology.

The intelligent feature recognition methodology (IFRM) presented in this chapter consists of three main phases: (1) a data file converter, (2) an object form feature classifier and (3) a manufacturing features classifier (production rules). The first phase converts a CAD data in IGES/B-rep format into a proposed object oriented data structure. The second phase

classifies different part geometric features obtained from the data file converter into different feature groups. The third phase maps the extracted features to process planning's point of view. Figure 2-8 shows a basic flowchart of the proposed system. The sections that follow will describe the steps of feature extraction in details.

2.3.1 Conversion of CAD data files into Object Oriented Data Structure

As mentioned earlier, IGES is a standard file format for the data defining the object drawing in 3D CAD systems in B-rep structure. The entry fields in IGES format consist of an object's geometric and topological information. The geometric information includes the definition of lines, planes, circles, and other geometric entities for a given object, and the topological information defining the relationships between the object's geometric components, for example, in terms of loops (external loop and internal loop).

An external loop gives the location of major geometric shapes and an internal loop represents a protrusion or a depression (pocket or hole) on an external loop. The fundamental IGES entities, which are related to representing a solid in B-rep structure, are discussed in the following subsection to understand how these entities are defined[16].

2.3.1.1 Basic IGES Entities

Line (entity 110)

A line in IGES file is defined by its end points. The coordinates of start point and terminate point are included in parameter data section of this entity.

Circular Arc (entity 100)

To represent a circular arc in modeling space, IGES provides the information including a new plane (X_T, Y_T) in which the circular lies, the coordinates of center point, start point, and terminate point. A new coordinate system (X_T, Y_T, Z_T) is defined by transferring the original coordinate system (X_O, Y_O, Z_O) via a transformation matrix and all coordinates of points (center point, start point, and terminate point) related to this new coordinate system. The order of end points is counterclockwise about Z_T axis.

Transformation Matrix (entity 124)

This entity can give the relative location information between two

coordinate systems, X_O, Y_O, Z_O coordinate system and X_T, Y_T, Z_T coordinate system.

$$\begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix} X \begin{pmatrix} X_O \\ Y_O \\ Z_O \end{pmatrix} + \begin{pmatrix} T_1 \\ T_2 \\ T_3 \end{pmatrix} = \begin{pmatrix} X_T \\ Y_T \\ Z_T \end{pmatrix} \quad (2.1)$$

Where

$$\begin{vmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{vmatrix} = 1$$

Surface of Revolution (entity 120)

A surface is created by rotating the generatrix about the axis of rotation from the start position to the terminal position. The axis of rotation is a line entity. The generatrix may be a conic arc, line, circular arc, or composite curve. The angles of rotation are counterclockwise about the positive direction of rotation axis.

Point (entity 116)

A point is defined by its coordinates (X, Y, Z).

Direction (entity 123)

A direction entity is a non-zero vector in 3D that is defined by its three components with respect to the coordinate axes. The normal vector of surface can be determined by this entity.

Plane surface (entity 190)

The plane surface is defined by a point on the plane and the normal direction to the surface.

Vertex List (entity 502)

This entity is used to determine the vertex list which contains all the vertexes of the object.

Edge List (entity 504)

This entity is used to determine the edge list which contains all the edges of the object.

Loop (entity 508)

This entity is used to determine the loops which involved in all faces of the object.

Face (entity 510)

This entity is used to determine faces which consist of the object.

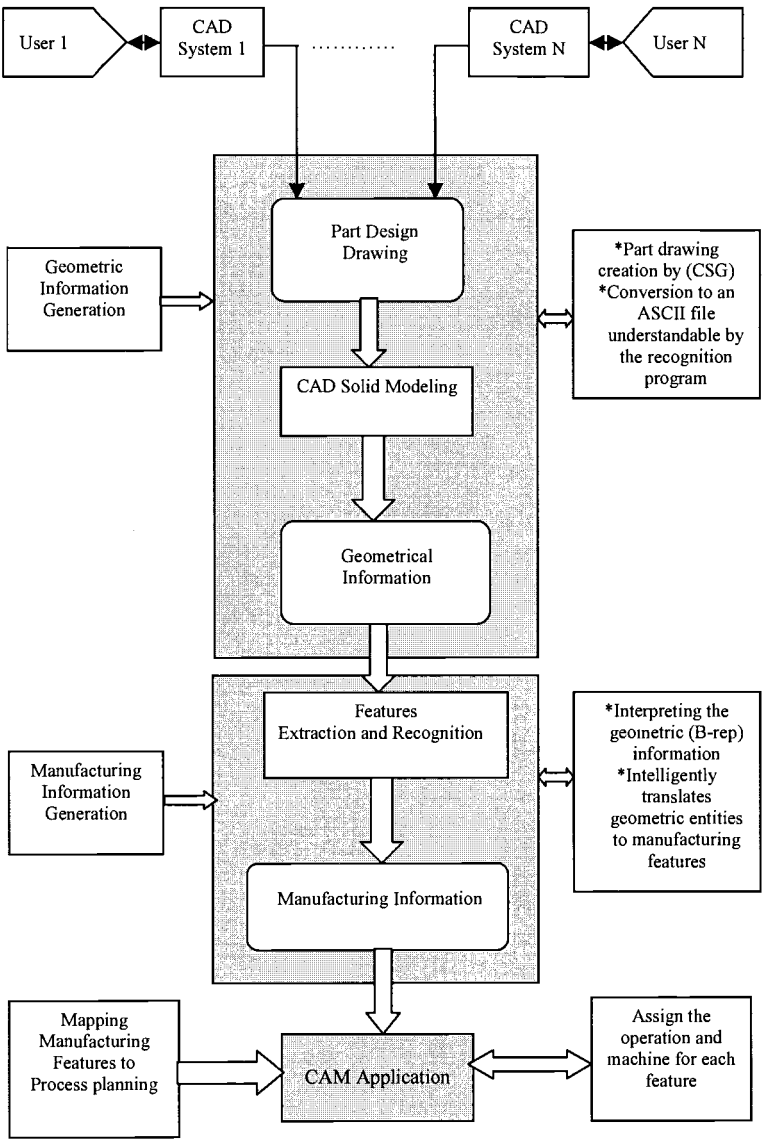


Figure 2-7. Structure of the proposed methodology

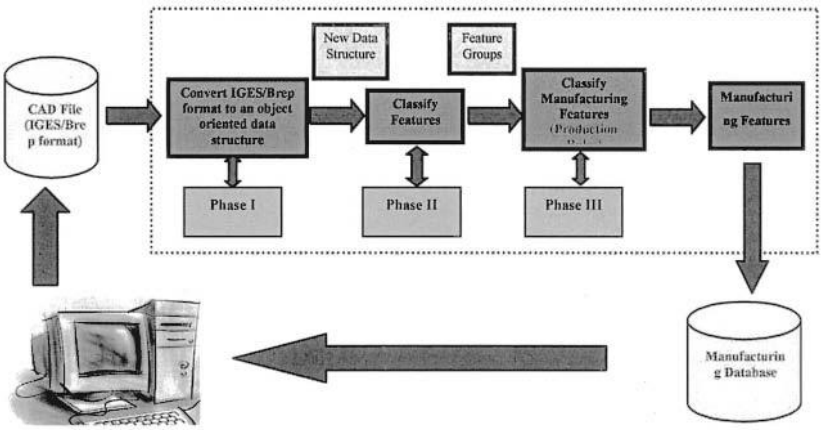


Figure 2-8. Flowchart of extraction and classification of features

Shell (entity 514)

The shell is represented as a set of edge-connected, oriented used of faces. The normal of the shell is in the same direction as the normal of the face.

Right Circular Cylindrical Surface (entity 192)

The right circular cylindrical surface is defined by a point on the axis of the cylinder, the direction of the axis of the cylinder and a radius.

2.3.2 The Overall Object-Oriented Data Structure of the Proposed Methodology

In order to have a good generic representation of the designed object for CAM applications especially for process planning, the overall designed object description and its features need to be represented in a suitable structured database. An object oriented representation will be used in this research. The first step toward automatic feature extraction will be achieved by extracting the geometric and topological information from the (IGES/Brep) CAD file and redefining it as a new object oriented data structure as demonstrated in Figure 2-9.

In this hierarchy, the highest level data class is the designed object (shell). An object consists of manufacturing features that can be classifies into form features which decomposed of either simple or compound/intersecting features. A simple feature is the result of two intersecting general geometric surfaces while compound/intersecting feature is one that results from the interaction of two or more simple features (slot

and pocket) as shown in Figure 2-10. Features are further classified into concave or convex as attributes in the generic feature class. Concave features consist of two or more concave faces, and convex features are decomposed of either one or more convex faces or the interaction between other features in the object as shown in Figure 2-11.

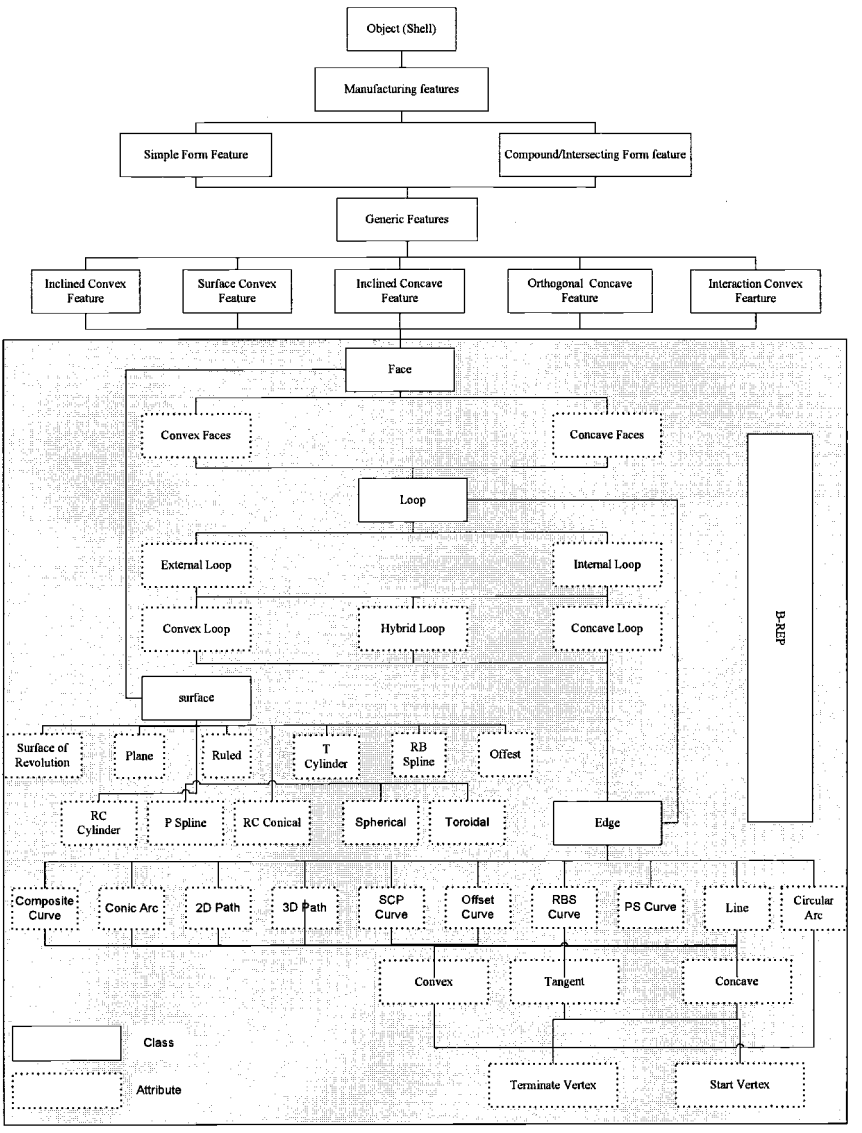


Figure 2-9. Hierarchy of classes and attributes of the designed object

Due to the attributes of the geometric entities of form features (FF), they will be classified into interior form feature ($FF_{interior}$), which is located inside the basic surface, and exterior form feature ($FF_{exterior}$), which is formed by the entire basic surface with its adjacent surfaces. The basic surface refers to the surface in which there are features located in that surface. For the interior form features ($FF_{interior}$), they can be further classified into two low-level categories, convex interior form feature ($FF_{interior\_convex}$) and Concave interior form feature ($FF_{interior\_concave}$). $FF_{interior\_convex}$ is the convex portion in a basic surface, while $FF_{interior\_concave}$ is the concave geometric portion in the surface[3].

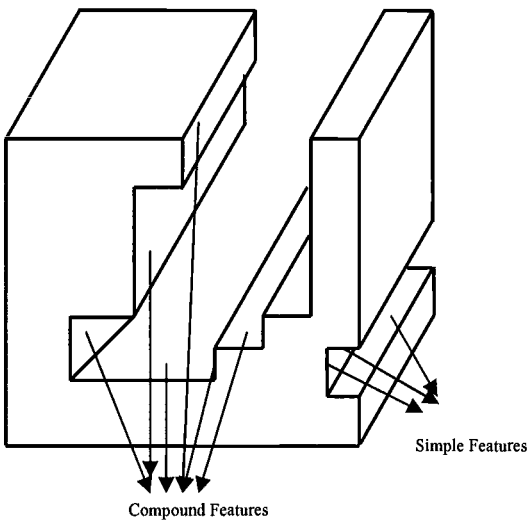


Figure 2-10. Simple and Compound Features

Examples of these form feature categories are shown in Figure 2-12. Figure 2-12 (a) shows a convex portion of the top surface investigated (basic surface) and hence this constitutes a $FF_{interior\_convex}$ (boss), while Figure 2-12 (b) shows a $FF_{interior\_concave}$ in the basic surface. A $FF_{exterior}$ is shown in Figure 2-12 (c) in which $FF_{exterior}$ is constituted by the entire basic surface and its two adjacent surfaces. The $FF_{interior\_concave}$ in Figure 2-12 (b) is a through cylindrical hole. A blind cylindrical hole or a pocket in a basic surface is also a $FF_{interior\_concave}$.

To define concave features, it is basically equivalent to identifying concave faces which are simply defined by a concave edge that joins two adjacent faces. A concave edge is determined by the concavity test that will

be explained later. In general, the edge is defined by a pair of vertices described in the part drawing properties in terms of coordinates (X, Y, Z). On the other hand, convex features can be defined and classified as either inclined, interaction, or surface as shown in Figure 2-13. Inclined convex features are defined by a set of convex faces which are not parallel or perpendicular to minimal enclosing box. The second type of convex feature (interaction) results from the interaction of two or more features. Surface convex features are features that lie on the minimal enclosing box as seen in Figure 2-13.

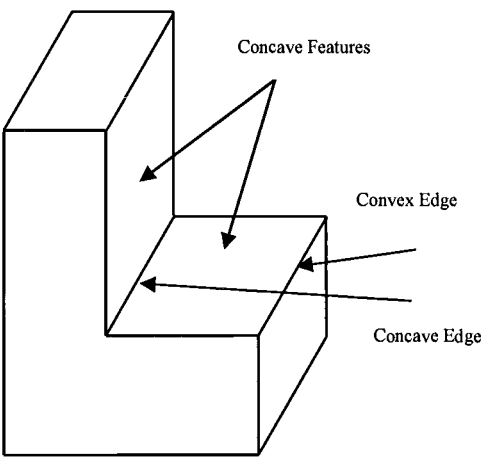


Figure 2-11. Convex and Concave Features and Edges

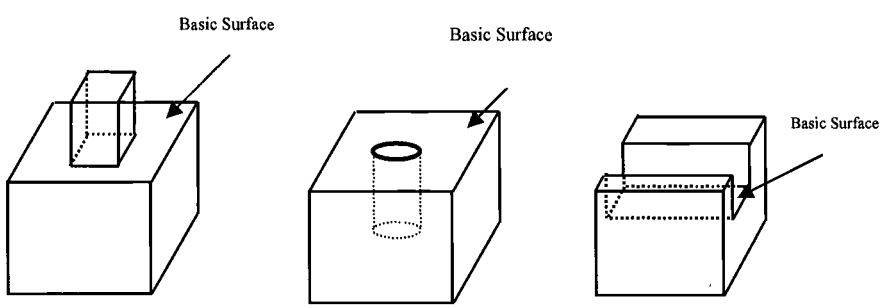


Figure 2-12. Classification of Interior and Exterior Form Features

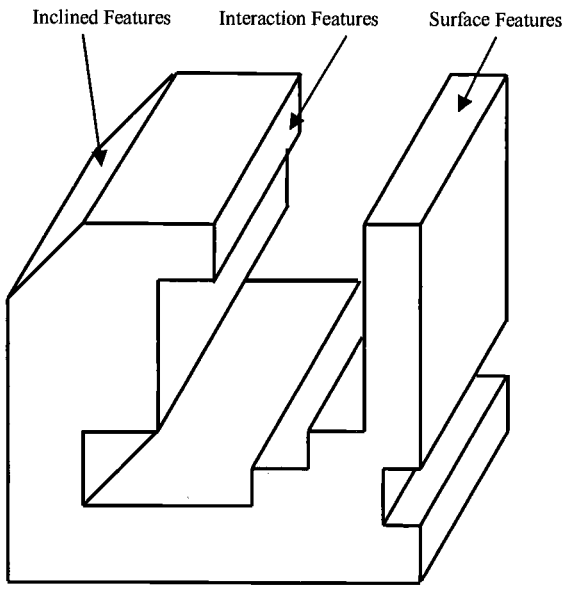


Figure 2-13. Classifications of Convex Features

2.3.2.1 Geometry and Topology of B-rep

The basic geometric entities of a 3D CAD model based on B-Rep description are vertex, edge and face. The compound entities, which consist of basic geometric entities, are shell and loops. Shells and loops are the topological entities since only topological but not geometric information is assigned to them. A solid machining object (O) model can be expressed as

$$O = (V \rightarrow v \in \text{Vertex}, E \rightarrow e \in \text{Edge}, F \rightarrow f \in \text{Faces}) \qquad (2.2)$$

where V, E and F are the sets of object vertices, edges and faces, respectively, and v, e, f are their respective elements. In this expression, each edge has two vertices and is shared by two adjacent surfaces. Each face has certain edges enclosing a specific 2D or 3D shape. The enclosed chain of edges in a surface can form one or more loops. The external loop bounds the face and the others (internal) are located inside the face.

2.3.2.1.1 Classification of Edges

Edges constitute the wireframe of a 3D solid model and they are the intersection boundaries of two adjacent faces. The following proposed edge

categories facilitate the representation of the proposed methodology as proposed in Figure 2-9. To represent edge categories, the normal vectors of the two faces connected that edge and the edge direction are determined as shown in Figure 2-14. Then by applying the connectivity test that will be discussed later, the edges are classified into convex, concave, and tangent edges as shown in Figure 2-15. Also, the angle between the two faces (ϕ) is determined.

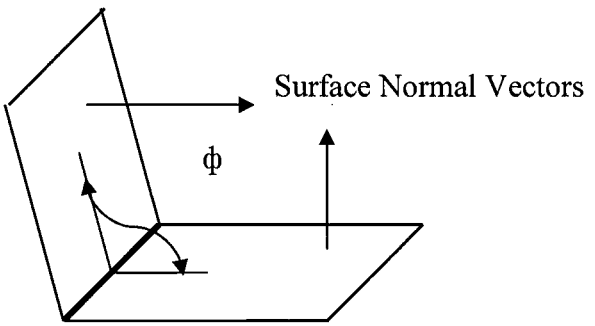
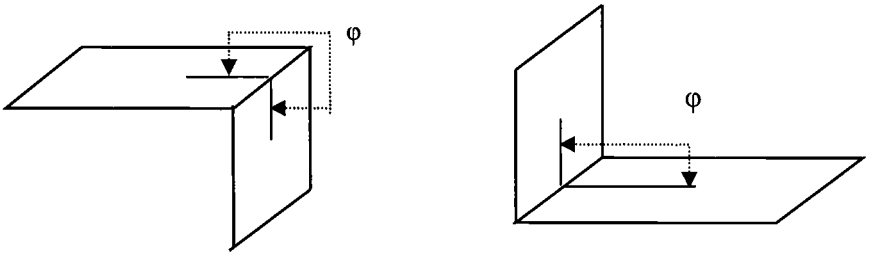


Figure 2-14. The surface normal vectors

2.3.2.1.2 Classification of Loops

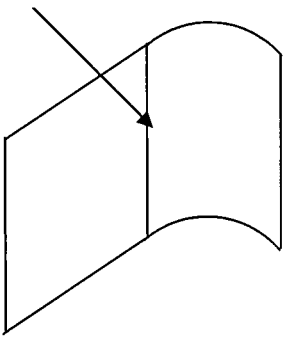
A loop is the border of a surface. It is also the intersection boundary of the surface with its adjacent surfaces. In this research, a loop is used as a fundamental reference to identify the interior and exterior form features. The loop can be classified as proposed in this research into the external loop and internal loop.

External Loop is the outside boundary of a basic surface of which the loop is investigated, while internal loops are located inside the basic surface. In a basic surface, an external loop is the maximum boundary of the basic surface and the internal loop is the internal interaction boundary of the basic surface with its internal features. In Figure 2-16, there are three loops in the basic surface, one is an external loop and the other two are internal loops, which are the intersection boundaries of the basic surface with its internal convex and concave features. In addition to the loops can be further classified into other different categories as shown in Table 2-1.



(a) Convex Edge (b) Concave Edge

Tangent Edge



(c) Tangent Edge

Figure 2-15. Classification of Edges

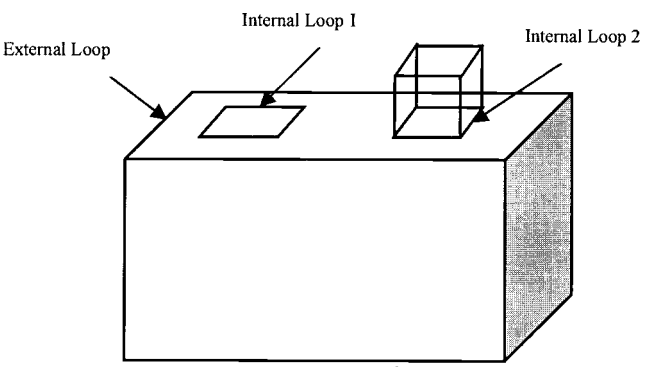


Figure 2-16. Classification of Loops

2.3.2.2 Definition of the Data Fields of the Proposed Data Structure

Generally, Faces are the basic entities that constitute the features, which are further defined by edges that are represented in terms of vertices, which are defined in terms of coordinates in CAD file. Therefore, the hierarchy of the designed object that was described in previous section (Figure 2-9) represents multilevel of different classes. All classes, except for the super class representing an object as a whole, are objects of classes that are higher up in the data structure. For example, each edge object is represented in terms of vertex objects. Table 2-2 displays the data attributes required for each class in the object oriented data structure that is defined before.

Table 2-1. Classification of Loops

Type	Definition	Case
Convex Loop (Loop class type = Convex)	All the edges are convex	If a convex loop is an internal loop, it is called internal convex loop (Internal) (Loop 2 in Fig.2-16). The external convex loop is represented as (External)
Concave Loop (Loop class type = Concave)	All the edges are concave	If a concave loop is an internal loop, it is called internal concave loop (Internal) (Loop 1 in Fig.2-16). The external concave loop is represented as (External)
Hybrid Loop (Loop class type = Hybrid)	The edges have convex and concave types	If a hybrid loop is internal loop, it is called hybrid internal loop (Internal), otherwise, It is hybrid external loop is represented as (External)

2.3.2.3 Algorithms for Extracting Geometric Entities from CAD File

The IGES file is sequentially read (on a line basis) and parsed into appropriate entry classes known as **DEntry** and **PEntry**. As the most important and useful sections of the IGES are the Directory section and the Parameter section. **DEntry** represents an entry in the directory section while **PEntry** represents an entry in the parameter section. The collection of Directory entry classes are contained in a container class called **DSection.**

Table 2-2. Definitions of classes and attributes

Class Name	Attribute	Type
Point	X_Coordinates	(Real)
	Y_Coordinates	(Real)
	Z_Coordinates	(Real)
Vertex	Inherits Point	Point
Verttex_List	Vertex_ID	(Integer)
	Vertex_Count	(Integer)
	Vertex_List	(Vector of Vertex pointers)
Edge	Edge_ID	(Integer)
	Edge_Type	(Enumerated Constants)
	Start_Vertex	(Vertex Pointer)
	Terminate_Vertex	(Vertex Pointer)
	Concavity	(Enumerated Constants)
	Face_Pointers [2]	(Array of Face Pointers)
	Loop_Pointers [2]	(Array of Loop Pointers)
	Dimension	(real)
Edge_List	Edge_Count	(Integer)
	Edge_List	(Vector of Edge Pointer)
Loop	Loop_ID	(Integer)
	Loop_Concavity	(Enumerated Constants)
	Loop_Type (External or Internal)	(Enumerated Constants)
	Edge_List	(A list of edges)
	Face_Pointers	(Pointer to face class)
Surface	Surface_Type	(Enumarated Constants)
Face	Face_ID	Number
	Surface_Pointer	(Pointer to the Surface)
	External_Loop	(Loop Pointer)
	Internal_Loop_Count	Number
	Internal_Loop_List	(Vector of Loop Pointers)
Shell	Vertex_List	(Object of Vertex_List class)
	Edge_List	(Object of Edge_List class)
	Loop_List	(Vector of Loop Pointers)
	Surface_List	(Vector of Surface Pointers)
	Face_List	(Vector of Face Pointers
	Name	(String)
	IGES_File	(Object of IGES_File Class)
Feature	Feature_ID	(Number)
	Feature_Type	(Enumerated constants)
	Feature_Origin	(Vertex Pointer)
	Length	(Real)
	Width	(Real)
	Height	(Real)
	Redius	(Real)
	Face_List	(Vector of Face Pointers
	Direction	(An object of Point Class)
Compound Feature	Feature_ID	(Number)
	Feature_Type	(Enumerated constants)
	Feature_List	(Vector of Feature Pointers)
	Merging_Feature	(Integer)

Similarly, the Parameter entry classes are contained in the **PSection** class. A **Parser class** object is created using these classes to parse the information present in the entries and classify the information into different classes that are used to represent different entities of the diagram described by the IGES file. Two algorithms for extraction of data from the IGES file into a proper set of data structures are defined as follows:

2.3.2.3.1 Algorithm for extracting entries from directory and parameter sections

// Algorithm to extract the directory entries
// and the parameter section entries from the iges file.
// This process takes place during the construction of an object of IGESFile class.
// Each such object represents one IGES file.

1. Create a file descriptor IgesFile
2. Create an empty dSection1 class (container to store dEntry objects).
3. Create an empty pSection1 class (container to store pEntry objects).
4. Open the Iges file for reading using IgesFile file descriptor
   // Read the file to scan and extract the directory and parameter sections.
5. While ReadLine line1 from the Igesfile
   - 5.1 If line1 belongs to Directory section
     - 5.1.1 If line1 is the first line of Dsection
       - 5.1.1.1 Set dIndex to 1
     - 5.1.2 ReadLine line2 from the Igesfile
     - 5.1.3 Create an object dEntry1 of class DEntry
     - 5.1.4 Set dEntry1 index using dIndex
     - 5.1.5 Initialize dEntry1 using string Line1+Line2.
     - 5.1.6 Add dEntry1 to dSection1 class
     - 5.1.7 Set dIndex = dIndex + 1
   - 5.2 If line1 belongs to Parameter Section
     - 5.2.1 If line1 is the first line of PSection
       - 5.2.1.1 Set pIndex to 1
     - 5.2.2 Create an empty string Line2
     - 5.2.3 while pEntry data incomplete
       - 5.2.3.1 ReadLine Line3 from the Igesfile
       - 5.2.3.2 Append Line3 to Line2
     - 5.2.4 Create an object pEntry1 of class PEntry
     - 5.2.5 Set pEntry1 index equal to pIndex
     - 5.2.6 Initialize pEntry1 using string Line1+Line2
     - 5.2.7 Add pEntry1 to pSection1 class
     - 5.2.8 Set pIndex = pIndex + 1

5.3 If line1 belongs to Terminate Section
5.4 exit while loop
6 End of while loop

2.3.3.3.2 Algorithm for extracting the basic entities of the designed part

// Part-extraction module, contained in the Shell class.
// This module extracts entities and groups them into lists
// For example: it creates a list of all Faces in the object represented by the IGES file.
Procedure to extract entities.
1 Create an object vertexList1 of vertexList class
2 Create an object edgeList1 of edgeList class
3 Create a vector of pointers loop List to Loop class
4 Create a vector of pointers face List to Face class
5 Begin parsing entities from the IGES File object
5.1 Initialize counter i=1
5.2 For i=1 to size of dSection1 object from IGES File object
5.3 dEntry1 = DEntry object of index i from dSection1
5.4 if dEntry1 is a vertex list
5.4.1 pEntry1 = PEntry object pointed to by dEntry1, obtained from pSection1 of corresponding IGES File object
5.4.2 Initialize counter j=1
5.4.3 For each vertex present in the pEntry1 object do
5.4.4 Instantiate a new vertex1 object of class Vertex
5.4.5 Assign id as j to vertex1.
5.4.6 Initialize the object with vertex data from pEntry1
5.4.7 Add to vertexList1 a pointer to vertex1
5.4.8 Increment j by 1
5.4.9 End of For loop
5.5 End For
5.6 Initialize counter i=1
5.7 For i=1 to size of dSection1 object from IGES File object
5.8 dEntry1 = DEntry object of index i from dSection1
5.9 If dEnry1 is an edge list
5.9.1 pEntry1 = PEntry object pointed to by dEntry1, obtained from pSection1 of corresponding IGES File object
5.9.2 Initialize counter j = 1
5.9.3 For each edge present in the pEntry1 class do

5.9.4 Instantiate a new object edge1 of class Edge
5.9.5 Assign id as j to edge1
5.9.6 Retrieve dEntry2 object that contains edge specific data
5.9.7 Retrieve pEntry2 object corresponding to dEntry2
5.9.8 Instantiate a new object edgeS that is specific to the edge type
5.9.9 Initialize the edgeS object with edge1 and data from pEntry2
5.9.10 Assign start and terminate vertex to the edge using pointers from the vertexList1 object.
5.9.11 Add to edgeList object a pointer to edgeS
5.9.12 Increment j by 1
5.9.13 End For
5.10 End For
5.11 Initialize counter i=1
5.12 For i=1 to size of dSection1 object from IGES File object
5.13 dEntry1 = DEntry object of index i from dSection1
5.14 If dEntry1 is a loop
5.14.1 pEntry1 = PEntry object pointed to by dEntry1, obtained from pSection1 of corresponding IGES File object
5.14.2 Create an instance loop1 of Loop class
5.14.3 Assign an id to loop1 using (size of loop List + 1)
5.14.4 For each edge in the pEntry1
5.14.5 If next edge is a Vertex
5.14.5.1Add to the loop1 a pointer to the vertex
5.14.6 Else Add to the loop1 a pointer to the edge
5.14.7 End For
5.14.8 Add to loop List the pointer to loop1.
5.15 End For
5.16 Initialize counter i=1
5.17 For i=1 to size of dSection1 object from IGES File object
5.18 dEntry1 = DEntry object of index i from dSection1
5.19 If dEntry1 is a Face
5.19.1 pEntry1 = PEntry object pointed to by dEntry1, obtained from pSection1 of corresponding IGES File object
5.19.2 Create an instance face1 of Face class
5.19.3 Assign an id to face1 using (size of faceList +1)
5.19.4 Obtain dEntry2 containing the DEntry of the Surface type of the Face from dSection1.
5.19.5 Instantiate a surface1 object specific to the type of the surface of the face.

5.19.6 Add to face1 a pointer to that surface object
5.19.7 For each Loop on the Surface
5.19.7.1Add to face1 a pointer to loop object from loop List that represents this loop.
5.19.7.2For each Edge in the loop add a pointer to face1.
5.19.8 End For
5.19.9 Add to face List the pointer to face1
5.20 End For
6 End of extract entities procedure.

2.3.3.4 Extracting Form Features from CAD Files

The edge direction and the face direction are the basic entities information that is used to extract both simple and compound form features from the object data structure. The edge directions in object models can be defined such that, when one walks along an edge, its face is always on the left-hand side. When an edge is in the external loop of a face, its direction will be in a counter clockwise direction relative to the surrounding face[5, 6]. On the other hand, when an edge is in the internal loop of a face, its direction will be clockwise as shown in Figure 2-17.

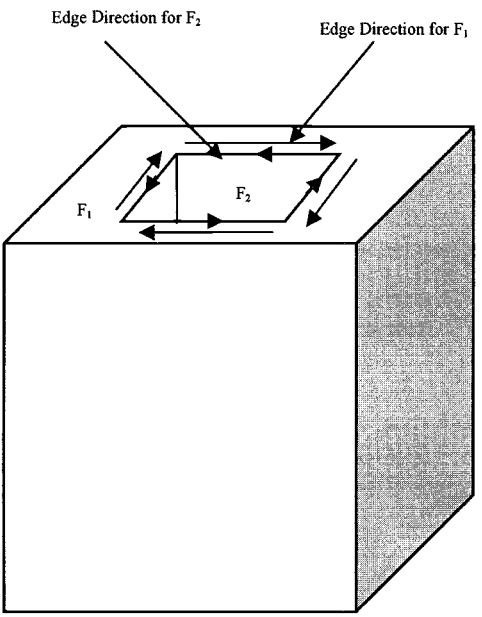


Figure 2-17. The Direction of Edge

The concave edge test used in research is based on cross product of the normal vectors of the two faces joined by a given edge. This is done by applying vector geometry to the face and edge direction vectors. Figure 2-18 shows the symbols used in this test where the i^{th} face is designated as F_i, its corresponding normal direction vector is defined as $\mathbf{N_i}$ in the upward direction with respect to the given face, and the k^{th} edge is designated as E_k .

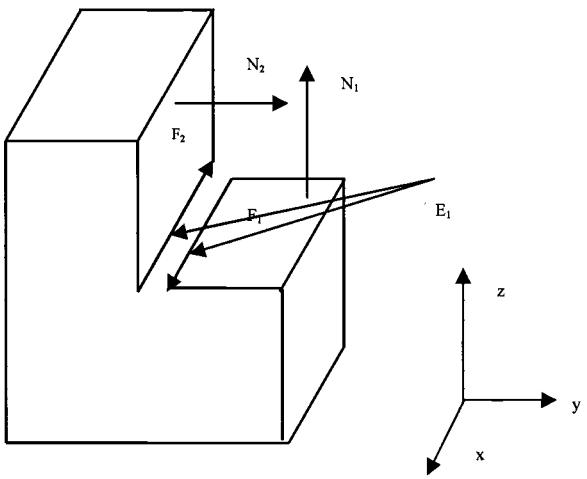


Figure 2-18. A concave Edge Example

The edge (E_K) shared by two faces (F_i and F_j) where the order is right to left from the left side of the edge view perspective. The direction vectors of the faces as described above (N_i and N_j). Finally the edge's directional vector is given with respect to face F_i using loop L_i that contains the edge (E_k). The following is the methodology for concavity test[7]:

[1] The cross (vector) product (V) of the directional vectors of the faces is determined as follows:

$$V = N_i \, X \, N_j \tag{2.3}$$

[2] The direction of the edge E_k with respect to the face F_i is determined. The normal vector N_i of face F_i must be the first component in the cross product of step 1.

[3] If the direction vector of edge E_k from step 2 is in the same direction of cross product V, then the edge E_k is convex edge that concludes F_i and F_j are convex faces, otherwise, it will be concave edge and F_i and F_j are concave faces. Also, if the cross product vector v is a zero vector that means the edge is tangent category.

2.3.3.4.1 An Example for identifying the concave edge/faces

The following is an example for identifying the concave edge/faces by using the example in Figure 2-18:

1. Find the face normal vectors $N_1 = [0\ 0\ 1]$ and $N_2 = [0\ 1\ 0]$.
2. Find the edge direction vector for E_1 with respect to $F_1 = [1\ 0\ 0]$
3. Find the cross product (V) = [0 0 1] X [0 1 0] = [-1 0 0]
4. The edge direction of E_1 and V have the opposite direction, so the edge E_1 is a concave edge and F_1 and F_2 are concave faces.

This procedure will be done on all the edges of the object to define the concave or convex faces. Moreover, concave features will be identified by the premise that concave faces include at least one concave edge with adjacent concave faces forming a concave face group. Each concave face group defines a concave feature. Similarly, adjacent convex faces form a convex face group.

2.3.3.4.2 Algorithm for determining the concavity of the edge

The following algorithm is used to find the concavity of a line entity. This algorithm is used by the Line class to find the entities.

1 $\text{Lengthof line} = \sqrt{[(\text{startX}-\text{termX})^2+(\text{startY}-\text{termY})^2+(\text{startZ}-\text{termZ})^2]}$

2 concavity = UNKNOWN
3 If face1 surface type == PLANE and face2 surface type == PLANE
  3.1.1 Assign crossDir = cross product of face1 normal vector and face2 normal vector
  3.2 If crossDir == 0
    3.2.1 concavity = TANGENT
  3.3 Else
    3.3.1 Calculate the direction vector edgeDir for the line with respect to the loop
      3.3.1.1 If crossDir is in the same direction as edgeDir
        3.3.1.1.1 concavity = CONVEX
      3.3.1.2 Else
        3.3.1.2.1 concavity = CONCAVE
4 If face1 surface type == RCCSURFACE and face2 surface type == PLANE

4.1 Find dir1 (direction) that is orthogonal to the plane containing the edge and the axis of face1
4.2 If dir1 and normal of face2 are orthogonal to each other
4.2.1 concavity = TANGENT

2.3.3.4.3 Algorithms for feature extraction (production rules)

The following are some of the algorithms used for extraction of features.

Rule 1: Feature: **STEP THROUGH** (Figure 2-19)
1. For every concave edge of type Line in the edge list.
2. If the two common faces (face1 and face2) of the edge (e_1) are plane and orthogonal to each other
   2.1. if outer loop concave edge count equals 1 in both the faces
      2.1.1. STEP THROUGH found.
      2.1.2. Create a new StepT object and add to feature list
3. End For

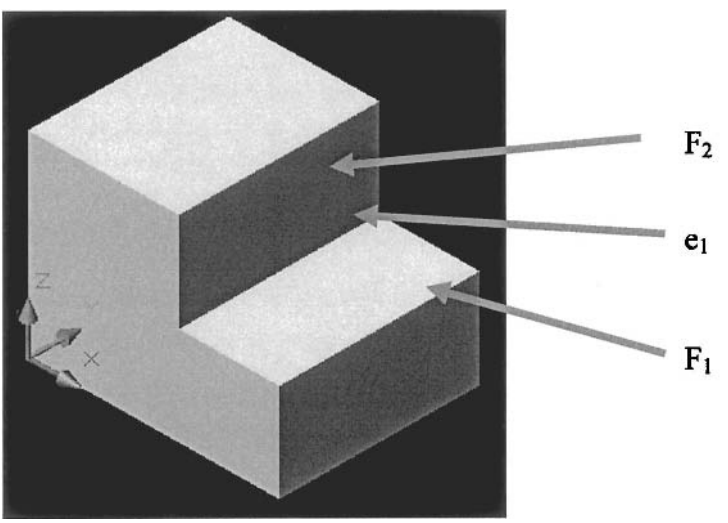


Figure 2-19. STEP THROUGH

Rule 2: Feature **STEP THROUGH ROUND CORNER** (Figure 2-20)
1. For every 3 tangent edges of type Line in the edge list (e_1, e_2, e_3)
2. If the common face of the 2 edges is quarter cylindrical surface (F_2, F_3).

2.1. The other two faces (F_1, F_4) connected to edges are perpendicular to each other.
2.1.1. If concave edge count of the outer loops of the four faces equals 0 each.
2.1.1.1. STEP THROUGH ROUND CORNER found
2.1.1.2. Create a new StepT_RC object and add to feature list.
3. End For

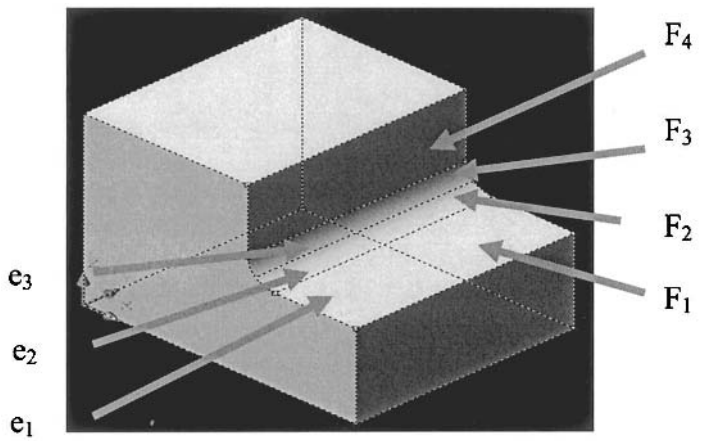


Figure 2-20. STEP THROUGH ROUND CORNER

In general, the following steps are the proposed methodology for feature's extraction and classifications:

Step 1: Extract the geometry and topology entities for the designed object model from IGES file:
(a) Identify vertices, edges, faces, loops of the object.

Step 2: Extract topology entities in each basic surface and Identify its type:
(a) Identify the total number of loops in each surface.
(b) Identify the basic surface due to total number of loops.
(c) Classify the loops into different types (concave, convex, and hybrid).

Step 3: Test the feature's existence in the basic surface based on loops.

Step 4: Identify feature type:
(a) Identify Exterior Form Features ($FF_{exterior}$) by searching for hybrid loop.
(b) Identify Interior Convex Form Features ($FF_{interior}$) by searching

for convex loop.

(c) Identify Interior Concave Form Features ($FF_{interior}$) by searching for concave loop.

Step 5: Identify the detailed features and extract the related feature geometry parameters:

(a) Identify feature's details (number of surfaces, surface type).

(b) Identify the parameters of each feature (length (L), width (W), height (H), radius (R).

(c) Identify the relative location of each feature due to the origin coordinates of the object.

Step 6: Identify the detailed machining information for each feature and the designed part:

(a) Identify the operation sequence of the designed part.

(b) Identify the operation type, the machine, and the cutting tool for each feature.

(c) Identify the tool approach/machining direction for each feature.

(d) Identify the removed machining volume for each feature.

2.4. An Illustrative Example

The designed object as shown in Figure 2-21 consists of five solid primitives which are four blocks (prismatic raw material, slot, blind step, and through pocket) and one cylinder (through hole). After the part is created, its IGES file will be generated. By applying the proposed methodology, the results are shown in Tables 2-3 to 2-8.

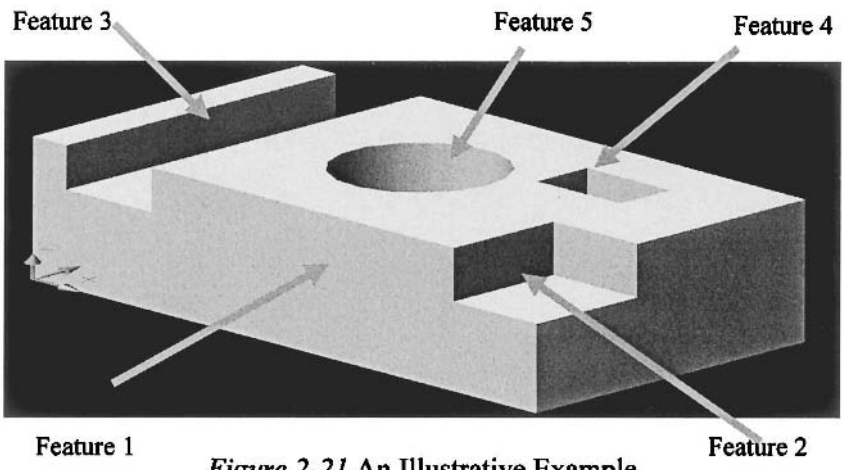


Figure 2-21. An Illustrative Example

Step 1: Extract the geometry and topology entities for the designed object

model from IGES file.

Step 2: Extract topology entities in each basic surface and Identify its type.

Table 2-3. Extraction of Vertices

NO.	Vertex ID	Coordinates (X,Y,Z)	NO.	Vertex ID	Coordinates (X,Y,Z)
1	[1]	(19,8,16)	18	[18]	(25,0,16)
2	[2]	(19,8,0)	19	[19]	(30,6,16)
3	[3]	(11,8,16)	20	[20]	(30,0,0)
4	[4]	(11,8,0)	21	[21]	(30,16,0)
5	[5]	(25,13,8)	22	[22]	(30,16,16)
6	[6]	(25,10,8)	23	[23]	(7,16,11)
7	[7]	(20,13,8)	24	[24]	(7,0,11)
8	[8]	(20,10,8)	25	[25]	(2,16,11)
9	[9]	(20,10,16)	26	[26]	(2,0,11)
10	[10]	(25,10,16)	27	[27]	(2,16,16)
11	[11]	(20,13,16)	28	[28]	(2,0,16)
12	[12]	(25,13,16)	29	[29]	(7,0,16)
13	[13]	(30,6,10)	30	[30]	(7,16,16)
14	[14]	(30,0,10)	31	[31]	(0,0,16)
15	[15]	(25,6,10)	32	[32]	(0,0,0)
16	[16]	(25,0,10)	33	[33]	(0,16,16)
17	[17]	(25,6,16)	34	[34]	(0,16,0)

Table 2-4. Extraction of Edges

Edge ID	Edge Type	Coordinates			Length /Radius	Concavity
		Starting Point	Terminate Point	Center		
[1]	Line	[1]	[2]		16	Tangent
[2]	Cir. Arc	[3]	[1]	(15,8,16)	4	
[3]	Line	[4]	[3]		16	Tangent
[4]	Cir. Arc	[2]	[4]	(15,8,0)	4	

Table 2-4. Extraction of Edges (cont.)

[5]	Line	[5]	[6]		3	Concave
[6]	Line	[7]	[5]		5	Concave
[7]	Line	[8]	[7]		3	Concave
[8]	Line	[6]	[8]		5	Concave
[9]	Line	[9]	[8]		8	Concave
[10]	Line	[10]	[9]		5	Convex
[11]	Line	[10]	[6]		8	Concave
[12]	Line	[11]	[7]		8	Concave
[13]	Line	[9]	[11]		3	Convex
[14]	Line	[12]	[5]		8	Concave
[15]	Line	[11]	[12]		5	Convex
[16]	Line	[12]	[10]		3	Convex
[17]	Cir. Arc	[4]	[2]	(15,8,0)	4	
[18]	Cir. Arc	[1]	[3]	(15,8,16)	4	
[19]	Line	[13]	[14]		6	Convex
[20]	Line	[15]	[13]		5	Concave
[21]	Line	[16]	[15]		6	Concave
[22]	Line	[14]	[16]		5	Concave
[23]	Line	[17]	[15]		6	Concave
[24]	Line	[18]	[17]		6	Convex
[25]	Line	[16]	[18]		6	Convex
[26]	Line	[19]	[13]		6	Convex
[27]	Line	[17]	[19]		5	Convex
[28]	Line	[14]	[20]		10	Convex
[29]	Line	[21]	[20]		16	Convex
[30]	Line	[22]	[21]		16	Convex
[31]	Line	[19]	[22]		10	Convex
[32]	Line	[23]	[24]		16	Concave
[33]	Line	[25]	[23]		5	Convex
[35]	Line	[24]	[26]		5	Concave
[36]	Line	[27]	[25]		5	Convex
[37]	Line	[28]	[27]		16	Convex
[38]	Line	[26]	[28]		5	Convex

Table 2-4. Extraction of Edges (cont.)

[39]	Line	[29]	[24]	5	Convex
[40]	Line	[30]	[29]	16	Convex
[41]	Line	[23]	[30]	5	Convex
[42]	Line	[29]	[18]	18	Convex
[43]	Line	[31]	[28]	2	Convex
[44]	Line	[31]	[32]	16	Convex
[45]	Line	[20]	[32]	30	Convex
[46]	Line	[27]	[33]	2	Convex
[47]	Line	[33]	[31]	16	Convex
[48]	Line	[22]	[30]	23	Convex
[49]	Line	[32]	[34]	16	Convex
[50]	Line	[34]	[21]	30	Convex
[51]	Line	[33]	[34]	16	Convex

Table 2-5. Extraction of Loops

Loop ID	Loop Type	Loop Category	Face ID	Edge ID
[1]	External	Concave	[1]	[1][2][3][4]
[2]	External	Concave	[2]	[5][6][7][8]
[3]	External	Hybrid	[3]	[9][10][11][8]
[4]	External	Hybrid	[4]	[12][13][9][7]
[5]	External	Hybrid	[5]	[14][15][12][6]
[6]	External	Hybrid	[6]	[11][16][14][5]
[7]	External	Concave	[7]	[1][17][3][18]
[8]	External	Hybrid	[8]	[19][20][21][22]
[9]	External	Hybrid	[9]	[23][24][25][21]
[10]	External	Hybrid	[10]	[26][27][23][20]
[11]	External	Convex	[11]	[26][19][28][29][30][31]
[12]	External	Hybrid	[12]	[32][33][34][35]
[13]	External	Hybrid	[13]	[36][37][38][34]
[14]	External	Hybrid	[14]	[39][40][41][32]
[15]	External	Convex	[15]	[25][42][39][35][38][43][43][44] [45][28][22]

Table 2-5. Extraction of Loops (Cont.)

[16]	External	Convex	[16]	[37][46][47][43]
[17]	External	Convex	[17]	[27][31][48][40][42][24]
[18]	Internal	Convex	[17]	[10][13][15][16]
[19]	Internal	Concave	[17]	[18][2]
[20]	External	Convex	[18]	[29][45][49][50]
[21]	Internal	Concave	[18]	[4][17]
[22]	External	Convex	[19]	[51][49][44][47]
[23]	External	Convex	[20]	[41][48][30][50][51][46][36][33]

Table 2-6. Extraction of Faces

Face ID	Surface Type	Normal Vector	Concavity	Number of Loops	Loop ID
[1]	Surface of revolution		Concave	1	[1]
[2]	Plane Surface (parameterized)	(0,0,1)	Concave	1	[2]
[3]	Plane Surface (parameterized)	(0,1,0)	Concave	1	[3]
[4]	Plane Surface (parameterized)	(0,0-1)	Concave	1	[4]
[5]	Plane Surface (parameterized)	(0,-1,0)	Concave	1	[5]
[6]	Plane Surface (parameterized)	(-1,0,0)	Concave	1	[6]
[7]	Surface of revolution		Concave	1	[7]
[9]	Plane Surface (parameterized)	(1,0,0)	Concave	1	[9]
[10]	Plane Surface (parameterized)	(0,-1,0)	Concave	1	[10]
[11]	Plane Surface (parameterized)	(0,0,-1)	Concave	1	[11]
[12]	Plane Surface (parameterized)	(0,0,1)	Concave	1	[12]
[13]	Plane Surface (parameterized)	(0,0,-1)	Concave	1	[13]
[14]	Plane Surface (parameterized)	(-1,0,0)	Concave	1	[14]

Table 2-6. Extraction of Faces (Cont.)

[15]	Plane Surface (parameterized)	(0,-1,0)	Convex	1	[15]
[16]	Plane Surface (parameterized)	(0,0,1)	Convex	1	[16]
[17]	Plane Surface (parameterized)	(0,0,1)	Convex	3	[17][18] [19]
[18]	Plane Surface (parameterized)	(0,0-1)	Convex	2	[20][21]
[19]	Plane Surface (parameterized)	(-1,0,0)	Convex	1	[2]
[20]	Plane Surface (parameterized)	(0,1,0)	Convex	1	[23]

Step 3: Test the feature's existence in the basic surface based on loops.

Step 4: Identify feature type.

Step 5: Identify the detailed features and extract the related feature geometry parameters.

Table 2-7. Extraction of Features

Feat. ID	Feature Type	Faces ID	Edges ID	Location	Feature Name	Dimension L	W	H	R
[1]	prismatic	[11][18][20] [15][18][19]	[29][30][37][40][44] [45][47][49][50][51]	[32] = (0,0,0)	Raw Material	30	16	16	
[2]	$FF_{exterior}$	[8][10][9]	[20][21][23]	[16] = (25,0,10)	Step_Blind	6	6	5	
[3]	$FF_{exterior}$	[13][12][14]	[34][[32]	[26] = (2,0,11)	Slot_Through	16	5	5	
[4]	$FF_{interior}$	[3][4][5][6][2]	[5][6][7][8] [9][12][14][11]	[8] = (20,10,8)	Pocket_Blind	8	5	3	
[5]	$FF_{interior}$	[1][7]	[1][2][3][4] [17][3][18]	(15,8,0)	Hole_Through	16			4

Step 6: Identify the detailed machining information for each feature and the designed part:

Table 2-8. Machining Information

Operation Sequence	Feat. ID	Feature Type	Operation Type	Machine	Cutting Tool	Tool Approach	Removed Volume
1	[3]	Slot_Through	Slotting_Milling	Milling	End Milling Cutter	[0,1,0] or [0,-1,0]	400.00
2	[2]	Step_blind	Shoulder_Milling	Milling	Side Milling Cutter	[-1,0,0] or [0,1,0]	180.00
3	[4]	Pocket_Blind	Pocket_Milling	Milling	End Milling Cutter	[0,0,-1]	120.00
4	[5]	Hole_Through	Drilling	Drilling	Twist Drill	[0,0,-1]	804.25

2.5. Summary

In this chapter, a methodology for feature analysis and extraction of prismatic parts for CAM applications is proposed and the implemented system is presented. This approach aims to achieve CAD/CAM integration. Different CAD or geometric modeling packages store the information related to the design in their own databases and the structures of these databases are different from each other. As a result no common or standard structure has so far been developed yet that can be used by all CAD packages. For this reason this proposed research will develop a feature recognition algorithm which has the ability to communicate with the different CAD/CAM systems.

Part design was introduced through CAD software and was represented as a solid model. The CAD software generates and provides the geometrical information of the part design in the form of an ASCII file (IGES) that is then used as standard format which provides the proposed methodology the ability to communicate with the different CAD/CAM systems. The boundary (B-rep) geometrical information of the part design is analyzed by a feature recognition program that is created specifically to extract the features from the geometrical information based on the geometric reasoning approach. The proposed feature recognition program is able to recognize the following features: slots, pockets, holes, steps, counter bore holes, counter sink holes, etc. These features, called manufacturing information, are mapped to process planning function as an application for CAM. The system is developed in C++ language on a PC-based system. Finally, a case study is used to illustrate the validity of the proposed methodology.

References

1. N. Ahmad and A. Haque, Manufacturing Feature Recognition of Parts Using DXF Files, Fourth International conference on Mechanical Engineering, Dhaka, Bangladesh, **1**(1), 111-115 (2001).
2. M.P. Bhandarkar, B. Downie, M. Hardwick, and R. Nagi, Migrating from IGES to STEP: One to One Translation of IGES Drawing to STEP Drafting Data, Computers in Industry, **41**(3), 261-277 (2000).
3. M.W. Fu, S.K. Ong, W.F. Lu, I.B.H. Lee and A.Y.C. Nee, An Approach to Identify Design and Manufacturing Features from A Data Exchanged Part Model, Computer Aided Design, **35**(1), 979-993 (2003).
4. C. Hoffmann, and J. Rossignac, A Road Map to Solid Modeling, IEEE Transactions on visualization and Computer graphics, **2**(1), 136-149 (1996).
5. J. Hwang, Rule-Based Feature Recognition: Concepts, Primitives and Implementation, Thesis, Arizona State University (1988).
6. C-H. Liu, D-B. Perng, and Z. Chen, Automatic form Feature Recognition and 3D part Recognition from 2D CAD Data, Computer and Industrial Engineering, **14**(4), 689-707 (1994).
7. S. Liu, M. Gonzalez and J. Chen, Development of An Automatic Part Feature Extraction and Classification System Taking CAD Data as Input, Computers in Industry, **29**(1), 137-150 (1996).
8. X. Ma, G. Zhang, S. Liu, and X. Wang, Measuring Information Integration Model for CAD/CMM, Chinese Journal of Mechanical Engineering, **16**(1), 59-61 (2003).
9. S. Mansour, Automatic Generation of Part Programs foe Milling Sculptured Surfaces, Journal of Materials Processing Technology, **127**(1), 31-39 (2002).
10. D. Natekar, X. Zhang, and G. Subbarayan, Constructive Solid Analysis: A Hierarchal, Geometry-Based Meshless Analysis Procedure for Integrated Design and Analysis, Computer Aided Design, **36**(5), 473-486 (2004).
11. A. Reqicha and H. Voelcker, Solid Modeling: A Historical Summary and Contemporary Assessment, IEEE Computer Graphics and Applications, **2**(2) 9-24 (1982).
12. O.W. Salomons, Review of Research in Feature Based Design, Journal of Manufacturing Systems, **12**(2), 113-132 (1993).
13. R. Sharma and J. Gao, Implementation of STEP Application Protocol 224 in An Automated Manufacturing Planning System,

Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, **216**(1), 1277-1289 (2002).
14. S. Somashekar and W. Michael, an Overview of Automatic Feature Recognition Techniques for Computer-Aided Process Planning, Computers in Industry, **26**(1) 1-21 (1995).
15. V. Sundaarajan and P.K. Wright, Volumetric Feature Recognition For Machining Components With Freeform Surfaces, Computer Aided Design, **36**(1), 11-25 (2004).
16. B. G. William, Initial Graphics Exchange Specification IGES 5.3, ANS US PRO/IPO-100 (1996).
17. I. Zeid, CAD/CAM Theory and Practice (McGraw-Hill, Inc., 1991).

http://www.springer.com/978-0-387-23290-4

Rapid Prototyping
Theory and Practice
Kamrani, A.K.; Nasr, E.A. (Eds.)
2006, XXXIV, 324 p., Hardcover
ISBN: 978-0-387-23290-4