THE DEFINITION AND RECOGNITION OF SHAPE FEATURES FOR VIRTUAL PROTOTYPING VIA MULTIPLE GEOMETRIC ABSTRACTIONS

by

RATNAKAR SONTHI

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

(Mechanical Engineering)

at the

UNIVERSITY OF WISCONSIN-MADISON

1999

UMI Number: 9956182

Bell & Howell Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

A dissertation entitled

The Definition and Recognition of Shape Features for Virtual Prototyping via Multiple Geometric Abstractions

submitted to the Graduate School of the
University of Wisconsin-Madison
in partial fulfillment of the requirements for the
degree of Doctor of Philosophy

by

Ratnakar Sonthi

Date of Final Oral Examination: November, 11, 1999

Month & Year Degree to be awarded: **December** ,1999 **May** **August**

* *

Approval Signatures of Dissertation Readers:

Signature, Dean of Graduate School

Virginia S. Hinshaw/EH

Readers' Page. This page is not to be hand-written except for the signatures

Abstract

Analyzing the visual, functional and manufacturability aspects of a part design requires information regarding shape features on the part. Most often the required shape feature information is not readily available for an analysis. One method to obtain the feature information from a part is via feature recognition.

The current research presents a novel approach to interactively define features and subsequently recognize features. Features are defined and recognized through the use of three levels of geometric abstraction. The abstractions used are 1) Boundary Representation (B-Rep) elements (faces and edges), 2) Curvature Regions and 3) Primitive Shapes. The abstractions are used as a basis for a Feature Definition Language and features are defined through a graphical user interface to the Feature Definition Language. The graphical user interface allows features to be defined interactively and flexibly in a CAD system dependent/independent manner. Subsequent to feature definition, feature recognition is performed via a sub-graph-matching algorithm.

By virtue of using the above three abstractions, the current approach allows the use of a single definition for recognizing features that differ in topology and geometry. Moreover, features for different manufacturing applications, such as drilling, milling and molding, are defined and recognized using the same algorithms. The information present in the features that are recognized from a part is utilized to perform a manufacturability analysis of the part.

ACKNOWLEDGEMENT

I thank my adviser Prof. Rajit Gadh, not only for his guidance but also for providing me the opportunity to join the doctoral program in the first place. I also thank my lab mates who have helped my research at the I-CARVE Lab. In specific, I thank Girish Kunjur and Stefan Schoen for their initial help in the research and to Fan Zhao for his help during the final stages. Though my interaction with Prof. John Uicker, Jr. has been limited, I would like to thank him for just being there, in case I needed any help.

A special thanks to my lab mate Shang-Sheng Liu for all the time that we spent together in the lab. His company has been the icing on my collective experience in the lab. I would like to thank Yong Lu and Shyamsundar Nuggehalli for their help during my defense.

The first two years of my stay at Madison were some of the most memorable years in my life. For this, I thank Viresh Ratnakar, Prakash Raman and Harinarayan Kambainathan. I thank Viresh for the influence that he has had on me. Being in his company, my outlook towards life has been transformed for good!

I thank my parents Krishna Sastry and Uma Devi for giving me an education and much more. I thank them for all those things that they have given - from their heart. Finally, I thank my wife and soul mate Himaja for all the little things that she does for me and her undying love and support.

Having thanked the people that I have known, I would now like to thank the nameless, formless Source from which all that we are originates. I would like to thank the Source for making me realize through experience that there was a possibility of its existence.

I would like to think that the decision to come to Madison for a Doctoral degree has been made not by "me" but the Source (from which all thoughts arise). For, it is in my Madison where my search started. In my search for the Source I have come to believe in a guiding light and I now associate the nameless and formless being of Sai Baba of Shirdi with that. I thank him for the fleeting experiences of "union" that he has offered me, which have made a potentially strenuous life of Doctoral Studies an enjoyable experience. Om Sri Sai.

Table of Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

1.1 Virtual Prototyping

Virtual Prototyping is the design and generation of an early version of a product in a computer based (i.e., "virtual") environment. This early version does not necessarily have all the features of the final product but has enough of the key features to allow testing of the product design against the product requirements, as shown in Figure 1-1. The term "Virtual" implies that a physical version of the product design is not yet created but a computer-based representation of the product is available to the user for observation, analysis and manipulation. The cost involved in generating a virtual prototype is generally less than the cost of building a physical prototype. Hence a virtual prototype can be an economical alternative to a relatively expensive physical prototype. Furthermore, the short cycle time from design to manufacture to testing in Virtual Prototyping ensures that the designer can be promptly informed of design errors.

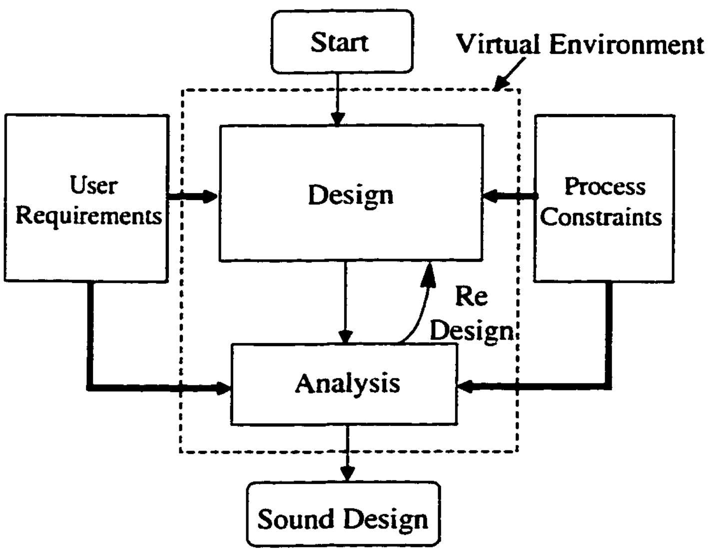


Figure 1-1: Virtual Prototyping

The above mentioned factors have induced companies to place increasing emphasis on Virtual Prototyping. The main steps involved in Virtual Prototyping are Design, Analysis and Re-design [95]. Once a part is designed, a variety of attributes of the part design are required to analyze it for applications such as manufacturability (e.g., injection molding) and manufacturing process planning. One such attribute of a part design that is required for an analysis is information about the shape of the part, also called "feature information" [4].

1.2 Features for Virtual Prototyping

In the context of Virtual Prototyping a feature is defined as

"*Information about the generic shape or characteristics of a product that can be associated with certain attributes and knowledge useful for reasoning about that product*"[74]

In the above definition, reasoning refers to applications such as manufacturing process planning [31][32], manufacturability analysis [27] and analysis for functionality. [13]. The utility of features in these three applications is explained as follows:

Manufacturing Process Planning: Consider the object shown in Figure 1-2 that must be manufactured by machining. Figure 1-2(a) shows the pictorial representation of the solid model of a part containing no feature information. Figure 1-2(b) shows the pictorial representation of the part containing machining feature (hole and base feature) information [93].

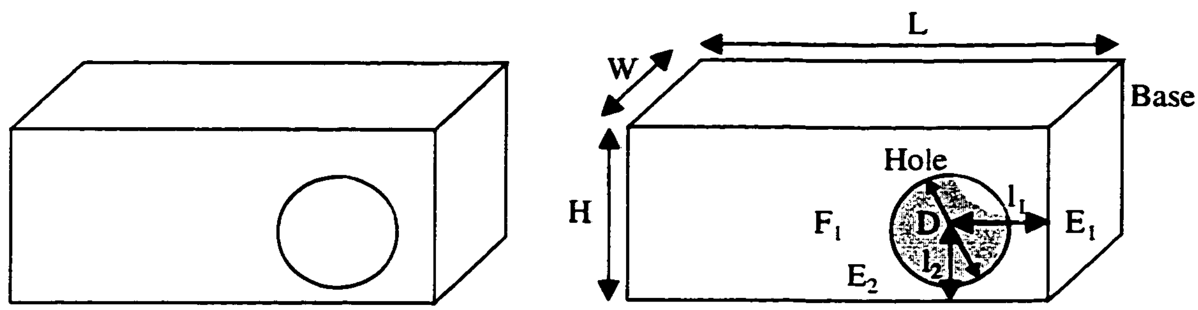
The information available in the object in Figure 1-2(a) is insufficient to plan the placement and size of the machine tool. On the other hand, consider the shape information available in the representation in Figure 1-2(b). In this representation, the following information is available:

(i) the object comprises a base and a hole

(ii) the length, width and height of the base feature are L, W and H

(iii) the diameter of the hole feature is D, the center of the hole is at a distance l_1 from edge E_1 on face F_1 and distance l_2 from edge E_2 on face F_1.

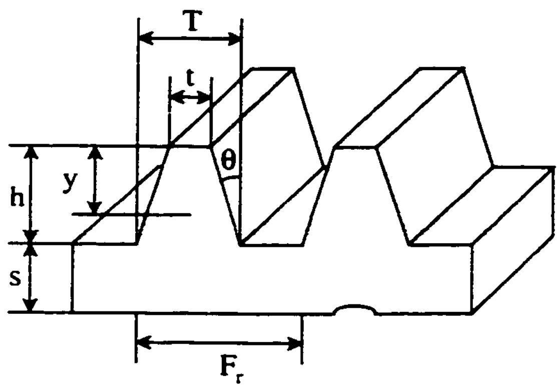
This information is sufficient to reason about the machining process (size of the stock, type of cutting tools to be used, cutting tool sizes and cutting tool placement/motion).



(a) Model with no feature information (b) Model with feature information

Figure 1-2: Machining process planning features

Manufacturability Analysis: Manufacturability Analysis is performed on a part to determine whether the part can be manufactured or not using a particular manufacturing process. Manufacturability Analysis requires information about the shape of the part and the requisite shape information is obtained via the attributes and characteristics of generic shape forms (cylinder, cube, etc.), i.e., features of the part. Figure 1-3 shows an example of the use of features for injection molding analysis [38][39]. The rib feature thickness and the wall feature thickness (thickness of the base feature) are used to analyze the part for sinkmarks and warpage using a design rule.



Design Rule for RIB feature

IF:	The material is GE NORYL N190
and	The wall thickness is [s]
and	The input root thickness [T] is greater than 0.8 [s]
THEN:	possibility of bad sinkmark = 9/10
and	possibility of warpage = 8/10
and	Warning Message: Reduce [T] to be less than 0.8 [s]

[Huh 91]

s: part wall thickness
h: rib height
t: rib top thickness
θ: rib draft angle
F_r: rib frequency
y: distance to the centroid
T: rib root thickness

Figure 1-3: Features for manufacturability analysis

Functional Analysis: Analysis of the functional aspects of a product design via Finite Element Analysis also requires knowledge of feature information. For example, feature knowledge is useful in the finite element meshing of a product design. Liu [51] presents an approach for automatic mesh generation through the use of features. Generating a finite element mesh for a feature is simpler than generating a mesh for the entire object. Features are first obtained from the solid model and meshed individually. The individual meshes for the features are then combined to obtain the mesh for the entire model.

Therefore, feature information is required for performing an analysis of the part. Prior to describing the methods to obtain feature information, the type of feature information that is required for an analysis is described.

1.3 Type of feature information required

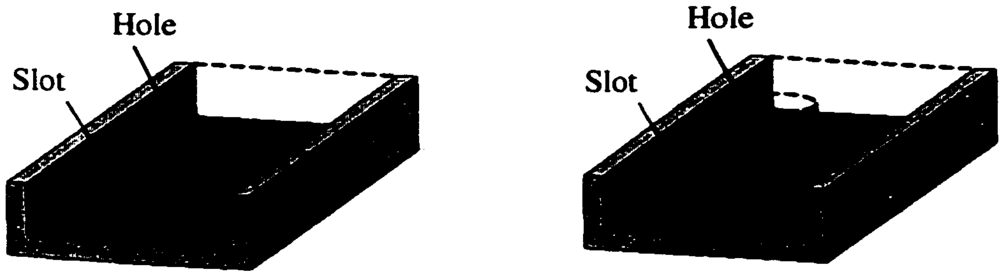
The type feature information that is required for reasoning about a part is as follows:

(a) topology and geometry of the feature, where geometry refers to the types of faces (cylindrical, planar, etc.), edges (linear, arc, etc.) and vertices of the solid model that belong to the feature and topology refers to the connectivity information between the faces, edges and vertices;

(b) feature dimensions such as the thickness of a rib and

(c) location of a feature with respect to other features on the part (for example, the location of the hole in Figure 1-2 with respect to the base).

The above feature information is sufficient when there are no interactions between the features in a model. However, when there are several features in an object there usually are interactions [62] between them. In addition to information about individual features and their parameters, information about the type of interaction between features -- e.g., interference interaction, adjacency interaction and remote interaction [62] -- is also required. Such information is used, for example, in the process of machining to plan the order of machining operations. For the object in Figure 1-4(a), the knowledge of the interaction between the hole and the slot, and the knowledge of the type of interaction ("hole is sitting on the slot" - interference interaction) allows determination of the sequence of machining operations - the slot should be machined before the hole [32]. However, if the hole is determined as shown in Figure 1-4(b), the hole and slot features can be machined independent of each other.



(a) Slot and Hole interaction (b) Variation in Hole feature

Figure 1-4: Feature interaction

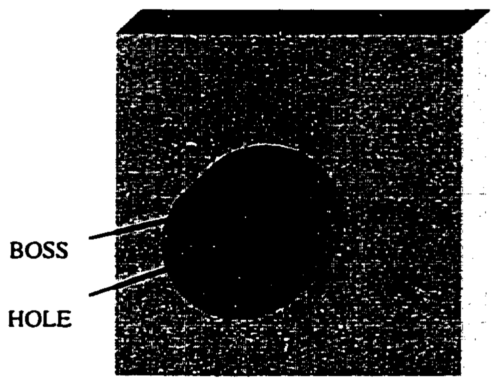


Figure 1-5: Composite feature.

Also, interaction between features potentially could result in composite features, the knowledge of which may again be needed for an analysis. In Figure 1-5 there is a cylinder with a hole in it, both together form a composite feature called a boss. The boss feature may result in a sink defect if the part were to be injection molded [38][39]. The magnitude of the thickness of the sink depends upon the combination of the cylinder diameter and the hole diameter.

1.4 Feature retrieval methods

The above type of feature information can be obtained through two techniques: (i) design-with-features [69] and (ii) feature recognition [46].

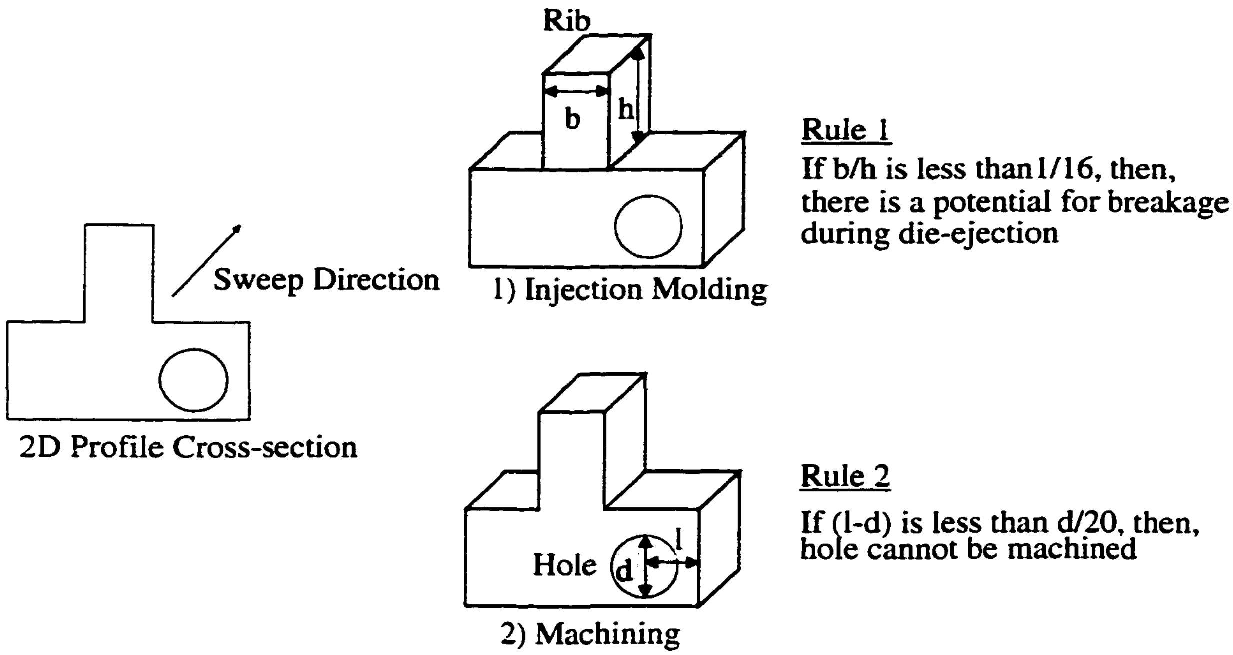


Figure 1-6: Example DFM Rules for a part

The design-with-features approach involves constructing the object with predefined features [9][13][66]. A designer selects features from a features library and creates the geometry using these features. The shape features required to analyze the design, e.g., for manufacturability, are thus readily available in the CAD database. However, this approach has several drawbacks. First, the designer creates part shapes using design features and a one-to-one mapping between design features and manufacturing features[1] may not exist. For example, the shape in Figure 1-6 can be created in a CAD system such as ProEngineer® by

[1] While arguments have been made to allow a designer to create shapes in terms of manufacturing features, this approach is found to restrict the designer's creativity.

sweeping a T-shaped cross-section with a hole in it, however, the manufacturing features for injection molding or machining do not correspond to the swept feature.

Second, the selected features may interact with other features and form new ones, which subsequently must be determined for a DFM analysis. Therefore, while design features are useful for generating the model shape, ensuring that manufacturing features are available requires a further step of feature recognition. Thus, an alternative approach to ensure the availability of features is to explicitly recognize features from the geometric model of the object. The features to be extracted are present in a features library and the feature recognition system (FRS) uses the definitions in the features library to recognize features. Generally, the definition of features in a features library depends on the type of the FRS. If the FRS is a rule-based system then the features are defined in terms of rules [33]. If the FRS is a graph-based system then the features are defined in terms of graphs [42]. The rules and graphs are usually in terms of Boundary Representation (B-Rep) entities such as faces, edges and vertices.

In both the methods (design-with-features and feature recognition) features need to be defined and represented before they can be created or recognized. Features are defined using either a special-purpose feature definition language or a programming language. However, in both these methods of feature definition, the type of features required for design/recognition is dependent on the manufacturing application. That is, different manufacturing applications require different sets of features. The next section presents the existing literature in Feature Definition, Feature-based Design and Feature Recognition.

1.5 Literature Survey

1.5.1 Feature Definition

In general, there are two main categories of feature definition: a) Language-based feature definition, and b) Interactive feature definition.

1.5.1.1 Language-based feature definition

In language-based feature definition, using an ordinary programming language or a special-purpose feature definition language. For example, if the language used is C++ then a feature definition is a C++ class including the methods in the class. One disadvantage of this approach is that the addition of a new feature definition requires compilation and linking with the existing software system. Moreover, the features defined in terms of the syntax of an implementation language may represent a non-intuitive way of defining features. In the special-purpose language-based feature definition, feature definitions are interpreted and coupled with the feature-based system or the feature recognition system.

The ASU features testbed [64][73] uses a feature definition language that is interpreted and parsed to obtain feature definitions in standard C language. In this system, executing the feature definition procedures in the C language creates new features. Sreevalsan [84] developed an interpreted language for feature definition in a feature recognition system as part of the ASU features testbed. The definitions in the language are interpreted into C-language functions and the functions are executed to recognize the features.

In the Helsinki University of Technology's HutCAPP feature modeler [54] a LISP-based feature definition language is used to interface with a C-based geometric modeling system GWB (Geometric Work Bench) [55]. The feature definitions include a reference to the GWB procedures that must be executed during feature creation.

Laakko and Mäntylä [47][48][49] describe a Modeling System called EXTDesign that also uses a LISP-based feature definition language for feature definition. A feature definition contains topological and geometric constraints and rules that the instances of the feature must satisfy. Their feature definitions are used for both feature-based design and feature recognition.

Krause, *et al.* [44] use a special-purpose feature definition language in their IMPPACT feature modeler. They use a textual feature definition language called PDGL (Part Design Graph Language) to create new feature definitions. These new feature definitions are then loaded by the system for feature creation in a feature-based system.

A language called GeoNode Modeling Language is used by van Emmerik [89][90] to define features in a feature-based system. In GeoNode, several types of geometric constraints such as parallelism and distance are specified in a feature definition. A tree of local coordinate systems is constructed from all the feature definitions. This tree is traversed to position and dimension the features after their creation.

1.5.1.2 Interactive feature definition

In Interactive feature definition, features are defined through user interaction with a graphics system. Salomons [68] describes a system called FROOM in which features are defined interactively by drawing a "conceptual graph" that includes the topological elements (faces) and constraints between them. Ranta et al., [60] describe a system where all feature instances in an existing model serve as templates for creating additional feature instances. New feature instances are created by cutting features from existing designs and pasting the cut features onto the new design. The process of cutting and pasting includes the transfer of geometric constraints from old to new designs.

An important point that should be noted here is that in all the systems mentioned above, features are defined in terms of the lowest level the topological elements (faces, edges and vertices) and in terms of geometric constraints between these topological entities.

After features are defined, the feature definitions are used to create features (in a feature-based system) or recognize features (in a feature recognition system). The next two sections presents the existing research in feature-based design and feature recognition.

1.5.2 Feature -based design

The Feature-based design (or design-with-features) approach involves constructing the object with predefined features [13][66]. Design-with-features techniques can be classified into three groups [70]:

1. Feature databases unassociated with solid models
2. Destructive modeling with features
3. Synthesis by features

The first method is a primitive design technique and it is mainly used to input product data for process planning. There is no geometric solid model corresponding to the feature model. In the second method, a part is created by the Boolean subtraction of features from a base stock. Features invariably correspond to volumes removable via machining. Hence, this is mainly applied for design and manufacture of machined parts. Most CAD Systems (I-DEAS from SDRC and ProEngineer® from Parametric Technology Corporation) support this approach to design. Turner and Anderson [87], Cutkosky *et al.* [10] and Mäntylä *et al.* [58] also describe systems using destructive modeling with features.

In the synthesis by features method both Boolean addition and subtraction are performed using features without a need for a base stock volume. Generic features are defined in terms of rules or procedures and features stored in libraries may be application (casting, injection molding, etc.) oriented. Some of the systems employing this method of feature creation are found in [72][65][63].

The design-with-features method has an advantage in that the manufacturability knowledge is readily present in a design. A model can be evaluated for manufacturability immediately after or during design. However, the manufacturability knowledge is most often restricted to the design-intended manufacturing domains. For example, a design intended for machining cannot be evaluated for injection molding. Hence there is a need for feature recognition also.

1.5.3 Feature Recognition

Feature Recognition from solid models can be categorized into two general classes: Volume-based and boundary representation (B-Rep) based feature recognition [71].

1.5.3.1 Volume-based feature recognition

Volume-based methods operate on either constructive solid geometry models, like CSG tree, or they produce and classify volume features from boundary models. CSG tree based recognition works by casting the CSG trees in some canonical form and matching sub-trees. Features are recognized from boundary representation models by subtracting the original object from the convex hull of the object (convex hull decomposition).

Woo [94] developed a Volume-based technique Alternating Sum of Volumes (ASV) Decomposition. Subtracting the volume from its convex hull decomposes the volume. The process is repeated for all resulting volumes until a null object is obtained. However, the ASV decomposition is non-convergent in some cases. As a solution, Kim and Wilde [42] combine ASV decomposition with remedial partitioning and obtain a convergent convex decomposition technique called Alternating Sum of Volumes with Partitioning (ASVP). Currently, ASVP decomposition can be applied only to solids that do not have non-linear entities because it is difficult to obtain the convex hull for objects with non-linear entities.

Wang [92] developed a modified volume decomposition approach in which swept feature volumes for machining applications are built with heuristics and rules. A raw material block is obtained by the addition of a series of feature volumes to the finished part.

1.5.3.2 Boundary Representation-based feature recognition

Unlike Volume-based approaches, B-Rep based approaches operate on the boundary models and use geometric and topological relations between boundary entities to find a match for pre-defined features. For each feature, the geometric and topological requirements are identified. Topological criteria for feature recognition include the number of topological entities that the feature must contain and their adjacencies. To find features, the boundary model is searched to see if the conditions corresponding to each feature are satisfied. The boundary-based approaches can be further classified into rule based, graph based, syntactic and hybrid methods.

Rule-based Methods: In rule-based methods, features are defined as rules. A general rule for a hole that was used by Henderson [33] is:

The hole begins with an entrance face. All subsequent faces of the hole share a common axis. All faces of the hole are sequentially adjacent. The hole terminates with a valid hole bottom.

However, the topological criteria alone are not sufficient to recognize features. Topological criteria are combined with material and geometric properties to recognize features to a satisfactory degree. The material properties indicate the presence or absence of material within the volume enclosed by the topological entities of a feature. Features are defined using production rules [34][35] such as,

IF (topological conditions & geometric conditions &
material conditions) THEN (shape is feature_x)

A set of rules is defined for each feature that is to be recognized. Each of the conditions in the rules is tested against the model for the recognition of the feature. However, the determination of rules for any given feature is a process of trial and error. Moreover, rule based systems perform exhaustive searches of the model database in order to find features of interest.

Gadh [23][24][25][26][27] has developed a rule-based method that overcomes the search complexity problem that is associated with rule-based methods. Gadh's method uses the notion of concave and convex loops to define a broad class of features. A Differential Depth Filter, which has a reduced search complexity, is used to determine the concave and convex loops in a model.

Graph-based methods: In graph-based methods, graphs in the solid model are matched with graphs representing the features to be recognized. Face Adjacency Graphs (FAG) are usually used to represent features. In a Face Adjacency Graph, each face is represented as a node and two nodes are connected if and only if they are adjacent faces in the solid model. Geometry information like the nature of the face is stored in the nodes and geometry information about the edges in the model is stored in the connecting arcs.

DeFloriani and Bruzzone [11] designed a graph-based feature recognition method based on FAGs for extracting certain classes of an object's shape features from a relational boundary model. They have a Generalized-Edge-Face Graph (GEFG) that describes objects with multiple shells and multiple-connected faces. Feature recognition and classification methods partition the GEFG into sub-graphs corresponding to the bi-connected and tri-connected components of the associated face-edge graph. The feature recognition process produces an object decomposition graph that provides an unambiguous description of the global shape of the object.

Joshi and Chang [41] represent parts and definitions of features by an Attributed Face Adjacency Graph (AFAG). An attribute value is assigned to each arc of the AFAG depending on the edge convexity or concavity. The graphs are stored as adjacency matrices. The novelty of their approach is that they propose heuristic rules to handle feature interaction. They consider, what they call, type I and type II feature interactions in their analysis. Type I feature interactions have edges as the boundaries of interaction and type II feature interactions have faces as boundaries of interaction.

Sakurai and Gossard [67] use a dual representation (CSG tree and B-Rep) solid modeling system for their feature recognition research. In their method a shape feature is defined as a single face or a set of contiguous features possessing certain characteristic facts in topology and geometry. Once a feature graph is matched, a volume corresponding to the feature is constructed by entity growing [14] and the feature is subsequently removed from the solid model.

Lentz and Sowerby [50] present a feature recognition approach from sheet-metal parts. They consider the properties of general concave and convex regions (which are obtained using curvatures on the surface) of a sheet metal component to extract features. They use a face adjacency hyper-graph representation for the various convex and concave regions. However, this approach is restricted by the assumption that all intersections of trimmed surfaces have to be C^1 continuous. Furthermore, their approach is highly specific to feature recognition in sheet-metal parts.

Other significant researches that use graph-based methods for feature recognition are [12][56][5][7][8][19][18][20][22].

Syntactic methods: Syntactic methods use geometric patterns typically described by a series of straight, circular or more complex curved line segments. The model is first expressed in terms of suitable primitives; lines and arcs, for example, in 2D-feature recognition. The feature patterns are also expressed in terms of the primitive elements. A symbolic encoding of the patterns is used; hence pattern (feature) recognition can be done using syntactic means.

Jakubowski [40] and Staley, et al. [85] use syntactic pattern recognition to recognize 2D profiles of holes. Choi, et al. [5] use syntactic pattern recognition in three dimensions by defining their primitives to be surfaces. For example, the syntactic elements used for recognizing holes are HSS (Hole-starting surface), HES (Hole-element surface) and HBS (Hole-bottom surface). They construct hole features from these primitives and represent them symbolically for syntactic recognition. A drawback with their method is the lack of geometric constraint specification between the HSS, HES and HBS. Due to this lack of constraint, there is a possibility that a cylindrical protrusion is erroneously recognized as a hole.

Hybrid methods: There has also been some research into hybrid approaches that blend graph-based and rule-based approaches. Recognition of detailed features in graph-based methods is a problem due to the difficulty in constructing a graph for detailed features. This problem is overcome by combining rule-based and graph-based methods. With the application of graph-based methods general features are obtained and with a further application of rule-based methods more specialized features are obtained. Vandenbrande and Requicha [88] have developed a hybrid system for the recognition of machinable features in a solid model. It automatically produces feature removal volumes and representations for feature interactions. Regli and Nau [61] developed another hybrid system to recognize defined classes of machinable features.

Boundary-based methods are not robust when feature interactions are present. Feature interactions alter the topology, which is the basis for forming the rules, graphs or algebraic expressions. Thus, such methods cannot recognize features exactly when feature variations are present.

1.6 Problem Statement and Overview of current approach

1.6.1 Limitations of the existing approaches for Feature Definition

In the feature definition methods described in the literature survey, the most common way of defining features is in terms of B-Rep elements (Faces, Edges and Vertices) and geometric constraint specification on the attributes (angle at an edge, length of an edge, etc.). The required feature information is obtained by querying the topology and geometry on the CAD model. However, there are several drawbacks when features are defined and recognized in terms of only the B-Rep entities - faces, edges and vertices, and these are as follows:

1. Feature definition is tedious.

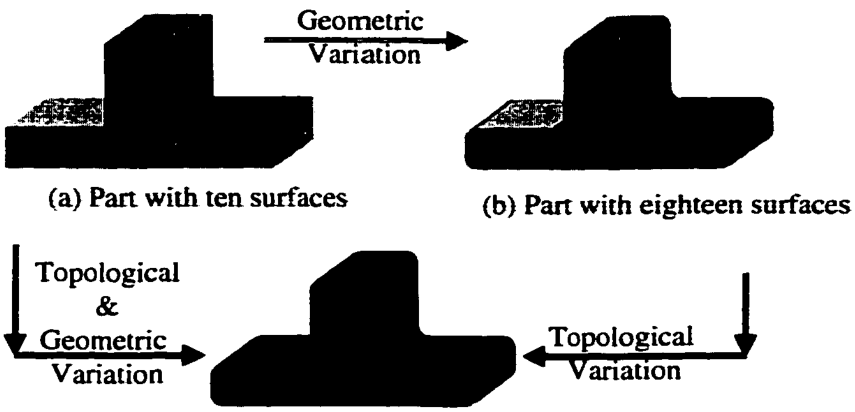
2. Feature definition is too detailed and therefore the feature database becomes very large.

3. As a result of the high detail in feature definitions, feature recognition algorithms become computationally expensive [59].

1.6.2 Limitations of the existing approaches for Feature Recognition

The existing volume-based approaches require the determination of the convex hull or the delta volumes of an object [17][61][91]. Determination of the convex hull or the delta volumes of a model with non-linear surfaces is a computationally expensive process and volume-based approaches have not been extended to models with curved surfaces. A significant number of approaches to feature recognition are B-Rep surface based. Some of the limitations of surface-based approaches are as follows:

- Rule-based and Graph-based approaches search for exact patterns of topology and geometry to determine features. For example, Figure 1-7 shows minor geometric variations of a solid model with different topologies. Figure 1-7(a) shows a part with ten surfaces. Figure 1-7(b) shows the part with some model edges being rounded/filleted. This creates additional eight surfaces, resulting in a total of eighteen surfaces on the model. Figure 1-7(c) shows another topological representation of the part in Figure 1-7(b) with only three surfaces. A Rule-based approach to find features requires different rules to be formulated to handle each case (the same is true for a Graph-based approach). Therefore, new rules/patterns are required for each topological instance of a feature. This could result in a large and potentially unmanageable number of rules/patterns.



(c) Part with three surfaces

Figure 1-7: Topological variations with minor geometric variations

- Another aspect of most existing approaches is their primary focus on form feature extraction for a single application domain. For example, the feature recognition approaches developed by Regli and Nau [61] and Vandenbrande and Requicha [88] apply primarily to the machining domain. Likewise, the approach developed by Lentz and Sowerby [50] is specific to sheet-metal parts.

1.6.3 Overview of the current research

Overcoming the limitations of existing systems: The current research presents a methodology whereby features can, firstly, be defined interactively and easily. Secondly, a user can choose to make the feature definition as detailed or as generic as required. Thirdly, a user has a choice to define features for any application domain, such as, machining or injection molding. Subsequent to their definition, the features can be recognized from a part. Based on the type of feature definition, the feature recognition algorithms allow topological and geometric variations of a feature to be recognized with a single feature definition. Moreover, features can be extracted for multiple extraction domains without varying the feature recognition algorithms.

Solution for overcoming the limitations of existing systems: A framework of multiple levels of geometric shape abstractions has been developed for use in the current research to define and therefore to ultimately extract features specific to manufacturing applications, such as, injection molding and machining. The detail of information in the different abstractions decreases with the increase in the level of abstraction [30] [36] [86]. This decrease in detail allows topological and geometric variations and in features to be ignored during feature recognition. A graphical front-end has also been developed to allow the user to define features interactively [83].

An illustration of the current approach (Curvature Region Aided Feature ExTraction System - CRAFTS) is shown in Figure 1-8.

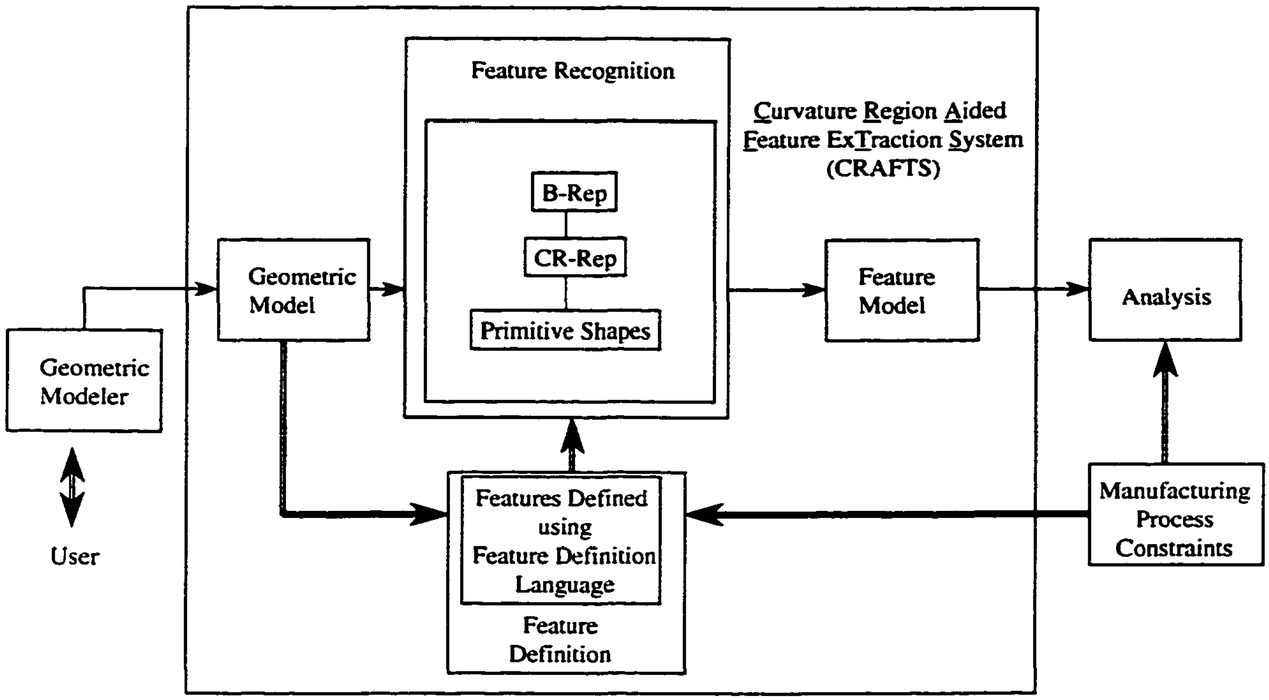


Figure 1-8: Overview of CRAFTS

The input to the system is the B-Rep of an object and the output is the feature information; which is obtained using a feature recognition algorithm. As mentioned earlier the two major parts in the system are feature definition and feature recognition. Three geometric abstractions, namely, B-Rep, Curvature Region and Primitive Shape are utilized in the current research [28][45][77] [78][79][80][81]. A feature is defined either as a B-Rep Graph, a CR-Rep Graph or as a Primitive Shape. In addition, constraints are specified on the shape parameters of the feature. The Feature Definition Language is in terms of, (a) the entities in the above abstractions, (b) attributes (such as length, angle at an edge, diameter, etc) of features and (c) constraints (such as less-than, greater-than, perpendicular-to, etc) on the attributes.

After a feature is defined, feature extraction is performed based on the type of feature definition. For example, if a boss feature definition is in terms of Curvature Regions then a graph match is performed between the CR-Graph of the boss and the CR-Graph of the part. A successful graph match results in the determination of the defined feature.

1.7 Thesis Organization

Chapter 2 presents a description of the geometric abstractions and the approach used to obtain them from a model. Chapter 3 presents the Feature Definition Language that is used to represent features. The Feature Definition Language is in terms of the abstractions in Chapter 2. A description of the user interface to the Feature Definition Language is presented in Chapter 4. In Chapter 5, the feature recognition algorithms that are utilized to obtain the features are described. Chapter 6 presents sample results of feature definition and feature recognition. In Chapter 7, the conclusions are presented and some future research directions are suggested.

2 Geometric Abstractions

In the current research, a geometric abstraction is a perception of the geometry, which assists the analysis of a design against the requirements. Three levels of geometric abstractions are used in the current research, (1) Boundary Representation (B-Rep) Elements, (2) Curvature Regions, and (3) Primitive Shapes (Protrusions and Depressions). Each level of abstraction represents a physical aspect of an object, which can be visualized and used to reason about the object. In the current research, it is assumed that the native geometric and topological representation of a model is the B-Rep. The Curvature Region and Primitive Shape abstractions are derived from a model by performing geometric and topological queries on the model. The three geometric abstractions are described in detail in the next few sections.

2.1 Boundary Representation (B-Rep)

In this abstraction, a model is represented in terms of the geometry and topology of faces, edges and vertices. This representation is typically used in most CAD systems such as ProEngineer®, SDRC-IDEAS and UniGraphics. For example, a cube is represented by its faces (F_1, F_2, F_3, etc.), edges (E_1, E_2, etc.) and vertices (V_1, V_2, etc.) (Figure 2-1). The geometry of the faces and edges is in terms of the equations of surfaces and curves (planes and lines for the cube). The faces are topologically connected to model edges, which in turn are connected to model vertices. This is the lowest abstract level of representation, which is the generic standard for storing and querying shapes in most existing CAD systems.

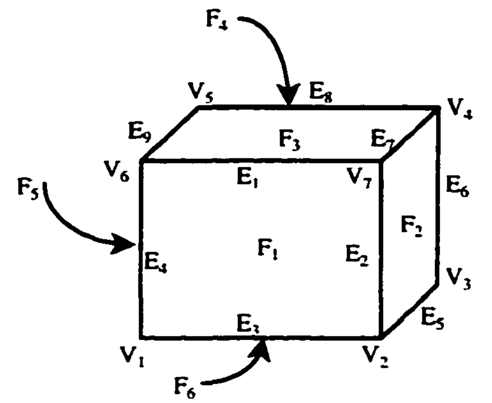


Figure 2-1: Boundary Representation of a cube.

Analysis on a part that is represented as a B-Rep model is usually performed by querying the topology and evaluating the geometric attributes of the part. Some examples of geometric attributes are:

(a) tangent at a point on an entity (edge/face),

(b) tangent direction of a linear edge, normal at a point on an entity face,

(c) dot product of a normal at point and a vector direction, and

(d) curvature at a point on a surface.

Based on how they are evaluated, geometric attributes can be classified either as: intrinsic attributes and extrinsic attributes. The evaluation of intrinsic attributes is dependent only on the local geometry. The evaluation of extrinsic attributes is based on the interaction between local geometry and external geometric constraints. An example of an intrinsic local geometric attribute is the principal curvature, $\mathbf{C_p}$, at a point on a surface. For a point on a surface, there are two types of curvature: geodesic and normal [52], [53]. The normal curvature at a point on a surface varies with the tangential direction on the surface. The maximum and minimum values of the curvature at a given point are referred to as the principal curvatures. The two principal curvature directions are always perpendicular to each other, except at umbilical points [57], [21], [37].

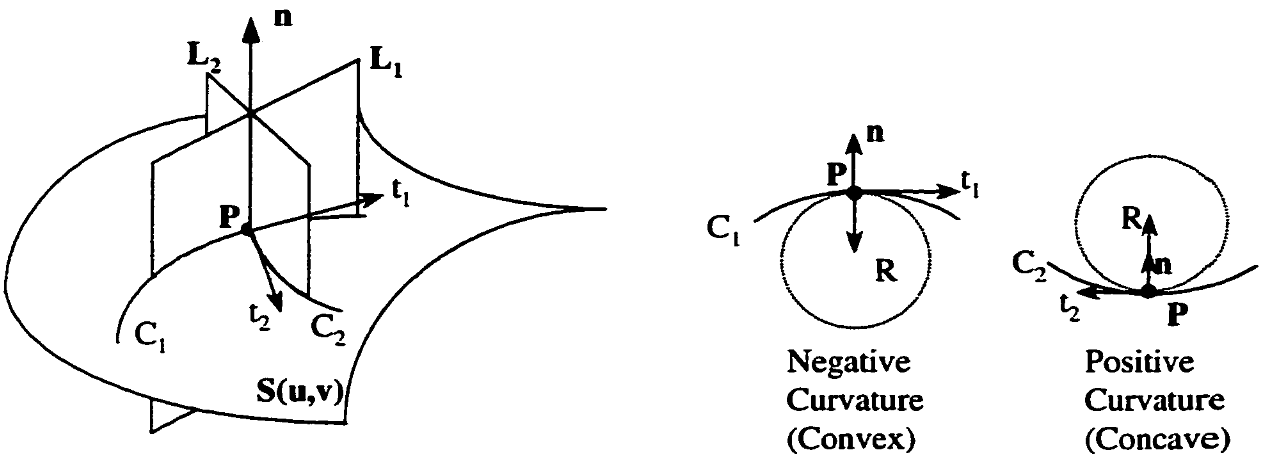


Figure 2-2: Principal directions of curvature at a point P

Figure 2-2 shows a surface and the surface tangent directions, t_1 and t_2 of the principal curvatures at the point P. A principal curvature at P is defined as positive (+) or concave if the radius vector from P to the center of the circle corresponding to the radius of curvature is along the direction of the normal at the point. Similarly, a principal curvature at P is defined to be negative (-) or convex if the radius vector from P to the center of the circle corresponding to the radius of curvature is against the direction of the normal at the point. When the radius of curvature is infinity, the resultant curvature is zero (0). The curvature type τ at a point P is designated by $[s_1, s_2]$ where s_1 and s_2 are signs of principal curvatures, κ_1 and κ_2 such that ($\kappa_1 > \kappa_2$). For example, the point P shown in Figure 2-2 has $\tau = [+,-]$. If κ_1 is zero then s_1 is 0; similarly, if κ_2 is zero then s_2 is 0. The principal curvature attribute is used to define the Curvature Region Abstraction in the next section.

Another example of an intrinsic local geometric attribute is the angle at a point on an edge, measured at the point from inside the solid model. Based on this angle the edges of a B-Rep model are classified into three types: convex, concave and neutral. If the angle between the two adjacent surfaces of an edge, measured from inside the surface, is less than 180°, then the edge is convex. However, if the angle is greater than 180°, it is concave. If the angle equals 180°, then the edge is neutral. Edges of these three types are shown in Figure 2-3.

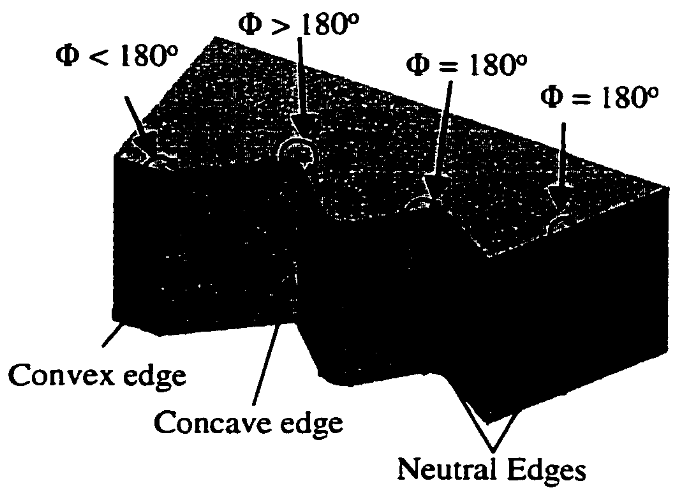


Figure 2-3: Convex, concave and neutral edge types

An example of an extrinsic local geometric attribute is the dot product ($\vec{N} \cdot \vec{V}$) of the normal ($\vec{N}$) and a given vector ($\vec{V}$). This attribute can be used to determine the silhouette of a model [15]. A silhouette of an object is the outline of the object that is seen when it is viewed along a particular direction (called the view direction). The silhouette of a model can be used to determine the parting line of a model [29][80] or for recognizing form features in a model [75][76][24][27].

In the current research, a B-Rep-Graph is constructed from the B-Rep data of a model. The B-Rep-Graph is used for feature definition and feature recognition.

2.1.1 Boundary Representation Graph (B-Rep-Graph)

The B-Rep-Graph is constructed by querying the topology and evaluating the geometric attributes of the model. The nodes in a B-Rep-Graph are called BR-Nodes (shown in Figure 2-4). In the B-Rep-Graph, there is a node corresponding to each face and edge in the model. In the current research, the vertices of the model are not considered in the B-Rep-Graph. Each BR-Node contains information about:

a) the type of node (Face Node or Edge Node),

b) the list of neighbors of the node,

c) the id of the B-Rep entity (edge or face),

d) the edge type (linear, arc, etc) or the face type (planar, cylindrical, etc), and

e) the edge nature (convex, concave or neutral) if it is an Edge Node.

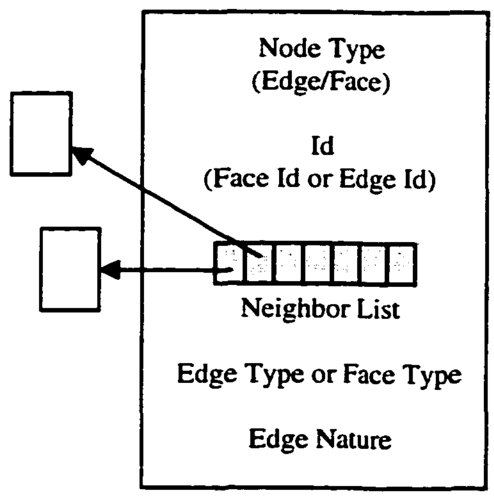


Figure 2-4: BR-Node

The number of entities in the B-Rep-Graph is equal to the number of faces and edges in the B-Rep model.

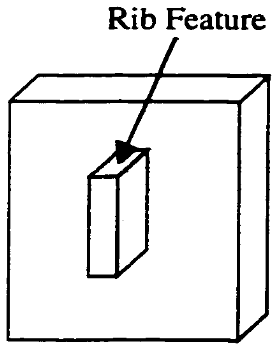
2.1.1.1 Algorithm for obtaining the B-Rep-Graph

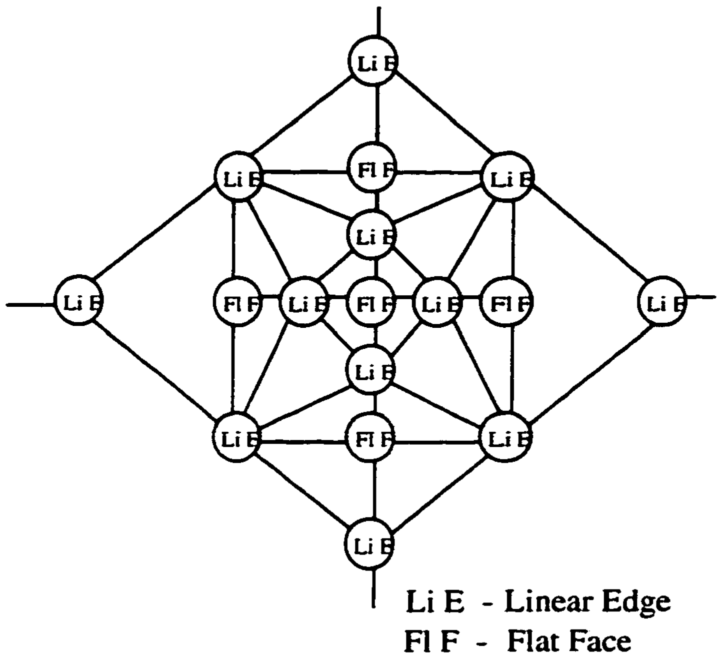
The algorithm for obtaining the B-Rep-Graph from the B-Rep model of a part is as follows:

```
For each face F in the model{
        Get (Create if not there) BR-Node FNode for the face F
        For each edge E in face F{
                Get (Create if not there) BR-Node ENode for the edge E
                Add ENode to neighbor list of FNode
                Add FNode to neighbor list of ENode
                Add ENode to neighbor lists of nodes of adjacent edges of E
                Add nodes of adjacent edges of E to neighbor list of ENode
        } // End edge for loop
} // End face for loop
```

Algorithm 2-1: Obtaining the B-Rep-Graph

After this algorithm is executed on a model the B-Rep-Graph of the model is created. For example, for the part shown in Figure 2-5(a), the partial B-Rep-Graph comprising the faces and edges of the rib feature is shown in Figure 2-5(b). The time complexity for evaluating the B-Rep-Graph is $O(n_F + n_E)$, where n_F is the number of faces in the model and n_E is the number of edges in the model.





(a) B-Rep model (b) Partial B-Rep-Graph

Figure 2-5: Partial B-Rep-Graph comprising the faces and edges of the rib feature

2.2 Curvature Region Abstraction

An important local aspect of a shape is curvature. For example, the deviation of a curve from a straight line is obtained through its curvature. Similarly, the curvature of a surface gives the deviation of the surface from a plane. The shape information necessary for reasoning about an object can be obtained from the curvature of the object - i.e., curvatures of the points on the object. Therefore, the curvature properties at points on a model are used to define an abstraction (called Curvature Region) that is later used for feature definition and extraction. The Curvature Region Abstraction is defined in terms of an aggregate geometric abstraction.

Definition: An **aggregate geometric abstraction** is an abstraction that comprises a group of entities that as an aggregate share a common geometric attribute.

For example, consider faces F_1, F_3, F_4 and F_6 of the cube shown in Figure 2-1. The normals of these four faces (intrinsic geometric attributes) are perpendicular to the normal of face F_2. The faces F_1, F_3, F_4 and F_6 along with the attribute of perpendicularity of the normals to a common direction constitute an aggregate geometric abstraction. An aggregate geometric abstraction is an aggregation of two or more points/entities with the same geometric attributes.

Definition: A Curvature Region (CR) is an aggregate geometric abstraction in which all the points have the same sign for s_1 and s_2, where s_1 is the sign of the maximum principal curvature (or 0 if the maximum curvature is zero) and s_2 is the sign of the minimum principal curvature (or 0 if the minimum curvature is zero).

A Curvature Region is characterized by its curvature type τ (= $[s_1, s_2]$). Any object can be divided into regions of points with identical curvature types, i.e., regions which are [+,+] (concave regions), [-,-] (convex regions), [+,-] (saddle-like regions), [+,0] (concave regions), [0,-] (convex regions), or [0,0] (flat regions). The six different types of Curvature Regions are pictorially shown in Figure 2-6.

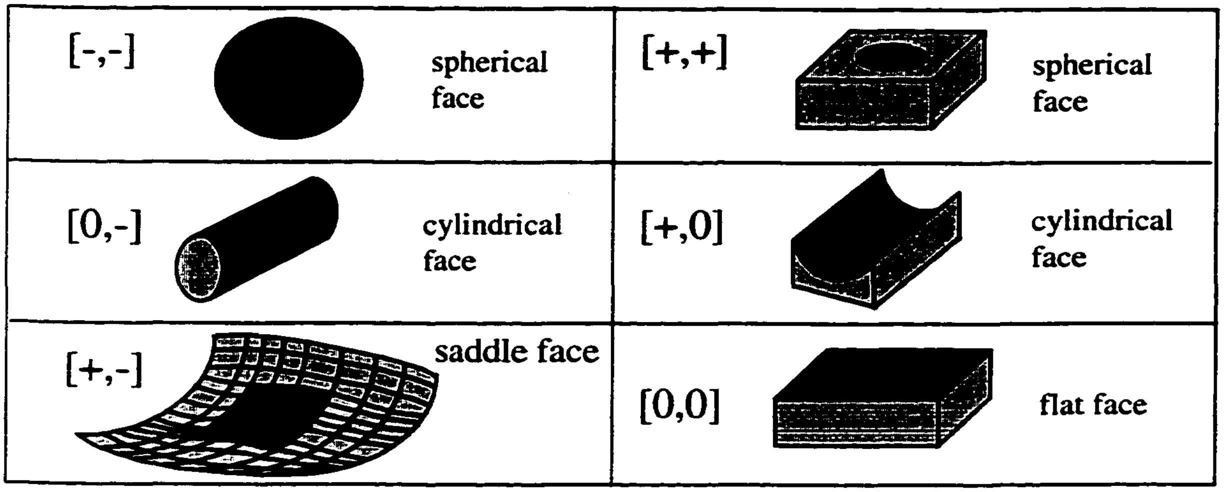


Figure 2-6: Six types of Curvature Regions.

2.2.1 Obtaining Curvature Regions for Edges, Vertices and Faces

Curvature Regions (CRs) on a model could be determined by evaluating the principal curvatures at all points on the model surfaces. However, the curvature of an object resides in edges and vertices also [43], as a result, all the entities (faces, edges and vertices) in a model must be mapped to CRs. However, the edges and vertices a model form C^1 discontinuities and the curvatures at points on edges and vertices are indeterminable. Therefore, an alternate method is used to determine the CRs at edges and vertices.

Since the convexities and concavities, i.e., the signs of the principal curvatures, alone are of interest, a sharp edge/vertex and a smooth edge/vertex (called virtual surface [96][97]) with infinitesimal radius are considered equivalent. This equivalence is utilized in determining the CRs of edges and vertices.

The following sections discuss the mapping of faces and face boundaries, i.e., edges and vertices, in the B-Rep, to the CR-Rep.

2.2.1.1 Obtaining Curvature Regions for Edges

To determine the CRs on the edges of an object, the non-neutral edges of the model are considered equivalent to virtual surfaces that are C^2 continuous. Since neutral edges[2] do not form discontinuities on the model's surface, they are not considered during the determination of the Curvature Region abstraction. Figure 2-7 shows a non-neutral edge **E** and its equivalent virtual surface $\mathbf{E_S}$.

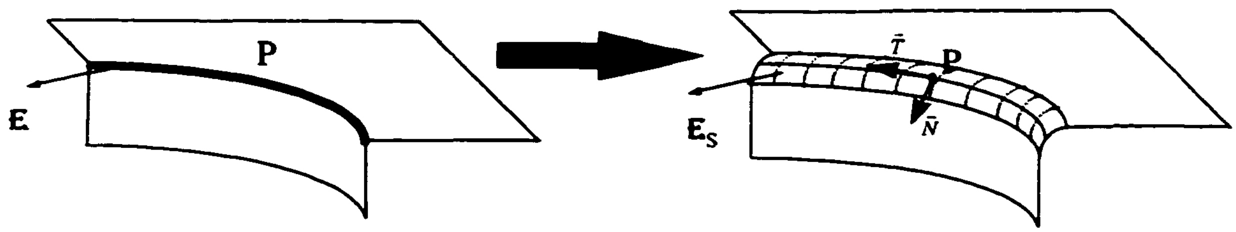


Figure 2-7: Virtual surface of an edge

One approach to determine CRs on the virtual surface $\mathbf{E_S}$ is by computing the sign of the principal curvatures at every point on $\mathbf{E_S}$ and combining identical Curvature Region points that are adjacent to each other. However, this approach is computationally expensive since it involves: 1) creating the equivalent surface, 2) evaluating the curvatures at all the points on the equivalent surface and 3) mapping the curvatures back to the edge. To overcome this problem, an alternate method is used in which the equivalent surfaces of sharp edges are not actually created. Instead, the equivalence between a virtual surface and an edge is utilized only in developing algorithms to evaluate the curvature signs on edges. The same principle is used to evaluate the curvature signs on the vertices also.

[2] Neutral edges are those edges that are neither concave nor convex, i.e., there is no change in the direction of surface normal across them.

The alternate method that is used to determine the CRs of an edge considers the tangent and normal curvatures on the virtual surface $\mathbf{E_S}$. The tangent curvature, $\mathbf{k_T}$, at a point P on $\mathbf{E_S}$ is the curvature at P such that the curvature direction is along the tangent direction at point P on **E** (illustrated as direction $\bar{T}$ in Figure 2-7). The normal curvature, $\mathbf{k_N}$, at a point P on $\mathbf{E_S}$ is the curvature at P along a direction perpendicular to $\bar{T}$ (shown as direction $\bar{N}$ in Figure 2-7). Assuming that the radius of rounding r is infinitesimal, at all points on $\mathbf{E_S}$, $\mathbf{k}_N\left(=\frac{1}{r}\right)\rightarrow\pm\infty$. This implies that $\mathbf{k_N}$ is one of the two principal curvatures of $\mathbf{E_S}$ (since it must be either maximum or minimum). The other principal curvature lies in a direction perpendicular to the normal direction, and is therefore $\mathbf{k_T}$. Therefore, the principal curvatures on $\mathbf{E_S}$ are equivalent to the tangent and normal curvatures. The signs of the principal curvatures (which are required to determine the CRs) at any point on $\mathbf{E_S}$ are thus equal to the signs of $\mathbf{k_T}$ and $\mathbf{k_N}$ at that point. The curvature type, τ, at any point on the virtual surface $\mathbf{E_S}$ of an edge **E** is obtained as:

$$\tau = [n, t], \text{ where, } n = \text{sign of } \mathbf{k_N} \text{ and } t = \text{sign of } \mathbf{k_T} \qquad \text{Eq. 1}$$

Since the equivalent surface $\mathbf{E_S}$ is not created, τ is evaluated on the adjacent faces of edge **E**. Therefore, there are two τ values corresponding to each edge; one per each adjacent face. To compute the CRs, the edges of a model are classified into the following categories:

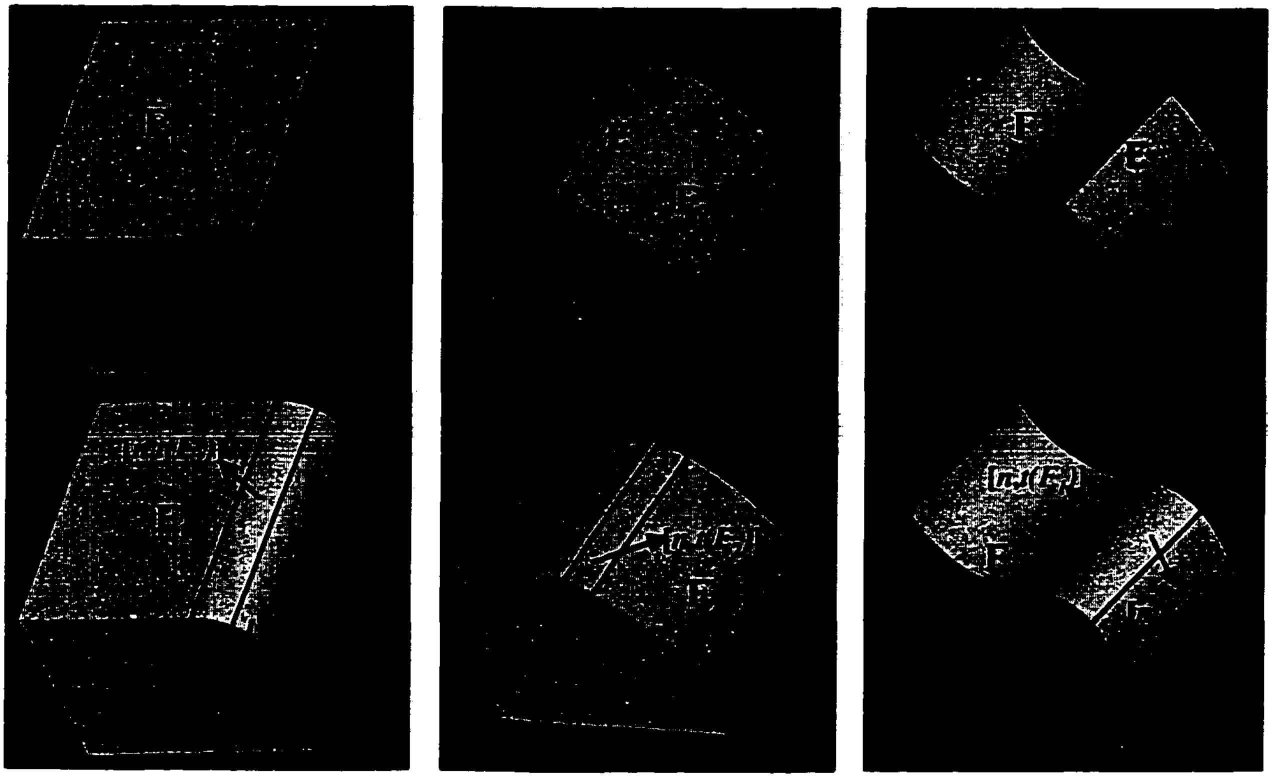
<u>Category 1: Edge having two adjacent planar faces</u> Figure 2-8(a) shows a linear edge **E** that is adjacent to two planar faces (F_1 and F_2) and its virtual surface $\mathbf{E_S}$. n (sign of the normal curvature) depends on the concavity of the edge. n is '+' if the edge is concave and '-

' if the edge is convex. $\mathbf{k}_T$ is 0 because the curvature on the planar faces is zero, hence *t* is 0. In the case of edge **E** in Figure 2-8(a), *n* is '-'. Hence, by (Eq. 1) for all points on $\mathbf{E}_S$, τ = [0,-]. Therefore, the CRs corresponding to **E** are two [0,-] regions.

Category 2: Edge with only one adjacent non-planar face An example of such an edge, **E**, and its virtual surface, $\mathbf{E}_S$, is shown in Figure 2-8(b). As in the previous case, *n* at a point P on the virtual surface is '+' if the edge is concave and '-' if the edge is convex. *t* for each CR is evaluated at P on the non-planar face. In Figure 2-8(b), *t* is '0' and *n* is '-' at all points on $\mathbf{E}_S$, hence τ for the curvature regions corresponding to **E** are [0,-] and [0,-].

Category 3: Edge with two adjacent non-planar faces An example of such an edge, **E**, and its virtual surface, $\mathbf{E}_S$, is shown in Figure 2-8(c). As in the previous case, *n* at a point P on the virtual surface is '+' if the edge is concave and '-' if the edge is convex. *t* for each CR is evaluated at P on the corresponding non-planar face. In Figure 2-8(c), *t* is '0' on both the non-planar faces and *n* is '-' at all points on $\mathbf{E}_S$, hence τ for the CRs corresponding to **E** are [0,-] and [0,-].

The curvature type along an edge can vary if, (i) *t* changes value along the edge, and/or, (ii) *n* changes value along the edge. Since *t* and *n* are independent of each other, they can vary simultaneously along an edge, and if they do, the edge consists of more than two CRs.



(a) (b) (c)

n = normal curvature
'+' if edge is concave
'-' if edge is convex

$t(F)$ = tangential curvature of face F

Figure 2-8: Curvature Region for an edge

2.2.1.2 Obtaining Curvature Regions for Vertices

The CR of a vertex is also evaluated by considering the equivalent surface of the vertex. Figure 2-9(a) shows the vertices in a B-Rep model and Figure 2-9(b) shows their equivalent virtual surfaces. It is assumed that the virtual surface, V_S of a vertex V contains a single curvature type.

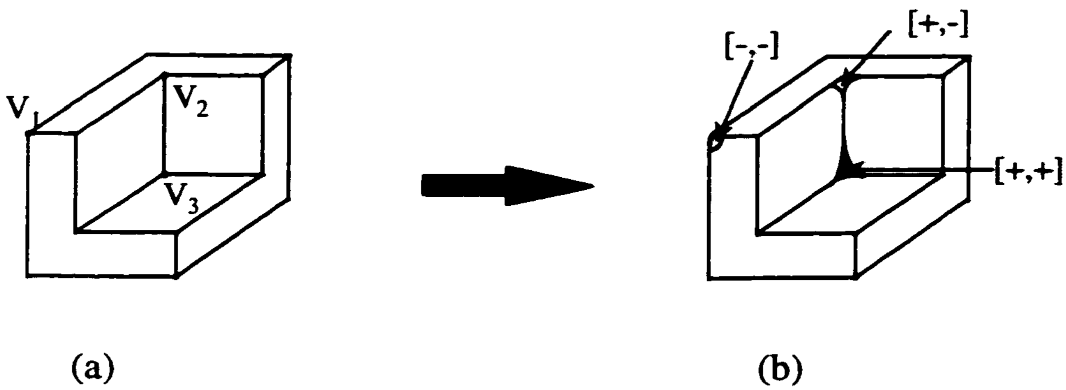


Figure 2-9: Virtual surfaces of vertices

Type	N_{convex}	$N_{concave}$	$N_{neutral}$	κ(V)
1	>0	0	0	[-,-]
2	0	>0	0	[+,+]
3	>0	>0	>=0	[+,-]
4a	<=2	0	>0	Null CR
4b	>2	0	>0	[-,-]
5a	0	<=2	>0	Null CR
5b	0	>2	>0	[+,+]

N_{convex} Number of convex edges incident at V
$N_{concave}$ Number of concave edges incident at V
$N_{neutral}$ Number of neutral edges incident at V
κ(V) Curvature type at Vertex V

Table 2-1: Curvature Type at a vertex V

Similar to the edge CR evaluation, the CR on a vertex is evaluated directly on the vertex without creating the virtual surface. A vertex is assigned a curvature type based on the concavity and convexity of the edges at the vertex. For a vertex V, the curvature type is obtained by mapping κ(V). Table 2-1 defines the mapping κ(V), which is a function of the number of convex, concave and neutral edges at the vertex. The columns N_{convex}, $N_{concave}$ and $N_{neutral}$ correspond to the number of convex, concave and neutral edges incident at V and the last column displays κ(V). The rows in Table 2-1 correspond to the different types (Type 1, Type 2, etc.) of vertices. For example, a vertex of Type 1 (row 1) has one or more incident convex edges and no incident concave or neutral edges and it maps to the curvature type [-,-]. For a vertex of Type 4a or 5a, there is no change in geometry in the immediate neighborhood of the vertex, hence it does not contribute to a CR and therefore maps to a null CR.

Figure 2-10 shows τ values for various types of vertices, using Table 2-1. Vertex **V1** adjoins 3 convex edges, **E1**, **E6** and **E7**. Hence, it is a Type 1 vertex and κ(**V1**) = [-,-]. Vertex **V10** adjoins 3 concave edges, **E16**, **E19** and **E17**. Hence, κ(**V10**) = [+,+] (Type 2 vertex). Vertex **V11** adjoins 2 convex edges **E9** and **E15** and one concave edge **E16**. Hence, κ(**V11**) = [+,-] (Type 3 vertex). Vertex **V8** adjoins two neutral edges and therefore maps to a null CR (Type 4a or 5a vertex). Vertices **V7** and **V9** also map to null CRs since they are of Types 4a and 5a respectively.

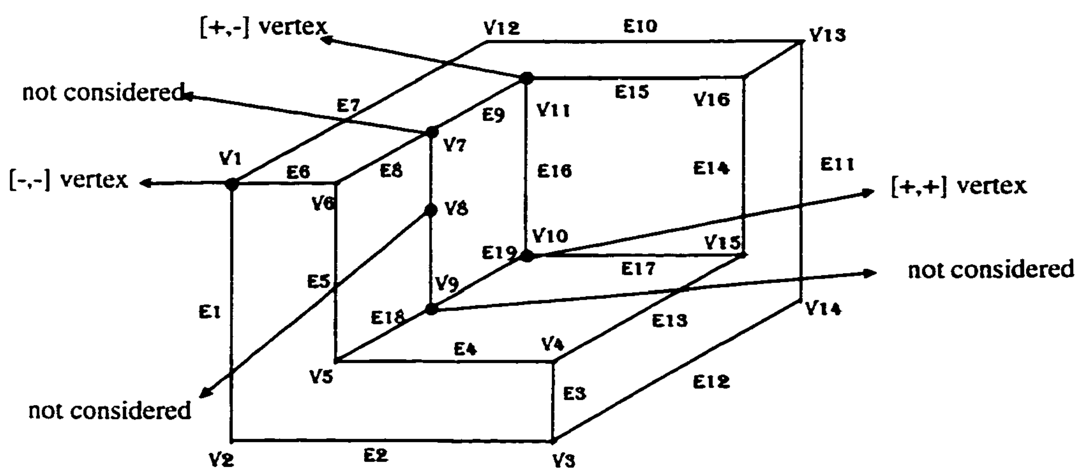


Figure 2-10: Vertices of various curvature types

2.2.1.3 Obtaining Curvature Regions for Faces

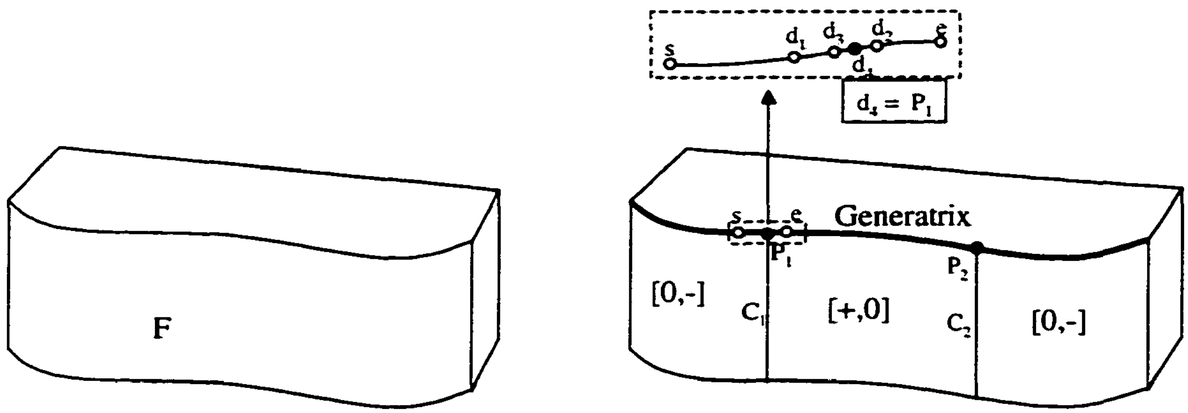
The CRs of a face are evaluated based on the geometry of the face. If the face for which the CR needs to be determined is planar, cylindrical or spherical, the τ value at all points on the face is identical; therefore, obtaining τ at a single point is sufficient. For a planar face, τ = [0,0], and for a cylindrical face τ = [0,-] or [+,0], depending upon the orientation of the face with respect to the solid. Similarly, for a spherical face, τ = [-,-] or [+,+].

Definition: A surface trichotomy is a partition of a surface into three types of regions: convex, concave and saddle [16].

To obtain the CRs for all other types of faces (which are assumed to be curvature continuous), the surface is trichotomized into convex, concave and saddle regions. For a face (F in Figure 2-11) that is generated by sweeping a curve (called generatrix) along a straight line, the curvature regions are found by first determining the curvature types of sample points along the generatrix. Second, determining the points (P_1 and P_2) on the generatrix at which the both the principal curvatures are zero ([0,0] curvature type). Third, sub-dividing the surface using curves (C_1 and C_2) on which all the points have curvature type of [0,0].

To determine [0,0] curvature points on the generatrix, two adjacent sample points with different curvature types are obtained and, subsequently, the interval between the points is recursively sub-divided. Consider the point P_1 as shown in Figure 2-11(b). P_1 is generated after recursively sub-dividing the interval between the points 's' and 'e'. The curvature type at point 's' is [0,-] and that at point 'e' is [+,0]. After the first sub-division, point 'd_1' of curvature type [0,-] is obtained. The next sub-division is performed on the interval between

'd_1' and 'e' since they are of different curvature types. This process is repeated until a point of curvature type [0,0] is obtained. In the above example, the point P_1 is obtained after 4 sub-divisions. Since it is assumed that the surface is curvature continuous, the tangential curvature changes continuously from '-' to '+'. Therefore, there exists a point at which the tangential curvature is zero and the process is guaranteed to terminate.

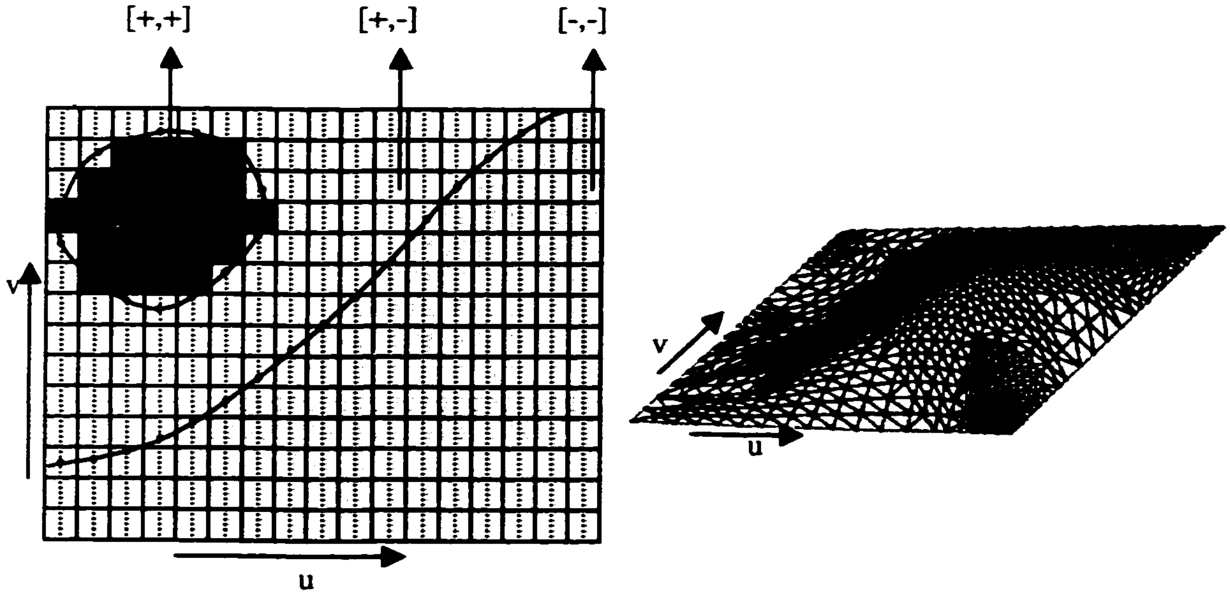


(a) Swept Object

(b) Curvature Regions on a swept surface

Figure 2-11: CRs of a Swept Face

For other non-planar faces, such as Spline and Bezier, a finite number of discrete sample points are chosen in the bi-parametric (u-v) space. The extent of discretization is based on the minimum size of a shape feature in a given manufacturing domain. The curvature type at each discrete point is found by evaluating the signs of the principal curvatures at that point. Points of zero Gaussian curvature (product of the principal curvatures) are obtained by using a recursive sub-division technique similar to the one employed in the above example. The sub-division is performed in both the u and v directions of parameterization.



(a) Discrete sample points on a non-planar surface

(b) Non-planar surface

Figure 2-12: Obtaining Curvature Regions on a surface

A sample uv-grid for a non-linear face, for which curvature types determined at discrete points on the face, is shown in Figure 2-12(a). Each uv-grid represents a curvature region; the curvature type of which is determined at the mid-point of the grid. The zero Gaussian curvature points shown in the figure are generated through a recursive sub-division in the v direction. The initial number of points sampled are chosen such that the features are determined with sufficient precision, which in turn depends upon the minimum feature size for a given application. By sampling an arbitrarily greater number of points on the surface for determining CR-Rep, an arbitrarily high precision is obtained.

The complexity of the algorithm to obtain the Curvature Regions for the edges, vertices and simple surfaces[3] is of the order of the number of entities (number of faces, edges

[3] A simple surface is a surface consisting of a single curvature region.

and vertices) in the B-Rep of the model. The complexity of the algorithm for surfaces that are not simple (such as NURBs) depends on the minimum size of the feature. This is because the size of the feature determines the number of sampled grid points.

Similar to the B-Rep-Graph, a Curvature Region Graph (CR-Graph) is constructed from the Curvatures Region abstraction. The CR-Graph is used for feature definition and feature recognition.

2.2.2 Curvature Region Graph (CR-Graph)

The CRs that are obtained from a model represented as a graph (CR-Graph). A node in the CR-Graph represents a CR, and adjacency between two CRs is represented as arcs between the corresponding nodes. Each node in the CR-Graph contains the following information: (1) τ value for the CR the node represents, (2) the topological entities on the B-Rep model that correspond to the CR, and, 3) the neighboring nodes of the CR-Node.

2.2.2.1 Algorithm for obtaining the CR-Graph

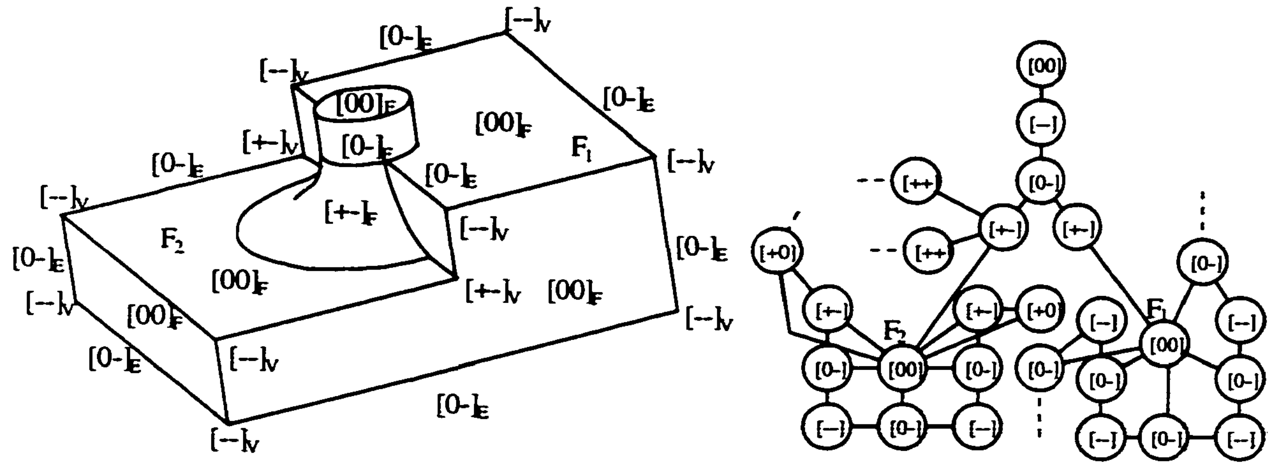
The algorithm for obtaining the CR-Graph is as follows:

```
For each face F in the model{
        Get (Create if not there) CR-Node FCRNode for the face F
        For each edge E in face F{
                If (E is a Neutral Edge){
                        Get (Create if not there) CR-Node AdjFCRNode for the adjacent face of F
                        Add AdjFCRNode to neighbor list of FCRNode
                        Add FCRNode to neighbor list of AdjFCRNode
                } // End if statement
                Else {
                        Get (Create if not there) CR-Node ECRNode for the edge E on face F.
```

```
                Add ECRNode to neighbor list of FCRNode
                Add FCRNode to neighbor list of ECRNode
                For each vertex V in edge E{
                    Get (Create if not there) CR-Node VCRNode for vertex V
                    If(VCRNode==NULLCR){ // for types 4a and 5a in Table 2-1
                        Get (Create if not there) CR-Node AdjECRNode for the next
                        non-neutral edge at vertex V
                        Add ECRNode to neighbor list of AdjECRNode
                        Add AdjECRNode to neighbor list of ECRNode
                    } // End if statement
                    else{
                        Add VCRNode to neighbor list of ECRNode
                        Add ECRNode to neighbor list of VCRNode
                    }
                } // End Vertex for loop
            } // End else statement
        } // End edge for loop
} // End face for loop
```

Algorithm 2-2: Obtaining the CR-Graph

After this algorithm is executed on a model the CR-Graph of the model is created. For example, shown in Figure 2-13 is a model and its partial CR-Graph (but without the topological entities). In Figure 2-13, for the purpose of clarity, some of the CRs that are adjacent to each other and identical in type are merged together. This is done through a process called Simplification, which is explained in the next section.



Solid Model   Partial CR-Graph

Figure 2-13: A model and its CR-Graph

The time complexity for evaluating the CR-Graph is $O(n_F + n_E)$, where n_F is the number of faces in the model, n_E is the number of non-neutral edges in the model (neutral edges are not considered during the evaluation of the CR-Graph). A vertex CR is evaluated by considering the incident edges at the vertex and hence the time taken to evaluate the vertex CRs is also dependent on n_E.

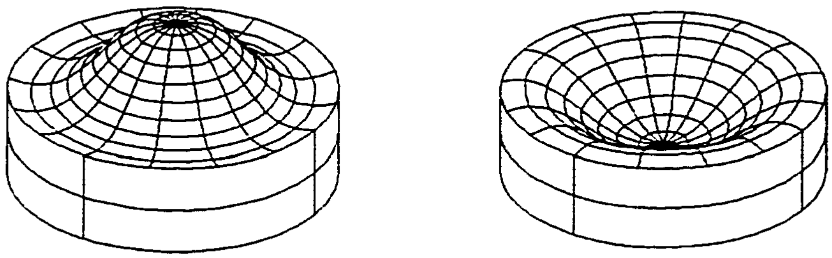
The number of entities in the CR-Graph is equal to the number of entities in the B-Rep model[4].

2.3 Primitive Shapes: Protrusions and Depressions

The Primitive Shape Abstraction of a model comprises the Protrusions and Depressions in the model. A protrusion is characterized by the presence of material on a part

[4] The number of entities in the CR-Graph will be more than the number of entities in the B-Rep model if the model has free-form surfaces or non-linear edges that result in multiple number of curvature regions.

(Figure 2-14(a)). A depression is characterized by the absence of material from a part (Figure 2-14(b)).



(a) Protrusion (b) Depression

Figure 2-14: Protrusion and Depression

In contrast to B-Rep elements and Curvature Regions, Protrusions and Depressions present a symbolic meaning to most designers. Therefore, a designer can define features more intuitively using the concept of Protrusions and Depressions.

Protrusions and Depressions are obtained from the Curvature Region Abstraction of a model via the application of two mappings, namely, Simplification (σ) and Primitive (π). These two mappings are described in detail in the following section.

2.3.1 Obtaining Protrusions and Depressions

2.3.1.1 Simplification Mapping (σ)

The Simplification Mapping (σ) is as follows,

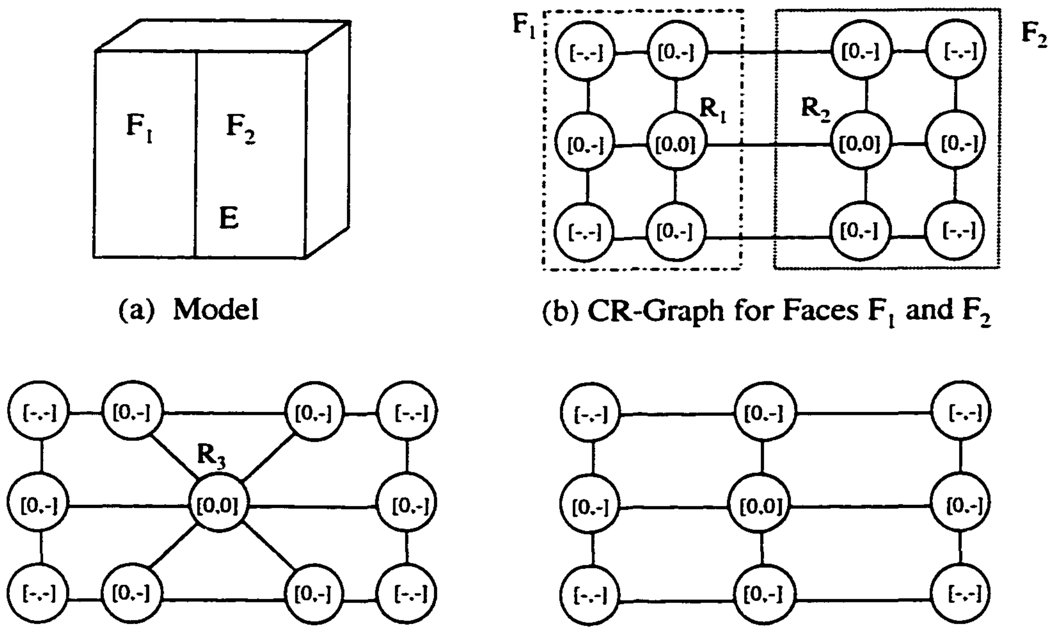
If curvature types τ ($[m,n]$) of two adjacent nodes R_1 and R_2 in the CR-Graph are identical, then the nodes R_1 and R_2 are replaced by node R_3 such that,

1) R_3 is of curvature type τ,

2) The set of adjacent nodes of R_3, R_{3adj} is determined as:

$$R_{3adj} = (R_{1adj} \cup R_{2adj}) - \{ R_1 \} - \{R_2\}. \qquad \text{Eq. 2}$$

Simplification of the CR-Graph combines adjacent CRs having identical curvature types to form a single CR (of the same curvature type). As an example, consider the object shown in Figure 2-15(a). The CR-Graph of this object, shown in Figure 2-15(b), contains several regions, two of which, R_1 and R_2, of type [0,0], are adjacent to each other. Figure 2-15(c) shows the CR-Graph after σ is applied to R_1 and R_2. It may be observed that nodes R_1 and R_2 combine to form a single region R_3, of type [0,0]. The same CR-Graph can be further simplified, resulting in the graph shown in Figure 2-15(d).



(a) Model

(b) CR-Graph for Faces F_1 and F_2

(c) Replacing R_1 and R_2 with R_3

(d) Further Simplification

Figure 2-15: Effect of Simplification of the CR-Graph

A useful property of the Simplified CR-Graph with respect to feature determination is that two parts that possess identical geometry but different topology map to an identical

Simplified CR-Graph. For example, the Simplified CR-Graph for the model in Figure 2-15(a), as shown in Figure 2-15(d), is identical to the Simplified CR-Graph of a part with identical geometry but without Edge E.

The Simplified CR-Graph is used as the basis for obtaining the Primitive Shape Abstraction. Utilizing the Simplified CR-Graph, two separate sets of interpretations are obtained: the first one corresponding only to Protrusions and the second one corresponding only to Depressions. Depending upon the manufacturing application, one interpretation may be sufficient or both interpretations may be necessary. For example, for machining, the depression interpretation is significantly more important since machining involves material removal. On the contrary, for injection molding, protrusion type features are more important, but some depression type features are also required. Therefore, the current approach determines both these interpretations independently, and subsequently adds these two interpretations to obtain the final Primitive Shape Abstraction. The Protrusions and Depressions are determined independently because a face may belong to a Protrusion as well as a Depression.

2.3.1.1.1 Algorithm for Simplification

The algorithm for simplifying a CR-Graph is as follows:

```
For each CR-Node N in the CR-Graph{
        For each neighboring node of N1 that is of the same type as N{
                If(N1 not already merged) Merge N1 with N
        }
}
```

Algorithm 2-3: Simplification of the CR-Graph

In the above algorithm, the merging of two nodes involves the transfer of, 1) neighbor list from node N_1 to N, and, 2) parent B-Rep elements from N_1 to N. The execution time of this algorithm is of the order $O(n_N + n_C)$ where n_N is the number of nodes in the CR-Graph and n_C is the number of connectivities (links between nodes) in the CR-Graph. That is, this is a linear time algorithm.

2.3.1.2 Primitive Mapping (π)

The mapping to obtain the Primitive Shape Abstraction (a set $\mathcal{P}$ of primitive shapes) from the Simplified CR-Graph $\mathcal{S}$ is referred to as π. Irrespective of the interpretation (Protrusion or Depression) required, a defined set of steps is followed to obtain the primitives. These steps are outlined in Figure 2-16, whereby π is decomposed into four sub-mappings π_1, π_2, π_3 and π_4.

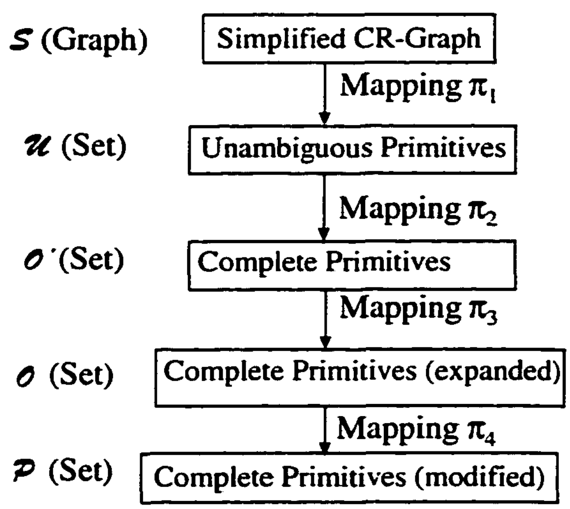
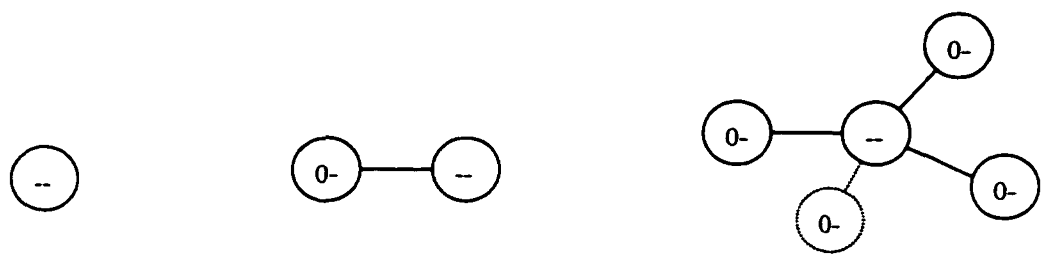


Figure 2-16: Interpreting the Simplified CR-Graph: Mapping π

The following definitions are used in determining the current mapping π.

Definitions:

1. **Minus-Minus Center (MMC)**: A sub-graph S (of [-,-] and [0,-] nodes) of the CR-Graph where: (i) the diameter[5] of S is less than or equal to 2, (ii) there exists only one [-,-] node, (iii) the [0,-] nodes in S are connected only to the [-,-], and (iv) all the [0,-] nodes in the CR-Graph that are adjacent to the [-,-] node belong to S. Figure 2-17 shows Minus-Minus Centers with the diameters being 0 (Figure 2-17 (a)), 1 (Figure 2-17 (b)) and 2 (Figure 2-17 (c)). The dotted line in MMC of diameter 2 (Figure 2-17 (c)) is to illustrate the condition (iv) above [82].



Diameter 0
(a)

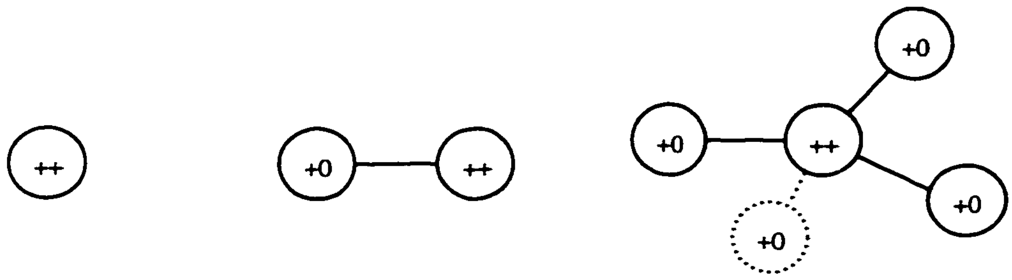
Diameter 1
(b)

Diameter 2
(c)

Figure 2-17: Minus-Minus Center

2. **Plus-Plus Center (PPC)**: A sub-graph S (of [+,+] and [+,0] nodes) of the CR-Graph where: (i) the diameter of S is less than or equal to 2, (ii) there exists only one [+,+] node, (iii) the [+,0] nodes in S are connected only to the [+,+] and (iv) all the [+,0] nodes in the CR-Graph that are adjacent to the [+,+] node belong to S. Figure 2-18 shows Plus-Plus Centers with diameters being 0 (Figure 2-18 (a)), 1 (Figure 2-18 (b)) and 2 (Figure 2-18 (c)). The dotted line in PPC of diameter 2 (Figure 2-18 (c)) is to illustrate the condition (iv) above.

[5] In a graph G, the largest distance between two vertices is called the diameter of G.



Diameter 0 (a)

Diameter 1 (b)

Diameter 2 (c)

Figure 2-18: Plus-Plus Center

1) **Minus-Zero Center (MZC)**: A [0,-] region that is not part of any MMC.

2) **Plus-Zero Center (PZC)**: A [+,0] region that is not part of any PPC.

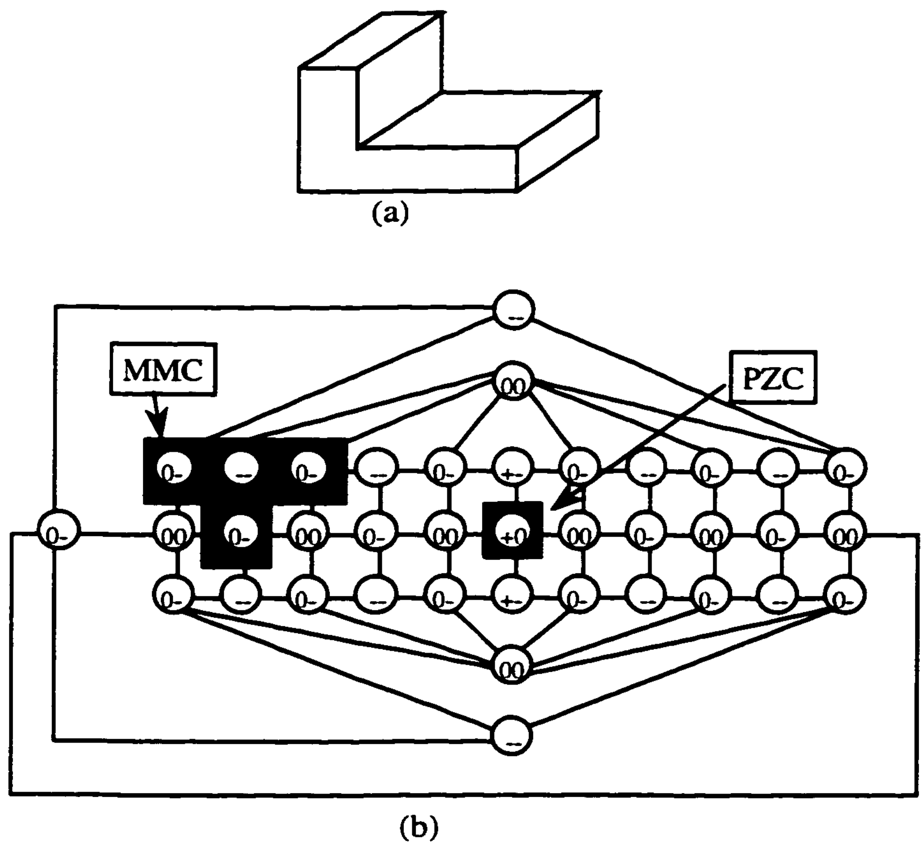


Figure 2-19: An example CR-Graph with MMC and PZC

Figure 2-19(b) shows instances of MMC and PZC in the CR-Graph of the object shown in Figure 2-19 (a). There are no instances of PPC and MZC in the CR-Graph of the object.

3) **Unambiguous Protrusion**: A Minus-Minus Center or a Minus-Zero Center.

6) **Unambiguous Depression**: A Plus-Plus Center or a Plus-Zero Center.

7) **Unambiguous Primitive**: Either an Unambiguous Protrusion or an Unambiguous Depression. The term "unambiguous" indicates that if a CR belongs to a protrusion then it cannot belong to a depression, and vice versa.

8) **Complete Primitive**: A Primitive that is created by "growing" an Unambiguous Primitive into its adjacent flat ([0,0]) and transition ([+,-]) regions. Let the set of CRs of an Unambiguous Primitive be S. A Complete Primitive is formed when S is expanded to contain the flat and transition CRs that are adjacent (in the CR-Graph) to the CRs in S. The term "growing" in the definition refers to the expansion of the set S to contain flat and transition CRs.

The above definitions are utilized in defining the mapping π. As illustrated in Figure 2-16 the mapping π that generates the Primitive Shape Abstraction from the Simplified CR-Graph ($\mathcal{S}$) comprises four sub-mappings, namely, π_1, π_2, π_3 and π_4. π_1 is the sub-mapping that determines the set of Unambiguous Primitives, $\mathcal{U}$, from $\mathcal{S}$. π_2 is the sub-mapping that generates Complete Primitives by growing the Unambiguous Primitives in $\mathcal{U}$ into adjacent flat and transition regions. The result of π_2 is a set of Complete Primitives, $\mathcal{O}'$. π_3 is the sub-mapping that generates new Complete Primitives by growing transition regions in $\mathcal{S}$, which do not belong to any of the primitives in $\mathcal{O}'$, to adjacent flat regions. Sub-mapping π_3 results in the expansion of the set of Complete Primitives to $\mathcal{O}$. π_4 is the sub-mapping that operates through a set of grouping rules on the Primitive Shapes in $\mathcal{O}$ to result in a modified set of Complete Primitives, $\mathcal{P}$. The mappings, π_1, π_2, π_3 and π_4, are symbolically represented as follows:

$$\mathcal{U} = \pi_1(\mathcal{S}) \quad \text{Eq. 3}$$

$$\mathcal{O}' = \pi_2(\mathcal{U}, \mathcal{S}) = \pi_2(\pi_1(\mathcal{S}), \mathcal{S}) \quad \text{Eq. 4}$$

$$\mathcal{O} = \pi_3(\mathcal{O}', \mathcal{U}, \mathcal{S}) = \pi_3(\pi_2(\pi_1(\mathcal{S}), \mathcal{S}), \pi_1(\mathcal{S}), \mathcal{S}) \quad \text{Eq. 5}$$

$$\mathcal{P} = \pi_4(\mathcal{O}, \mathcal{S}) = \pi_4(\pi_3(\pi_2(\pi_1(\mathcal{S}), \mathcal{S}), \pi_1(\mathcal{S}), \mathcal{S}), \mathcal{S}) = \pi(\mathcal{S}) \quad \text{Eq. 6}$$

The following sections describe the mappings π_1 through π_4.

2.3.1.2.1 Mapping π_1

Mapping π_1 maps $\mathcal{S}$ to the set $\mathcal{U}$. It is defined using Protrusion Growing Rules (which grow protrusions) and Depression Growing Rules (which grow depressions). Protrusion Growing Rules are designated by the mapping π_1^p and Depression Growing Rules are designated by the mapping π_1^d. Both are applied independently and the union of the two sets forms the final $\mathcal{U}$.

$$\mathcal{U} = \pi_1(\mathcal{S}) = \pi_1^p(\mathcal{S}) \cup \pi_1^d(\mathcal{S}) \quad \text{Eq. 7}$$

Mapping π_1^p: Mapping π_1^p forms a set of Unambiguous Protrusions from $\mathcal{S}$. π_1^p results in the formation of MMCs and MZCs from the CR-Graph. Minus-Minus Centers are created first from the CR-Graph. After the creation of MMCs, MZCs are created using [0,-] regions that do not belong to any MMC. By definition, MMCs and MZCs do not contain any concavity. Therefore, they unambiguously form protrusions.

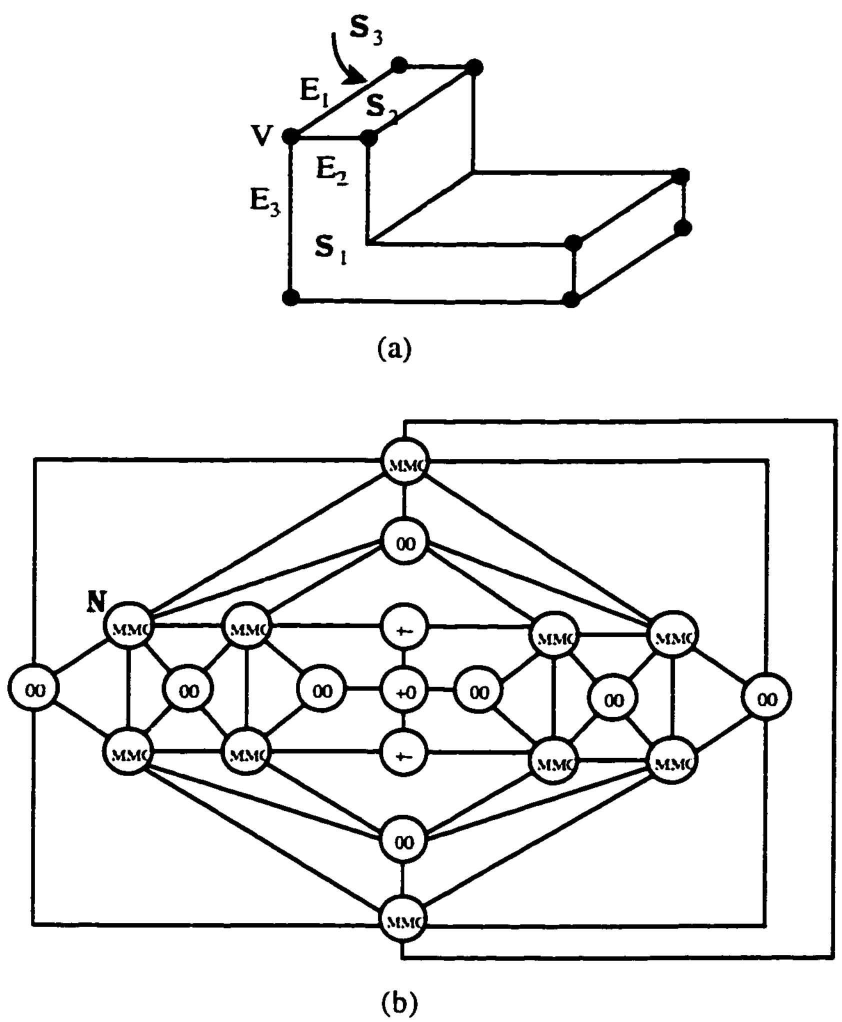


Figure 2-20: Forming MMCs and MZCs

Figure 2-20(b) shows the MMCs corresponding to the object in Figure 2-20(a) (the original CR-Graph for which was shown in Figure 2-19). For this object there are no instances of MZC. The MMCs correspond to the convex vertices along the edges incident on the vertices. For example, the MMC, N, in the CR-Graph corresponds to the following B-Rep entities: vertex V, edge E_1, edge E_2 and edge E_3.

Mapping π_1^d: Mapping π_1^d forms a set of Unambiguous Depressions from $\mathcal{S}$. The mapping π_1^d is identical to the mapping π_1^p except that it acts upon [+,+] and [+,0] regions instead of [-,-] and [0,-] regions respectively. π_1^d results in the formation of PPCs and PZCs from the CR-Graph. The mapping π_1^d follows a rationale identical to that for the mapping π_1^p.

2.3.1.2.2 Mapping π_2

Mapping π_2 grows each Unambiguous Primitive in $\mathcal{U}$ to its adjacent flat ([0,0]) and transition ([+,-]) regions, to form a set of Complete Primitives, $\mathcal{O}'$.

$$\mathcal{O}' = \pi_2(\mathcal{U}, \mathcal{S}) = \pi_2(\pi_1(\mathcal{S}), \mathcal{S}) \qquad \text{Eq. 8}$$

Similar to mapping π_1, mapping π_2 is defined using Protrusion Growing Rules (that grow protrusions) and Depression Growing Rules (that grow depressions). Protrusion Growing Rules are designated by the mapping π_2^p and Depression Growing Rules are designated by the mapping π_2^d. Both are applied separately and the union of the two sets of primitives is the final set. Thus,

$$\pi_2(\mathcal{U}, \mathcal{S}) = \pi_2^p(\mathcal{U}, \mathcal{S}) \cup \pi_2^d(\mathcal{U}, \mathcal{S}) \qquad \text{Eq. 9}$$

An example result of applying π_2^p is shown in Figure 2-21. Faces S_1, S_2 and S_3 are flat regions that lie adjacent to the Unambiguous Protrusion, node N in the CR-Graph of

Figure 2-21(b). After the application of mapping π_2^P, the Unambiguous Protrusion, N, grows into faces S_1, S_2 and S_3 and forms a Complete Protrusion (one element in $\mathcal{O}'$).

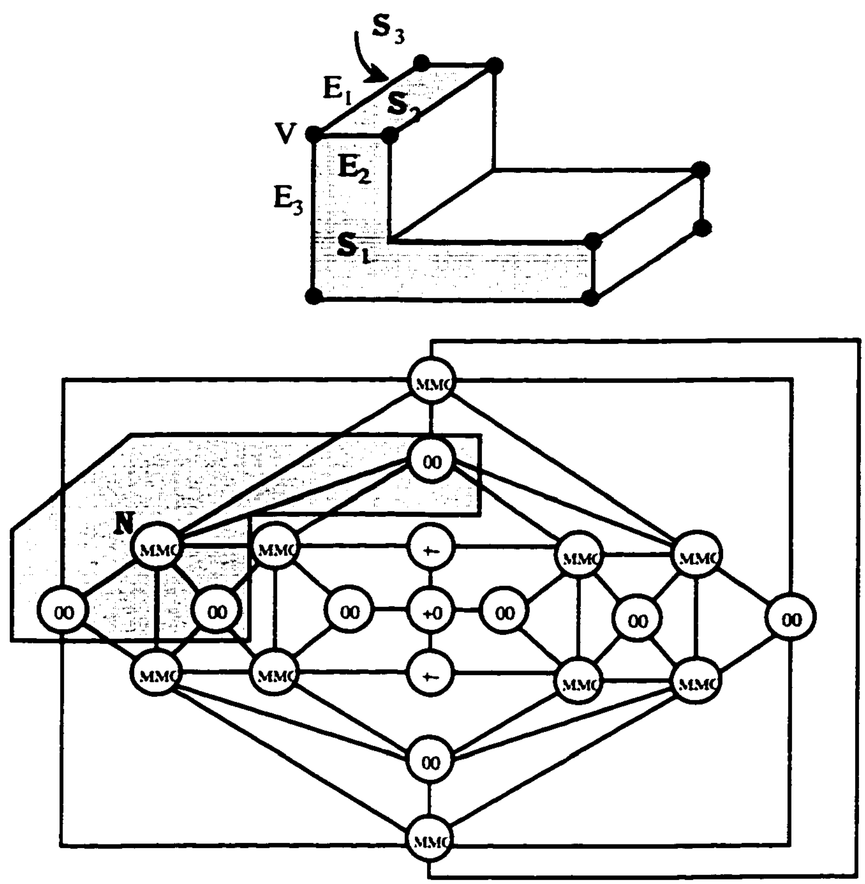


Figure 2-21: Unambiguous Protrusion extended to form a Complete Protrusion

2.3.1.2.3 Mapping π_3

Mapping π_3 grows transition ([+,-]) regions in $\mathcal{S}$ that do not belong to any primitive in $\mathcal{O}'$. Mapping π_3 is required because π_2 does not determine all the Complete Primitives. The mapping π_3 extends the set of Complete Primitives found using π_2. Mapping π_3 maps $\mathcal{S}$, $\mathcal{U}$ and $\mathcal{O}'$ to the set $\mathcal{O}$.

$$\mathcal{O} = \pi_3(\mathcal{O}', \mathcal{U}, \mathcal{S}) = \pi_3(\pi_2(\pi_1(\mathcal{S}), \mathcal{S}), \pi_1(\mathcal{S}), \mathcal{S}) \qquad \text{Eq. 10}$$

Similar to mappings π_1 and π_2, mapping π_3 is defined using Protrusion Growing Rules (that extend the set of Complete Protrusions) and Depression Growing Rules (that extend the set of Complete Depressions). Protrusion Growing Rules are designated by the mapping π_3^p and Depression Rules are designated by the mapping π_3^d. Both are applied separately and the union of the two sets is the final set. Thus,

$$\pi_3(\mathcal{O}', \mathcal{U}, \mathcal{S}) = \pi_3^p(\mathcal{O}', \mathcal{U}, \mathcal{S}) \cup \pi_3^d(\mathcal{O}', \mathcal{U}, \mathcal{S}) \qquad \text{Eq. 11}$$

Mapping π_3^p is defined by the rule: Transition ([+,-]) regions that are not part of any Complete Protrusion in the set $\mathcal{O}'$ form a Complete Protrusion by growing to adjacent flat ([0,0]) regions, if any.

The rationale for the rule is as follows. After the application of π_1 and π_2, there may exist transition ([+,-]) regions that do not belong to any Complete Protrusion. Such transition ([+,-]) regions are considered as separate protrusions (in the protrusion interpretation) due to the presence of convexity (- sign) in them. An example of such a transition ([+,-]) region is shown in Figure 2-22. The transition ([+,-]) region is completely surrounded by a [+,+] region. Therefore, it cannot become part of any protrusion after π_1^p and π_2^p are applied. With the application of π_3^p, the [+,-] region becomes a protrusion (intuitively, it may be inferred as a local protrusion).

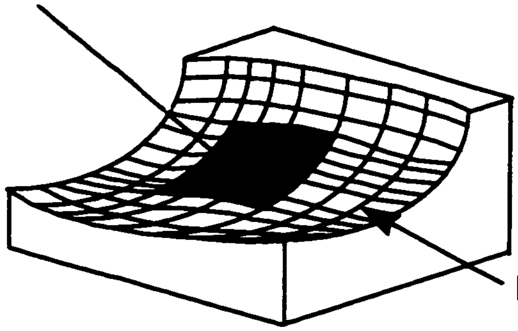


Figure 2-22: A transition region that is a protrusion

The rule for the mapping π_3^d follows a rationale identical to that for π_3^p.

The result of applying π_1, π_2 and π_3 is a set of Complete Primitives $\mathcal{O}$. The Complete Primitives in $\mathcal{O}$ may be used directly to obtain manufacturing features such as ribs, bosses and holes. However, the set $\mathcal{O}$ is still incomplete in the sense that there may not be any primitives corresponding to some of the manufacturing features. The primitives in the set $\mathcal{O}$ may be grouped together to obtain more primitives. Mapping π_4 (explained in the next section) is used to further extend the set of Complete Primitives so that there can be a Complete Primitive corresponding to every manufacturing feature.

2.3.1.2.4 Grouping of Complete Primitives: Mapping π_4

The mappings π_1, π_2 and π_3 described above obtain primitives by growing curvature regions. Mapping π_4 is defined via grouping rules that group together two or more overlapping or adjacent Complete Primitives in the set $\mathcal{O}$ based on certain criteria. Mapping π_4 is applicable when, in the set $\mathcal{O}$, there are two or more overlapping or adjacent primitives that together could be considered as a single primitive. Mapping π_4 is based on the notion that if there are two or more protrusions/depressions that are "connected" to each other, then the protrusions/depressions can be combined together to form a new protrusion/depression. The grouping criterion used for mapping π_4 is the "Maximal Commonality" of CR-Regions between grouped Complete Primitives.

Definition: A set S of n (>=1) Complete Primitives is Maximally Common if CR-Graphs of these primitives have at least one CR in common and no other Complete Primitive in the object contains the common CRs of the n Complete Primitives.

Since every CR is part of at least one Primitive Shape, corresponding to each CR there is at least one Maximally Common set. As an example for a Maximally Common set, consider the object shown in Figure 2-23. Consider a set S of four Complete Primitives (Protrusions) centered at vertices V_1, V_2, V_3 and V_4 that have in common only the CR corresponding to face F_1. The set S is Maximally Common since no other Complete Primitive contains the common CR (corresponding to face F_1).

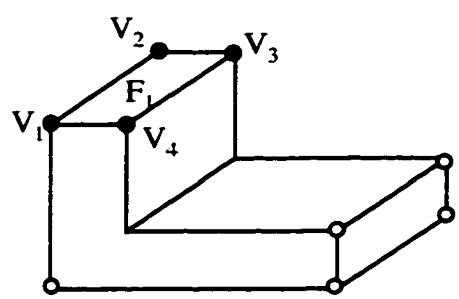


Figure 2-23: Maximal Common Set

By definition, a singleton set is Maximally Common if no other Primitive in the part has a CR in common with the Primitive in the singleton set.

An assumption that is being made here is that Grouping is performed only between primitives of the same type. That is, a Complete Protrusion is grouped with other Complete Protrusions, but not with a Complete Depression, resulting in another Complete Protrusion. Therefore, Maximally Common sets are restricted to contain primitives of the same type, i.e., all the primitives in a Maximally Common set are of the same type (protrusion or depression).

Let the n Complete Primitives that are obtained after mapping π_3 be called $L_{0,i}$ (i=1 to n). For example, for the object shown in Figure 2-23 there are eleven Complete Primitives ($L_{0,1}$ to $L_{0,11}$). Ten of the Complete Primitives are Complete Protrusions (MMCs and their surrounding flat and transition regions) and one is a Complete Depression (a PZC and its surrounding flat and transition regions).

The mapping π_4 is a recursive algorithm, where the i^{th} recursive step is described as follows:

Obtain all the Maximally Common sets using Complete Primitives of the previous level (i.e., at recursion step i the Complete Primitives $L_{i-1,1}$ to $L_{i-1,n}$ are used). If all the obtained Maximally Common sets are singleton sets, return without doing anything.

Let the number of Maximally Common sets obtained be m.

For (k = 1 to m){

Group the Complete Primitives in the k^{th} Maximally Common set and label the newly formed Complete Primitive as $L_{i,k}$.

}

Algorithm 2-4: i^{th} recursive step in mapping π_4

As an illustration of mapping π_4, again consider the object shown in Figure 2-23 and consider only Complete Protrusions for the sake of illustration. The Complete Protrusions centered at vertices V_1, V_2, V_3 and V_4 constitute a Maximally Common set. Using mapping

π_4, the Complete Protrusions are grouped together to obtain a Complete Primitive that is shown in Figure 2-24. For the sake of illustration, let the protrusions that exist prior to the first grouping operation be called L_0 Complete Protrusions (because they are labeled $L_{0,i}$ (say i = 1 to 4)) and after the first recursive grouping step let the resultant protrusions be called L_1 Complete Protrusions.

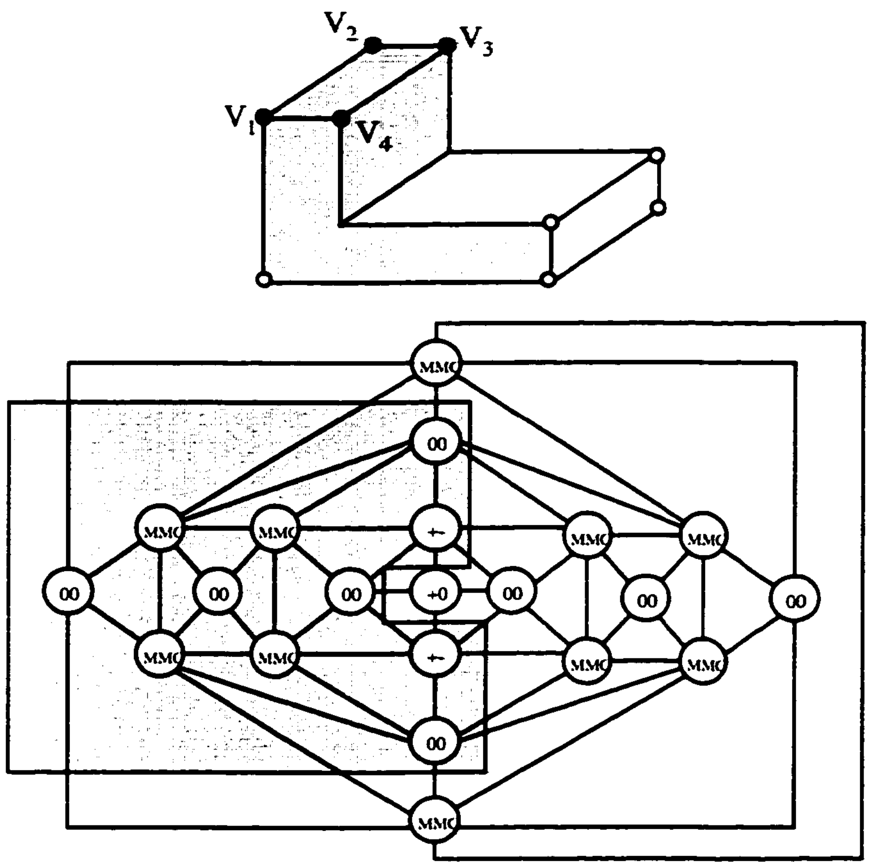


Figure 2-24: Grouping of Complete Primitives.

For the above object, after the first recursive step, when the Complete Protrusions in all the Maximally Common sets are grouped, six L_1 Complete Protrusions ($L_{1.1}$ to $L_{1.6}$) are obtained. The newly obtained Complete Protrusions are shown in Figure 2-25.

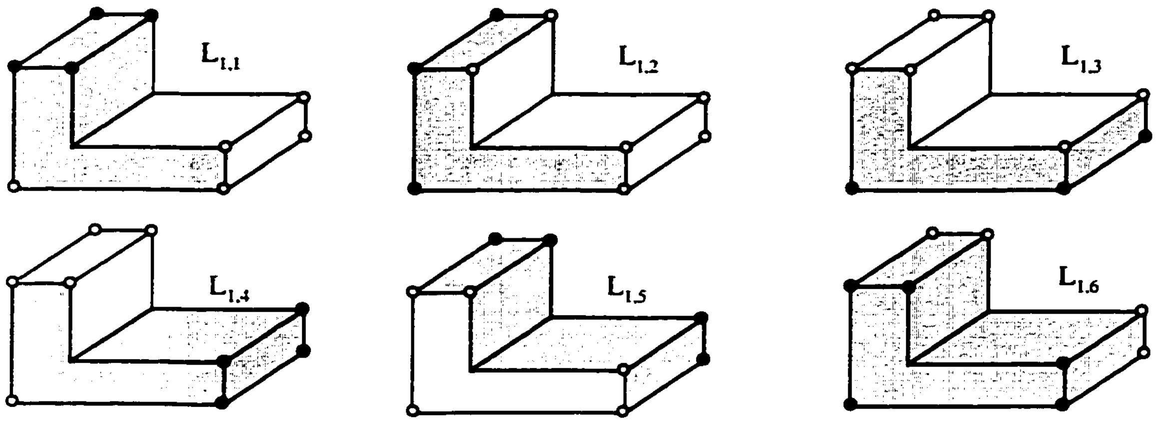
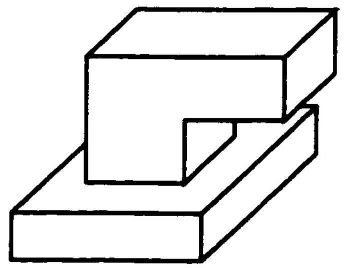


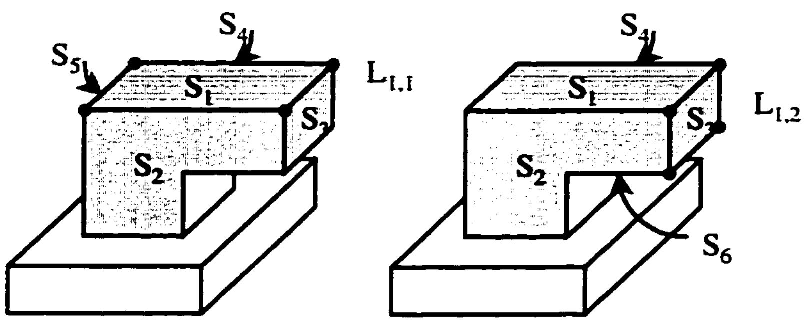
Figure 2-25: L_1 Complete Protrusions

After the second recursion step of mapping π_4, four Maximally Common sets of L_1 Complete Protrusions are obtained for the above case. The four Maximally Common sets are: {$L_{1.1}$, $L_{1.2}$, $L_{1.3}$, $L_{1.4}$, $L_{1.5}$}, {$L_{1.1}$, $L_{1.2}$, $L_{1.3}$, $L_{1.4}$, $L_{1.6}$}, {$L_{1.2}$, $L_{1.3}$, $L_{1.4}$, $L_{1.5}$, $L_{1.6}$}, {$L_{1.1}$, $L_{1.2}$, $L_{1.3}$, $L_{1.5}$, $L_{1.6}$}. When the L_1 Complete Protrusions in these sets are grouped, the entire object is obtained as a single Complete Protrusion and recursion stops at the next step because a singleton set is found.

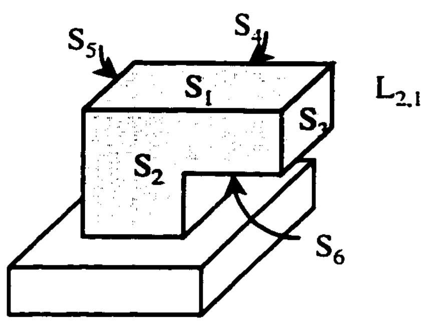
As another example, consider the object in Figure 2-26(a) and consider only the L-shaped protrusion for the analysis. Two L_1 protrusions, $L_{1.1}$ (constituting faces S_1, S_2, S_3, S_4 and S_5) and $L_{1.2}$ (constituting faces S_1, S_2, S_3, S_4 and S_6), shown in Figure 2-26(b), are created within the L-shaped protrusion after the first recursive step. These two protrusions are elements of a Maximally Common set of L_1 protrusions and hence they are grouped together in the next recursive step. As a result, an L_2 protrusion $L_{2.1}$ (constituting faces S_1, S_2, S_3, S_4, S_5 and S_6), shown in Figure 2-26(c), is obtained. The recursion stops after $L_{2.1}$ is obtained because the only Maximally Common set of L_2 protrusions is a singleton set that contains $L_{2.1}$.



(a) Model



(b) L_1 Protrusions



(c) L_2 Protrusion

Figure 2-26: Illustration of mapping π_4

The grouping algorithm as applied to depressions is similar to that of protrusions. As a result of mapping π_4, the set of Complete Primitives, $\mathcal{O}$, is extended to contain some more Complete Primitives. The extended set $\mathcal{P}$ of Complete Primitives is the Primitive Shape Abstraction of a model.

2.3.1.2.5 Algorithm for obtaining the Primitive Shapes

The Primitive Shapes of a model are obtained by analyzing the Simplified CR-Graph of a part. The steps followed in obtaining the Primitive Shapes are,

1) L_0 Primitive Shapes creation.

2) L_i Primitive Shapes creation.

L_0 Primitive Shape creation involves the creation of L_0 Protrusions and L_0 Depressions from the Simplified CR-Graph of the model. The algorithm for creating the L0 Protrusions is as follows:

```
// PrimitiveShapesArray is an array to store all the created Primitive Shapes
// Create the MMC based L0 Protrusions
For each CR-Node N of type [-,-] in the CR-Graph{
        Create a Protrusion Shape and add node N and its adjacent [0,0] and [+,-] nodes to the
        Protrusion
        Add adjacent [0,-] nodes of node N to the Protrusion and mark the [0,-] node as grouped
        Add adjacent [0,0] and [+,-] nodes of each [0,-] node in above step to the Protrusion
        Add Protrusion to PrimitiveShapesArray
} // End for loop
// Create the MZC based L0 Protrusions
For each CR-Node N of type [0,-] in the CR-Graph{
        if(node N is not already grouped){
                Create a Protrusion and add node N and its adjacent [0,0] and [+,-] nodes to the
                Protrusion
                Add Protrusion to PrimitiveShapesArray
        } // End if statement
} // End for loop
```

Algorithm 2-5: Obtaining L_0 Protrusions from the CR-Graph

The algorithm for creating the L_0 Depressions is similar to the one for L_0 Protrusions. L_i Primitive Shapes are created from the L_{i-1} Primitive Shapes. The algorithm for obtaining L_i Primitive Shapes has already been described in Algorithm 2-4.

The three abstractions that have been described in this Chapter are used in a Feature Definition Language that is utilized to define features and in feature recognition algorithms that are used to recognize features. The next three Chapters describe the Feature Definition Language, a graphical user interface to the FDL and the feature recognition algorithms.

3 Feature Definition Language

The abstractions in the previous chapter are utilized in a Feature Definition Language (FDL). The FDL is used to define features and the defined features are subsequently recognized in a model. A feature definition in the current research comprises a) the feature name, b) feature definition type, c) feature graph, d) attributes on the nodes of the graph and e) constraints between the attributes. The format for the feature definition is as follows:

```
FeatureName     FeatureType NodeMatchType
FEATUREGRAPH{
        Number of Nodes in the Graph
        Node1 NeighborCnt neighbor1_index neighbor2_index
        Node2 NeighborCnt neighbor1_index neighbor2_index ...
}
ATTRIBUTES {
        Number of Attributes
        Attributename1 AttributeType NodeNumber
        Attributename2 AttributeType NodeNumber ....
}
CONSTRAINTS {
        Number of Constraints
        Attributenamei ConstraintType AttributeNamej
        Attributenamej ConstraintType AttributeNamek ....
}
```

A feature is defined as a B-Rep-Graph, a CR-Graph or a Primitive Shape. The Feature Type in the above format is either BRGR (corresponding to B-Rep abstraction), CRGR (Curvature Region abstraction) or PROT/DEPR (corresponding to Primitive Shape Abstraction). $Node_1$, $Node_2$ in the above format correspond to the information about the nodes in the feature graph. If the graph used is a CR-Graph, the node information corresponding to a CR-Node (the Curvature Type) is specified in the definition. Similarly, if the graph used is a B-Rep-Graph, the node information corresponding to a BR-Node (edge/face type, edge/face geometry type, and edge nature) is specified in the definition. In the feature graph, the connectivities are specified as adjacency lists.

Attributes are specified on the nodes in the feature graph and constraints are specified between attributes. The types of attributes that are used in the current research are DIAMETER, LENGTH, FACENORMAL, ANGLE_AT_EDGE, FACEAREA and TANGENT_VECTOR. The types of constraints that are used are <, >, <=, >=, and == (which operate between two scalars), and *perpendicular-to*, *parallel-to*, *and anti-parallel-to* (which operate between two vectors).

An example of attribute specification to define the normal of a top face of a feature is as follows:

normal_topface FACENORMAL 1

Here, "normal_topface" is the attribute name, "FACENORMAL" is the attribute type and "1" is the index of location the feature's top face node in the graph specification. This attribute can be used as part of the definition for a rib feature whose top face normals are along the direction of normal of another face F. This can be specified as a constraint between the normals by using the constraint,

normal_topface *parallel-to* normalF

Here, "normal_topface" is the attribute name, "parallel-to" is the constraint type and "normalF" is the attribute corresponding to the normal of face F.

In the above format, the NodeMatchType in the feature definition refers to the feature recognition algorithm that is utilized in recognizing the defined feature. The next few sections describe the details of feature definition using the three levels of abstraction.

3.1 Feature Definition via B-Rep elements

Features have been defined using a B-Rep-Graph by several researchers [41][67][2][7][20][56][12]. The B-Rep-Graph definition used in this research is similar to that done by other researchers. An example of a B-Rep-Graph for a rib feature as used by other researchers is shown in Figure 3-1. Each node in the graph represents a face and each link in the graph represents an edge in the B-Rep model.

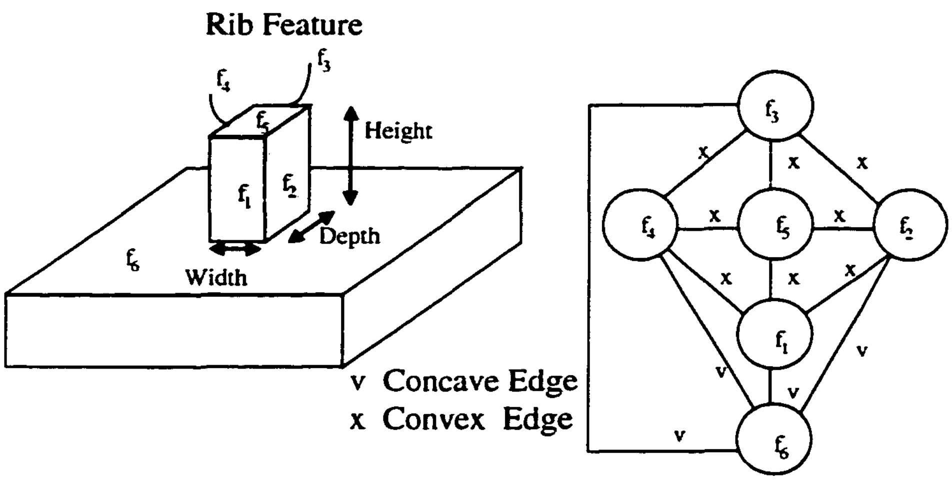


Figure 3-1: B-Rep Model and B-Rep graph

In a feature B-Rep graph in the current research, there is a node corresponding to each face and edge in the model. There is no node corresponding to a vertex. There are six types of face nodes and five types of edge nodes that can be used to in a feature B-Rep graph. The six types of face nodes are flat face, cylinder face, cone face, torus face, Spline face and B-

Spline face. The five types of edge nodes are: linear edge, arc edge, circular edge, Spline edge and B-Spline edge. For example, the rib feature in Figure 3-1 can be defined as shown in Figure 3-2. The links between nodes represent the connection relations between nodes.

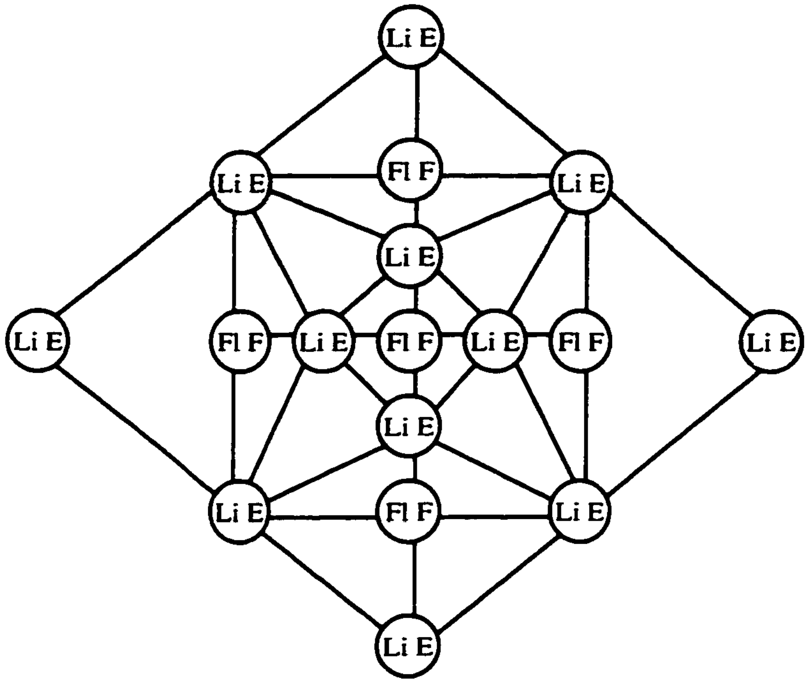


Figure 3-2: B-Rep graph for the rib feature

A feature definition in terms of a B-Rep elements comprises, 1) a BR-Graph that consists of BR-Nodes, 2) attributes attached to the nodes of the graph, and, 3) constraints on the attributes. Each entry corresponding to a BR-Node has information about the type of the node, geometry of the topological elements of the node, the edge nature if the node is an edge-node and the connectivity of the node (number of adjacent nodes and the node identities of the adjacent nodes).

Consider the example of a boss feature. The Feature Definition for a boss feature using the BR-Graph is as shown in Figure 3-3. There are 9 BRNODES in the definition. The face-node description includes the geometry of the face (planar, cylindrical, etc.) and the connectivity of the face-node to other nodes. For example, node 0 is a face-node that has a planar geometry and is connected to two other nodes, node 1 and node2. Similarly, the edge-node description includes the geometry of the edge (linear, arc, etc), the nature of the edge (convex, concave or neutral) and the connectivity of the edge-node to the other nodes. For example, node 1 is an edge-node whose edge geometry is an arc and the edge's nature is convex. Node 1 is connected to five other nodes, i.e., nodes 0, 2, 3, 5, 6.

```
BOSS1 BRGR
BRNODES{
 9
 FACE PLANE 2 1 2
 EDGE ARC CONVEX 5 0 2 3 5 6
 EDGE ARC CONVEX 5 0 1 4 5 6
 FACE CYL 4 1 5 6 7
 FACE CYL 4 2 5 6 8
 EDGE LINE NEUTRAL 6 1 2 3 4 7 8
 EDGE LINE NEUTRAL 6 1 2 3 4 7 8
 EDGE ARC CONCAVE 4 3 5 6 8
 EDGE ARC CONCAVE 4 4 5 6 7
}
```

```
ATTRIBUTES{
  5
  topface    TOPFACE 0
  sideface1  SIDEFACE 3
  sideface2  SIDEFACE 4
  diameter   EDGEDIAMETER  1
  height     FACELENGTH  3
}
CONSTRAINTS{
  1
  diameter less-than-equal-to height
}
```

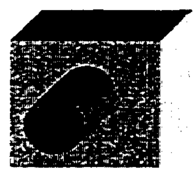


Figure 3-3: An example feature definition for a Boss using BR-Graph

After a feature's B-Rep graph is defined, attributes are attached to the nodes and then constraints are specified using the attributes. Examples of geometric attributes are: tangent at a point on an entity (edge/face), edge length, circular-edge diameter, face area, etc. For the example in Figure 3-3, five attributes have been defined. The face of node 0 is the top face of the boss and the faces of nodes 3 and 4 are the side faces of the boss. The diameter of the boss is the diameter of the edge of node 1 and the height of the boss is the FACELENGTH

(length of the longest linear edge on the face) of the face of node 3. One constraint has been added between the diameter and the height of the boss. The feature recognized is a boss feature only if the diameter of the boss is less than or equal to the height of the boss.

3.2 Feature Definition via Curvature Regions

As described earlier in the Chapter on Geometric Abstractions, there are six types of curvature regions, namely, [0,0], [0,-], [-,-], [+,0], [+,-] and [+,+]. A feature definition in terms of a Curvature Regions comprises, 1) a CR-Graph that consists of CR-Nodes, 2) attributes attached to the nodes of the graph, and, 3) constraints on the attributes. Each entry corresponding to a CR-Node has information about the type of the node and the connectivity of the node (number of adjacent nodes and the node identities of the adjacent nodes).

```
BOSS CRGRP
CRNODES{
      4
      ZEROZERO 1 1
      MINUSMINUS 2 0 2
      ZEROMINUS 2 1 3
      PLUSMINUS 1 2
}
ATTRIBUTES{
      2
      diameter DIAMETER 2
      height LENGTH 2
}
CONSTRAINTS{
      2
      diameter less-than-equal-to height
      diameter greater-than 2.0
}
```

Figure 3-4: An example feature definition for a Boss

An example of a feature definition using Curvature Regions is shown in Figure 3-4. The CRNODES information in the feature definition indicates that there are 4 nodes in the

CR-Graph of the boss feature. Two attributes, namely, diameter and height, are defined for the feature in Figure 3-4 and the constraints on the attributes are also specified.

The advantage of defining a feature using Curvature Regions is that a common definition can be used for all features that differ only slightly in their geometry and topology.

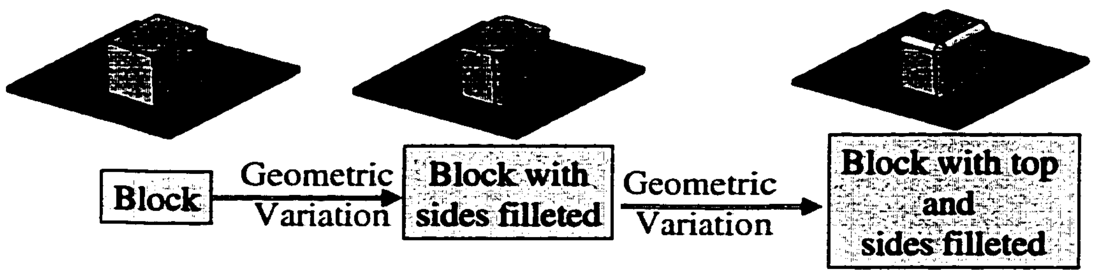


Figure 3-5: Variations of a rib feature.

```
RIB CRGRP
    CRNODES{
        21
        2 4 4 2 1 3
        3 4 8 5 0 16
        3 4 7 8 0 14
        3 4 6 7 0 15
        3 4 5 6 0 13
        4 3 1 4 12
        4 3 4 3 11
        4 3 3 2 10
        4 3 1 2 9
        3 4 14 16 8 17
        3 4 15 14 7 18
        3 4 13 15 6 19
        3 4 16 13 5 20
        2 3 11 12 4
        2 3 10 9 2
        2 3 11 10 3
        2 3 9 12 1
        6 1 9
        6 1 10
        6 1 11
        6 1 12
    }

ATTRIBUTES{
    9
    topface   TOPFACE 0
    sideface1 SIDEFACE 13
    sideface2 SIDEFACE 14
    sideface3 SIDEFACE 15
    sideface4 SIDEFACE 16
    length    LENGTH  1
    width     LENGTH  2
    height    LENGTH  9
    area      AREA    0
}
CONSTRAINTS{
    2
    width less-than length
    width less-than height
}
```

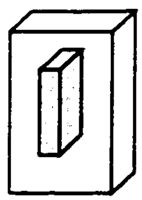


Figure 3-6: Rib feature definition

For example, all the features shown in Figure 3-5 can be extracted using one common feature definition in terms of Curvature Regions. The definition shown in Figure 3-6 can be used to extract all rib features, including minor variations of topology and geometry, such as those shown in Figure 3-5. If the definition were in terms of B-Rep elements (faces and edges) then each variation of feature would require a different definition.

3.2.1 Specifying the type of node matching algorithm

The feature recognition algorithm that is used in the current research is based on sub-graph matching [3]. Sub-graph matching is based on graph-matching and graph-matching is defined as follows:

Graph Matching: Given two graphs $G_1(V_1,E_1)$ and $G_2(V_2,E_2)$, to find a one-one and onto mapping f between V_1 and V_2 such that for $v_1, v_2 \in V_1, V_2, f(v_1) = v_2$ and for each edge of E_1 connecting any pair of nodes v_1 and $v_1' \in V_1$, there is an edge of E_2 connecting $f(v_1)$ and $f(v_1')$.

Sub-graph matching between two graphs $G_1(V_1,E_1)$ and $G_2(V_2,E_2)$ involves graph matching G_1 with a sub-graph of G_2.

During feature recognition, a sub-graph matching is performed between the graph (CR-Graph or B-Rep-Graph) of the defined feature and the graph (CR-Graph or B-Rep-Graph) of the entire part or of the Primitive Shapes. In the current research, node matching corresponds to the function f (in the above definition) that maps a node in graph G_1 to a node in graph G_2. The type of node matching used in the feature recognition algorithm to recognize a defined feature can be specified during the feature definition stage. There are three types of node matching that are utilized in the feature recognition and can be specified by a user during feature definition.

1) CRGRE: For an exact type of node matching. The criteria for matching two CR nodes is, a) the similarity of the CR type of the two nodes, and, b) the number of neighbors of the two nodes.

2) CRGRA: For an approximate type of node matching. That is, during the node matching process the number of neighbors of the two nodes that are matched does not have to be equal.

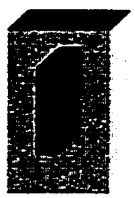
3) CRGRI: For an inexact type of node matching. That is, during matching all the sub-features that lie completely inside a face of a feature are ignored.

The feature type is CRGRP if the sub-graph matching has to be performed between the feature CR-Graph and the CR-Graphs of Primitive Shapes. In this case, an approximate node-match is performed between the nodes of the feature CR-Graph and Primitive Shape CR-Graph. The details of the node matching algorithms are given in the Chapter 5.

3.3 Feature Definition via Primitive Shapes

In addition to B-Rep elements and Curvature Regions, a feature can also be defined as a Primitive Shape with attributes and constraints. This type of feature definition is more generic than the definitions in terms of Curvature Regions or B-Rep elements. Two types of Primitive Shapes are allowed: Protrusion (PROT) and Depression (DEPR). A feature definition in terms of a Primitive Shapes comprises, 1) Type of Primitive Shape, 2) nodes (CRNODES) contained in the feature, 3) attributes attached to the nodes, and, 4) constraints on the attributes. Each entry corresponding to a CR-Node has information about the type of the node and the connectivity of the node (number of adjacent nodes and the node identities of the adjacent nodes).

For example, a rib feature may be defined as a Protrusion containing five flat faces (i.e., five [0,0] Curvature Regions), as shown in Figure 3-7.



```
RIB PROT
CRNODES{
     5
     2 0
     2 0
     2 0
     2 0
     2 0
}
ATTRIBUTES{
     0
}
CONSTRAINTS{
     0
}
```

Figure 3-7: Rib feature definition as a Primitive Shape

However, feature extraction using this definition results in all protrusion features consisting of five or more flat faces. Therefore, a cube (six flat faces) is also determined as a rib feature. To make a definition of this type less general, more information must be provided or more constraints must to be specified on the elements of the primitive shape. One method of providing more information is to specify the connectivity information between the elements in the Primitive Shape. As the definition is made progressively less general, in the limit, this definition is similar to either the feature definition using Curvature Regions.

The advantage of this definition is that by virtue of its generic nature, feature extraction is computationally faster. This type of definition can be used for obtaining estimate information that can be used for manufacturability analysis. For example, for an injection-molded part, if an estimate is required on the volume of rib features present on the part, then the feature definition shown in Figure 3-7 is used.

In summary, to define a feature, a user can employ: 1) B-Rep elements, or, 2) Curvature Regions, or, 3) Primitive Shapes. For example, as shown in Figure 3-8, a boss feature can be defined as a Primitive Shape (protrusion) with a cylindrical side face. A boss can also be defined as a CR-Graph, i.e., using curvature region elements in the Curvature Region Abstraction or completely in terms of the B-Rep elements, as an FEV Graph.

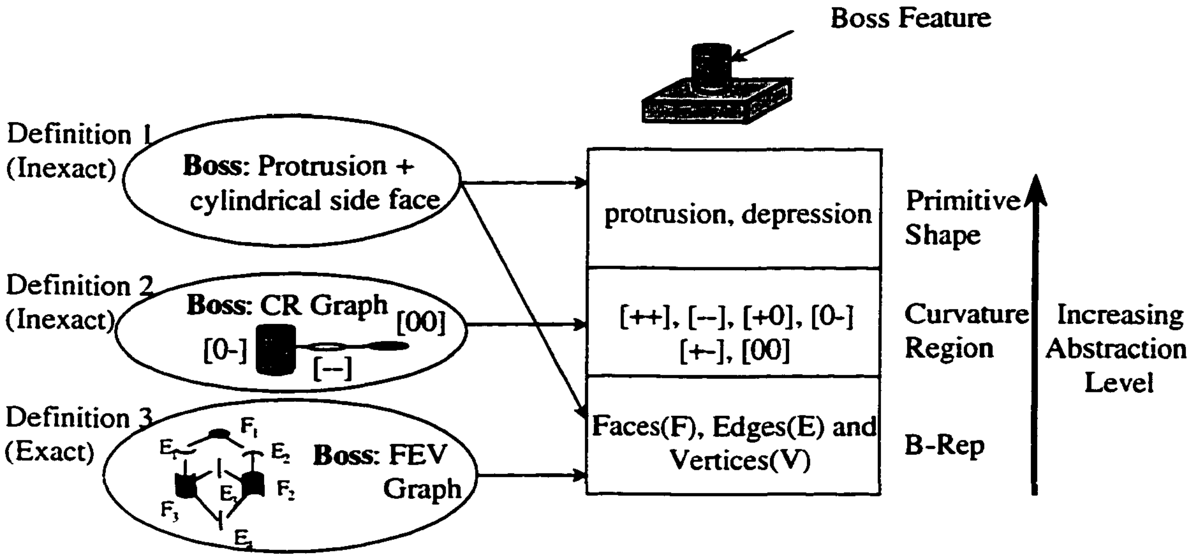


Figure 3-8: Feature definition using the different abstractions

A feature can, therefore, be defined in multiple ways using the different levels of abstraction. A feature definition is more precise when lower levels of abstraction are used, though the feature definition is not generic. For example, for the boss defined in Figure 3-8, there is no ambiguity in Definition 1 however it consists of a large graph. Definition 2 is more generic and Definition 3 is even more generic. A feature, defined in terms of an FEV Graph is the most precise and least generic and this is because an FEV Graph has to be generated for every geometric or topological variation of the feature.

After a feature is defined it can be recognized in a part using a feature recognition algorithm. The Feature Recognition algorithms are described in detail in Chapter 5.

A graphical front-end has been developed for the Feature Definition Language so that features may be defined interactively. The graphical interface also provides a template definition tool using which a feature can be defined by directly interfacing with the CAD model. This method of feature definition is similar to the one developed by Ranta *et al.* [Rant93] though the cut and paste of features is not applicable here. The graphical user interface is described in the next Chapter.

4 User Interface for Feature Definition

As described in the previous chapter, a Feature Definition Language is used in the current research to define features. Defining a feature in terms of the FDL directly is a cumbersome process. Therefore, a user interface has been developed to facilitate a user to define features interactively rather than editing/creating the feature definitions using a text editor. Two interfaces are provided to users to define features: 1) CAD system independent interface, 2) CAD system dependent interface.

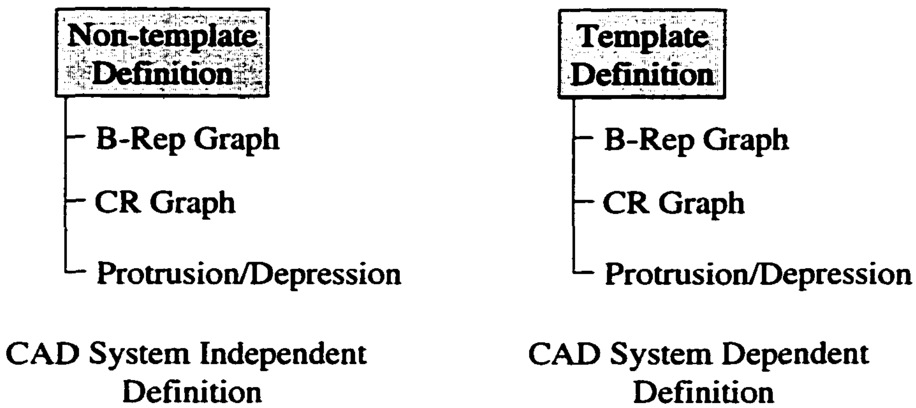


Figure 4-1: Non-template and Template definitions

In both the interfaces, features can be defined using any of the three geometric abstractions. The CAD system independent definition is also called non-template definition and the CAD system dependent interface is also called Template Definition (Figure 4-1).

4.1 Non-Template Feature Definition Interface

Using this type of interface, a user can create a feature definition by interactively creating a graph (BR-Graph or CR-Graph) and then adding attributes and constraints to the nodes in the graph. Figure 4-2 shows the initial window of the Non-Template User Interface.

For creating a feature definition, the user needs to choose an option from the "Options" menu first.

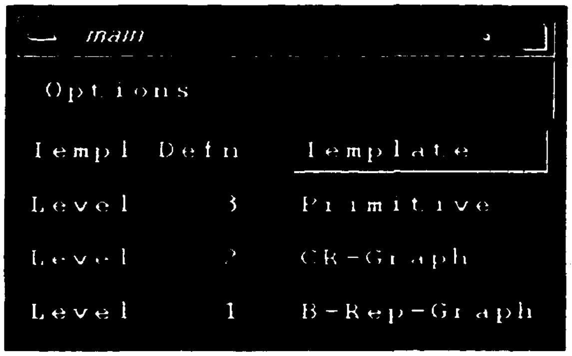


Figure 4-2: Initial window of the Non-Template User Interface

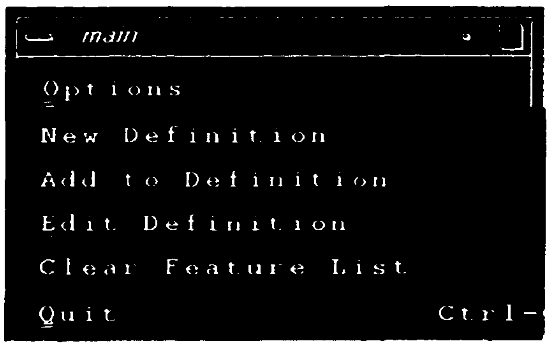


Figure 4-3: "Options" in FDL interface

The Options menu (shown in Figure 4-3) provides the user with the options,

- **New Definition**: This option is selected to define a new feature. After selection, the user is requested to provide a feature name. The new feature definition is added to a database of defined features. If the name of the new feature is the same as an existing feature in the database, the old feature definition information is overwritten with the new definition.

- **Add to Definition**: This option is selected to add information to an already existing feature definition.

- **Edit definition**: This option is selected to modify an existing feature definition by editing the B-Rep-Graph or the CR-Graph.
- **Clear Feature List**: This option is selected to clear all the definitions is the features database.
- **Quit**: This option is selected to exit the "FDL" interface.

After an appropriate menu option is selected, one of the buttons in the main window in Figure 4-2 is chosen.

- **B-Rep-Graph**: To define a feature by using B-Rep-Graph (Level 1).
- **CR-Graph**: To define a feature by using CR-Graph (Level 2).
- **Primitive**: To define a feature by using Primitive Shapes (Level 3).
- **Template**: To define a feature by selecting entities on a model in the CAD system. This option is used only for Template feature definition.

The following sections present the interfaces that are used for defining features using the three levels of abstraction.

4.1.1 Defining features using B-Rep-Graph

When the B-Rep-Graph definition option is chosen, a graphics window, as shown in Figure 4-4, is presented to the user to define a feature in terms of a B-Rep-Graph.

The graphics window contains icons corresponding to the 6 types of faces (flat, cylinder, cone, torus, Spline and B-Spline) and 5 types of edges (linear, arc, circle, Spline and B-Spline). Therefore, there are 11 types of nodes that are available. Adding the nodes of the graph and subsequently connecting the nodes creates a B-Rep-Graph. To complete the feature definition, subsequent to the B-Rep-Graph creation, attributes are attached to the nodes and constraints are specified between the attributes.

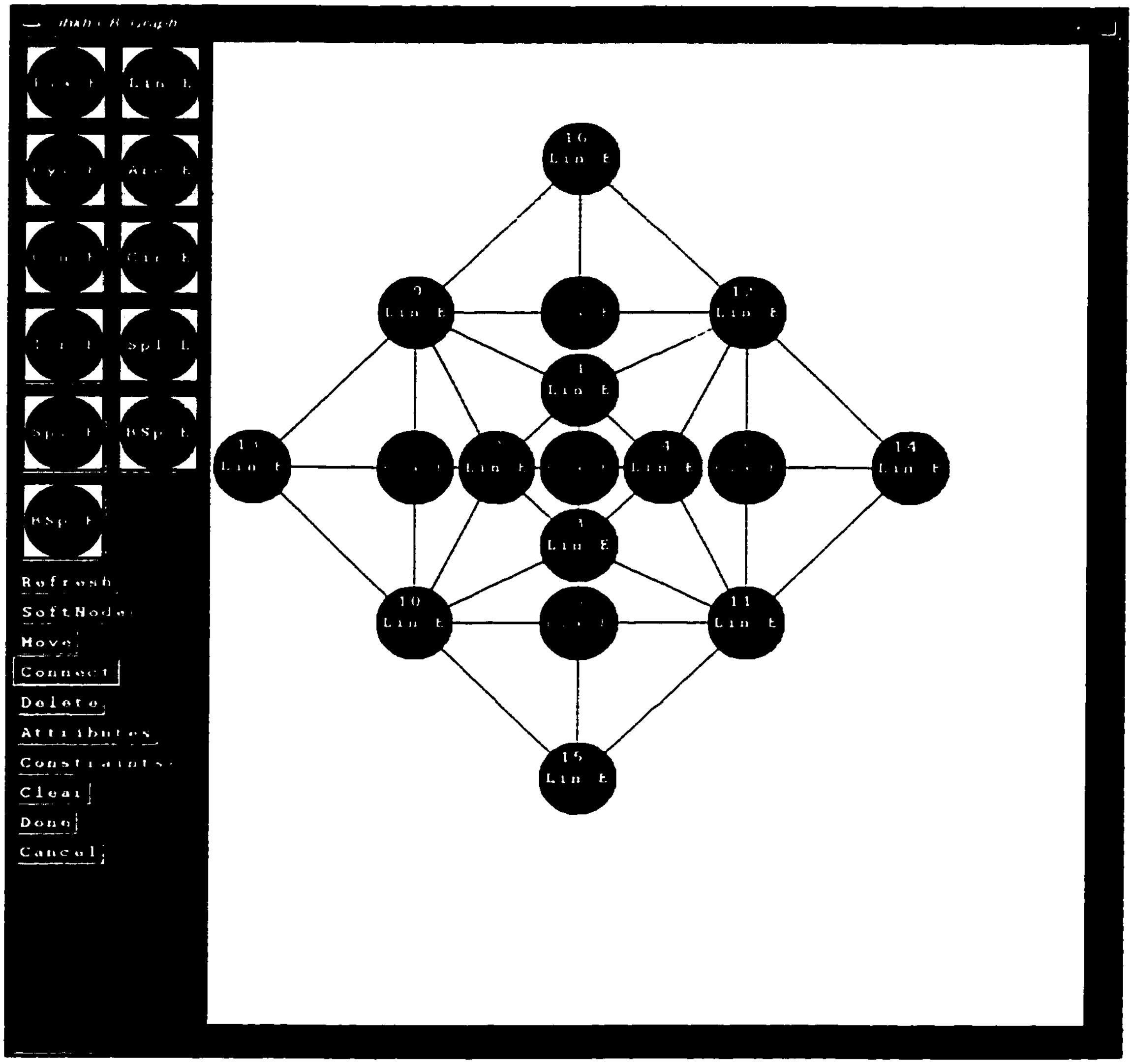


Figure 4-4: Interface for defining a feature using a B-Rep-Graph

Attributes to a node are added by choosing the "Attributes" option (shown in Figure 4-4) and selecting the node in the graphics window. Suppose the feature being defined is the rib feature, as shown in Figure 4-5. Assuming that the attributes are to be added to the nodes corresponding to faces F_1 and F_3, selecting the face nodes corresponding to the faces results in the display of two windows that are as shown in Figure 4-6.

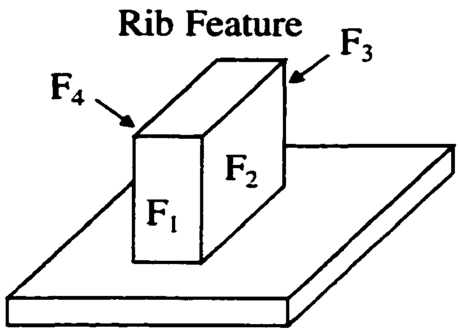


Figure 4-5: Rib feature

Entering the attribute names against the attribute types specifies the attributes of a node. In the example in Figure 4-6, the attributes that are being attached are: FACEAREA attributes f_area0, f_area1 and FACENORMAL attributes f_normal0, and f_normal1. Also, in the "IS_A" selection box, the "Sideface" option is chosen to indicate that the faces corresponding to the selected nodes are side faces on the feature. An additional option to specify the edge nature is presented when attributes are attached to an edge (Figure 4-8).

After the attributes are attached to the nodes, constraints are to the newly created attributes by choosing the "Constraints" option (shown in Figure 4-4). Figure 4-7 shows the display window that is used to specify the constraints. Choosing the two operands of the constraint and the constraint operator specifies a constraint. For the example in Figure 4-7, the f_area0 is being constrained to be less-than 600 units (Figure 4-7(a)). Also f_area1 is being constrained to be equal-to f_area1 and the face normals are being constrained to be anti-parallel-to each other (Figure 4-7(b)).

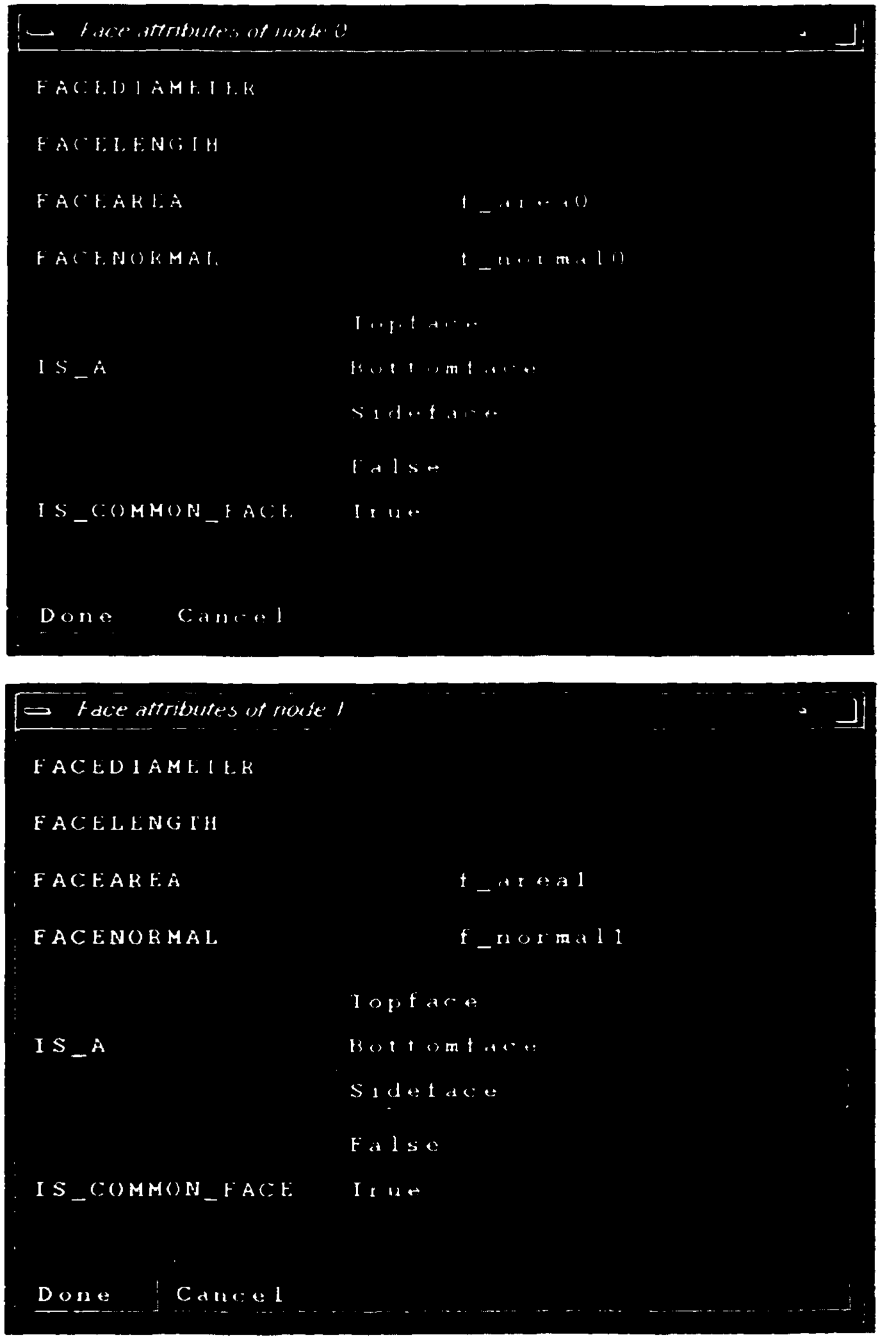
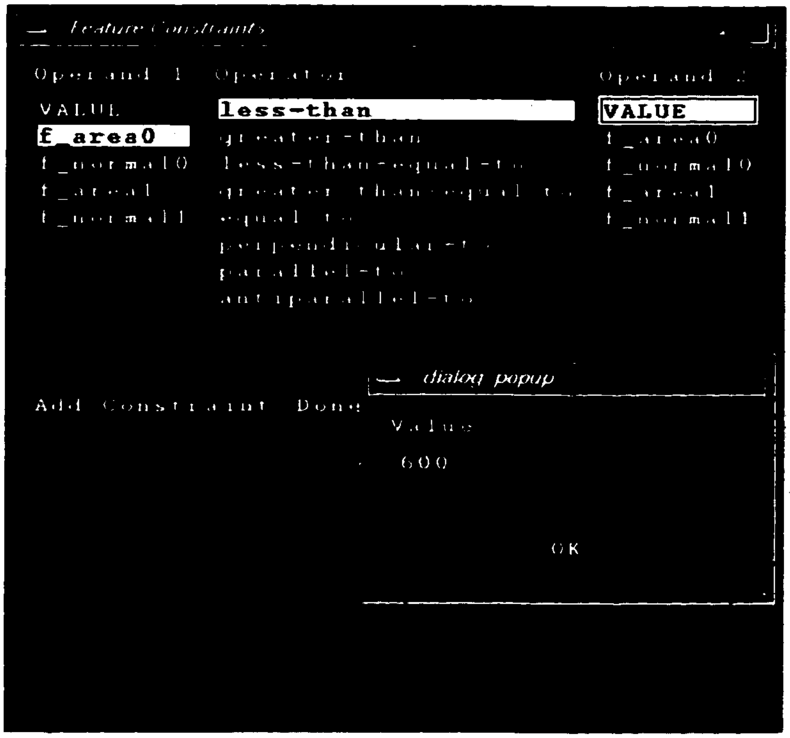
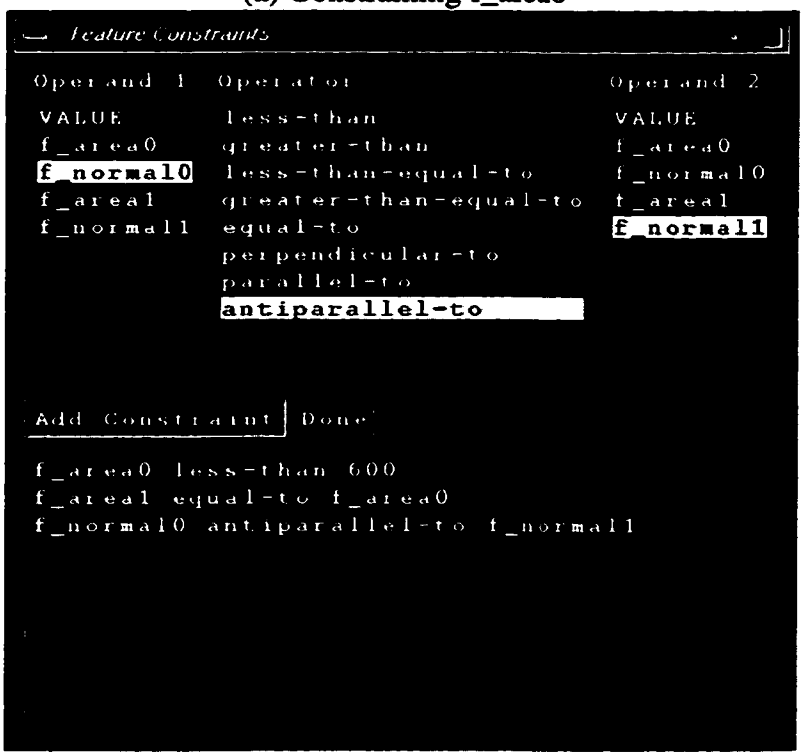


Figure 4-6: Adding attributes to face nodes



(a) Constraining f_area0



(b) Constraining face normals

Figure 4-7: Adding constraints to nodes

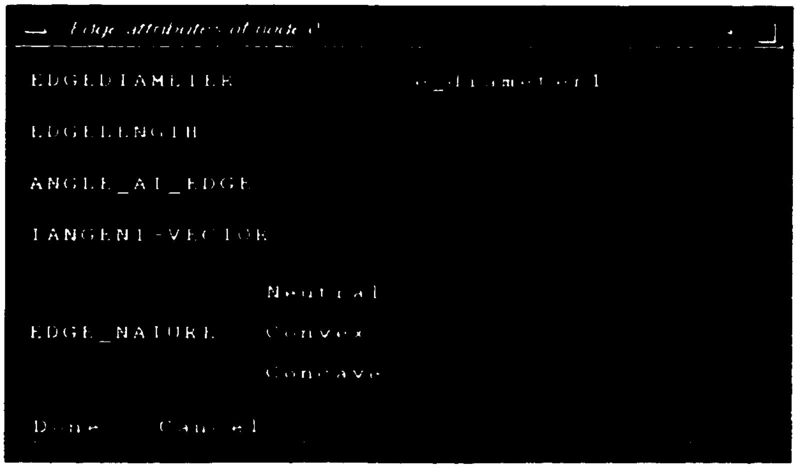


Figure 4-8: Adding attributes to edge node

To save the feature definition to a file, the "Save" option is chosen. The B-Rep-Graph feature definition is subsequently written to a file.

4.1.2 Defining features using CR-Graph

When the CR-Graph definition option is chosen, a graphics window that is similar to the B-Rep-Graph window is presented to the user (Figure 4-9).

The graphics window contains icons corresponding to the 6 types of Curvature Regions ([0,0], [0,-], [-,-], [+,0], [+,-], [+,+]). Adding the nodes of the graph and subsequently connecting the nodes creates a CR-Graph. To complete the feature definition, subsequent to the B-Rep-Graph creation, attributes are attached to the nodes and constraints are specified between the attributes. In the CR-Graph definition, the attributes and constraints are added in a manner identical to that in B-Rep-Graph definition.

The desired node-matching algorithm (CRGRA, CRGRP, CRGRE and CRGRI) is input when the feature definition is saved.

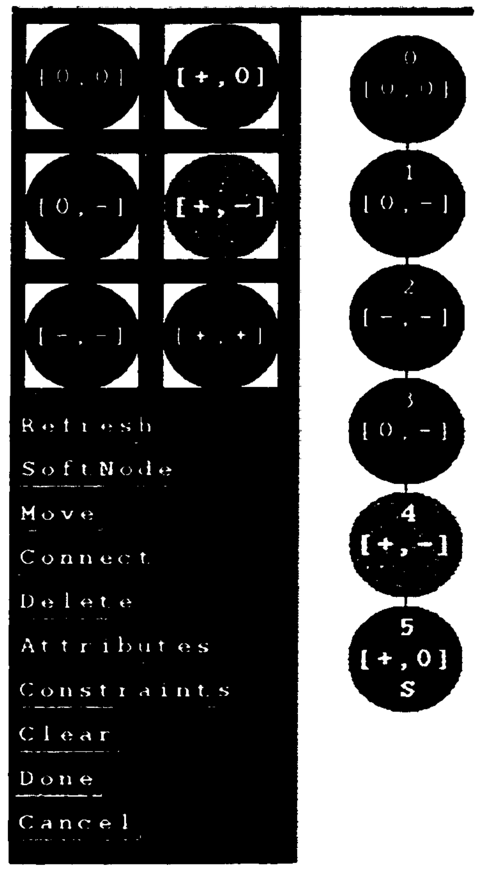


Figure 4-9: Interface for defining a feature via CR-Graph

4.1.3 Defining features using Primitive Shapes

To define a feature as a Primitive Shape, it is first defined as either a protrusion or a depression, using the window shown in Figure 4-10.

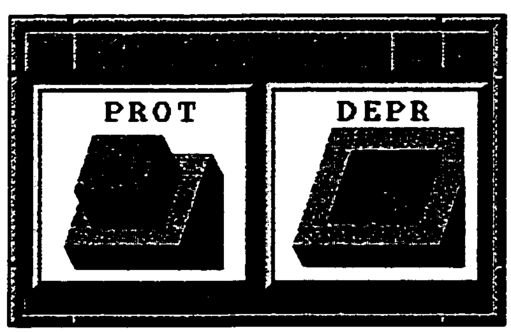


Figure 4-10: Primitive Shape Definition window

CR nodes, and attributes and constraints on the nodes are subsequently added using the CR-Graph definition interface. For example, a slot feature may be defined as: *depression with only one [+,0] region*. The [+,0] region is added to the definition of the slot feature using the CR-Graph window interface.

For a definition of type PROT or DEPR, the feature recognition is performed against the Primitive Shapes that are determined as described in Chapter 2.

4.2 Template Feature Definition Interface

Another method to define a feature is via a template definition tool, i.e., a features is defined by selecting entities on the CAD model, attaching attributes to the entities and specifying constraints. Therefore, the template definition tool provides a means to define a feature by directly interfacing with the CAD model. A feature defined as a template is represented as a B-Rep Graph or a CR-Graph or as a Primitive Shape.

A template feature definition is created by choosing the CAD System specific menu. Figure 4-11 shows the CAD system specific– "-Templ Defn" menu option for the ProEngineer® CAD system. Selecting the "-Templ Defn" menu option results in a "SELECTEMPL" sub menu. This menu allows a user to create the feature template in two ways (Figure 4-12(a)):

1. Using primitives: The "Protrusion" option allows the creation of a template from a Protrusion in the model and the "Depression" option allows the creation of a template from a Depression in the model. After choosing either "Protrusion" or "Depression" option, selecting any face or edge in the model results in the Primitive Shape that

includes the geometric entity selected to be highlighted. The CR-Graph of this Primitive Shape along with the attributes and constraints is stored as a template in a file. Subsequently the template is used to perform feature recognition.

2. Using faces and edges: Another method to define a feature template is to select the faces (using the "Face" menu option) and edges (using the "Edge" menu option) on the feature that needs to be stored as a template.

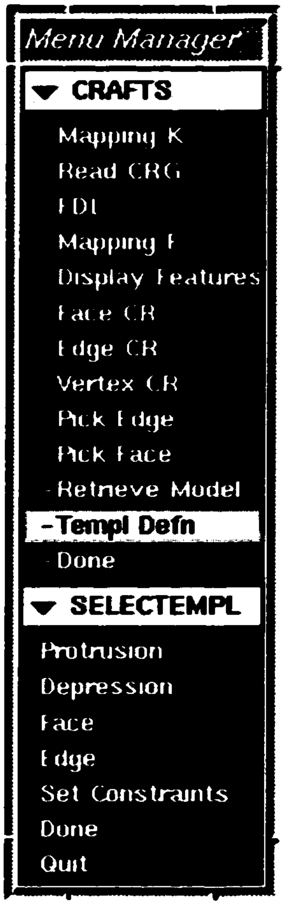
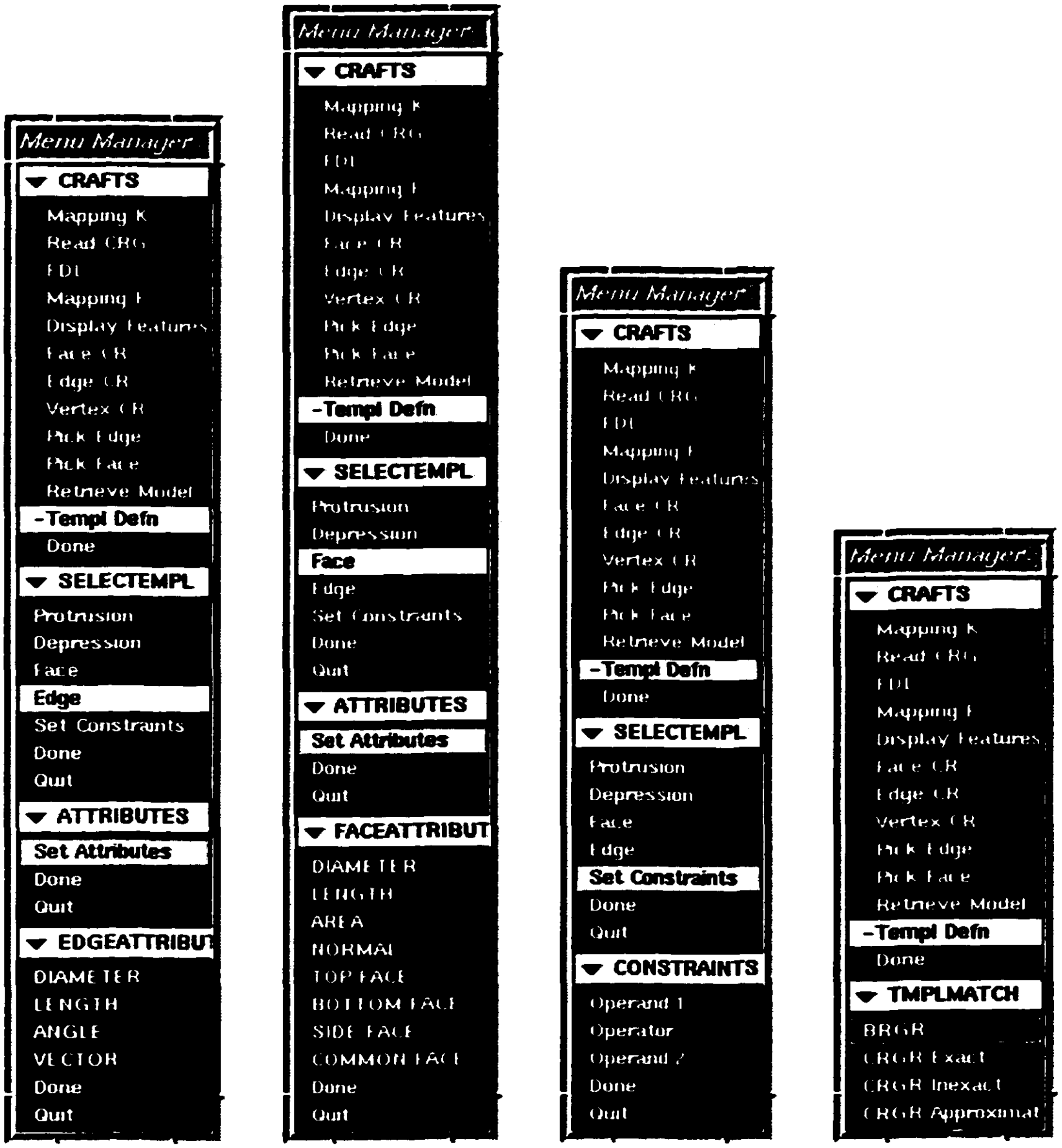


Figure 4-11: Interface for defining a feature via a template

Subsequent to the selection of the entities in the CAD model, attributes and constraints are added to the edges and faces. The method to create attributes and add constraints is identical to the method used in Non-Template Feature Definition. The only difference is that the CAD system (ProEngineer®) menus are used instead of the FDL windows.



(a) (b) (c) (d)

Figure 4-12: Assigning attributes and adding constraints to a template

Figure 4-12(a) and Figure 4-12(b) show the menus that are used to add attributes to the edges and faces in the model respectively. Figure 4-12(c) shows the menus that are used to add constraints. During the template creation the graph format (CR-Graph or B-Rep-Graph) of the template is also specified. The menu options corresponding to this are shown in Figure 4-12(d).

Using the above-described User Interfaces to the Feature Definition Language features are defined interactively. The defined features are recognized using Feature Recognition algorithms that are described in the next Chapter.

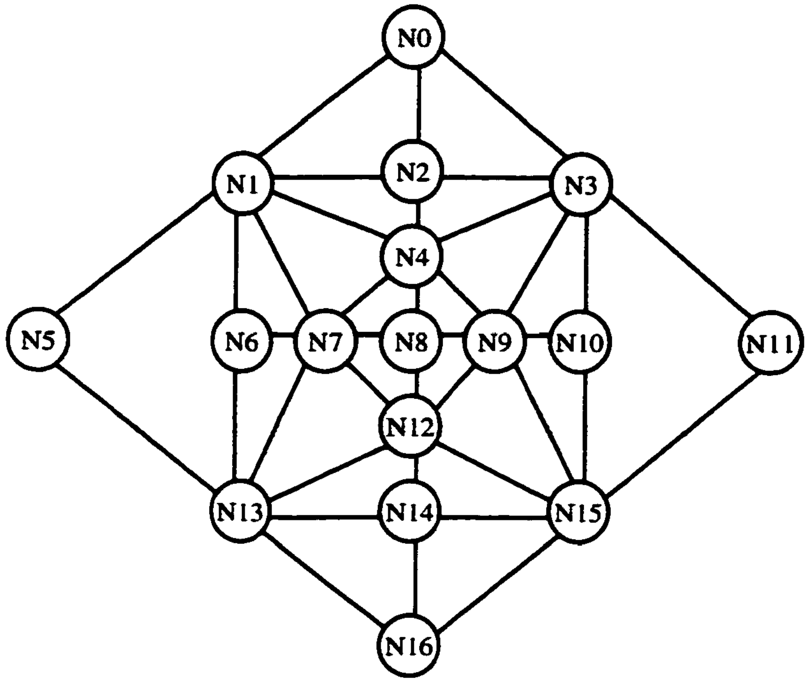
5 Feature Recognition Algorithms

In the current research, before performing feature recognition, the B-Rep-Graph, CR-Graph and the Primitive Shapes of the part are obtained by using the algorithms described in Chapter 2. The features are subsequently recognized using sub-graph-matching algorithms. The feature recognition algorithm that is employed is dependent on the feature definition. The feature recognition is based primarily on the type of feature graph (B-Rep-Graph or CR-Graph) in the feature definition. The next few sections describe the feature recognition algorithms that are used in the current research.

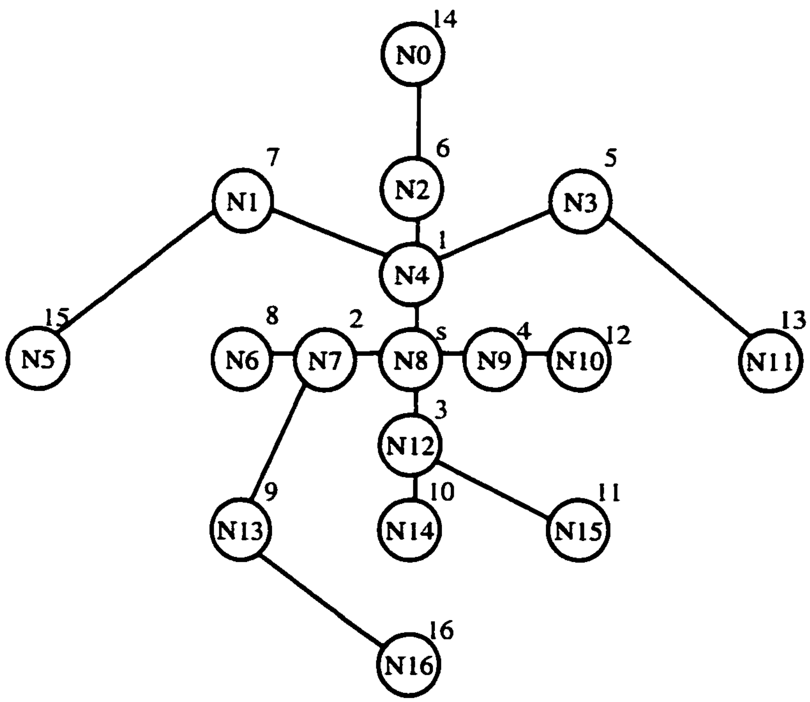
5.1 B-Rep-Graph-match Algorithm

A B-Rep-Graph-match algorithm is used to recognize a feature that is defined in terms of the B-Rep-Graph. The B-Rep-Graph of the feature is matched against the B-Rep-Graph of the entire part. The defined feature is recognized in the part when an exact match is found. The matching algorithm is based on the concept of a breadth-first search (BFS) of a graph [1].

The breadth-first search (BFS) explores a graph $G=(V,E)$ across the breadth of the graph between explored nodes and unexplored nodes. A node is examined the later the more it is distant form the starting node s. That is, the algorithm explores all nodes at distance k from the starting node s before discovering any nodes at distance $k+1$. As a result one gets the shortest paths to each node v of V that is reachable from s. A shortest path from s to v is the path that consists of the least number of edges. The tree that is generated from the BFS algorithm is called the BFS tree.



(a) Graph



(b) BFS Tree

Figure 5-1: Breadth First Search on a Graph

For example, for the graph shown in Figure 5-1(a), the BFS tree that is generated using the BFS algorithm is shown in Figure 5-1(b). The sequence of nodes visited starting from the node N8 is: N4, N7, N12, N9 at distance 1, N3, N2, N1, N6, N13, N14, N15, N10 at distance 2, and, N11, N0, N5, N16 at distance 3. A breadth-first match algorithm that is used in the current research is described as follows:

```
        Given two graphs G1 (feature graph) and G2 (part graph), and start nodes S1 and S2 in G1 and
G2 respectively, to match graph G1 in G2
If (node S1 matches with node S2){
        Get the neighbors of S1 and store them in list1 and get the neighbors of S2 and store them in
        list2.
        While( matching process is not complete){
                For each node N1 in list1{
                        Find matching node N2 in list2
                        Exit the algorithm with a match failure status if no match is found
                        If match is found, add N2 to list2temp
                        Mark N1 and N2 as visited
                } // End for loop
                For each node N1 in list1{
                        Get the neighbors of N1 and add each neighbor that is not already visited to
                        list3
                } // End for loop
                For each node N2 in list2temp{
                        Get the neighbors of N2 and add each neighbor that is not already visited to
                        list4
                } // End for loop
                Clear list1 and list2 and copy nodes from list3 to list1 and list4 to list2
                Matching process is completed if list1 does not have any nodes
                Repeat while loop
        } // End while loop
        Breadth-first match returns success
} // End if statement
Else {
        Breadth-first match returns failure
} // End else statement
```

Algorithm 5-1: Breadth-first match between two graphs

The above algorithm utilizes a node-matching algorithm to determine whether two nodes are identical or not. The node matching algorithm that is used for matching two BR-Nodes N_1 and N_2 is as follows:

```
// Node type is either FACE or EDGE
If (node type of N1 is not equal to node type of N2 OR number of neighbors of N1 is not equal to the
number of neighbors of N2){
        Exit the algorithm with a match failure
} // End if statement
If (node type of N1 is EDGE){
        If (geometry of edge of N1 is not the same as the geometry of edge of N2 OR nature of edge
        of N1 is not equal to the nature of edge of N2)
                Exit the algorithm with a match failure
} // End if statement
If (node type of N1 is FACE){
        If (geometry of face of N1 is not the same as the geometry of face of N2 OR type of edges in
        face of N1 is not the same as the type of edges in face of N2)
                Exit the algorithm with a match failure
} // End if statement
Node-match algorithm returns success
```

Algorithm 5-2: BR-Node matching

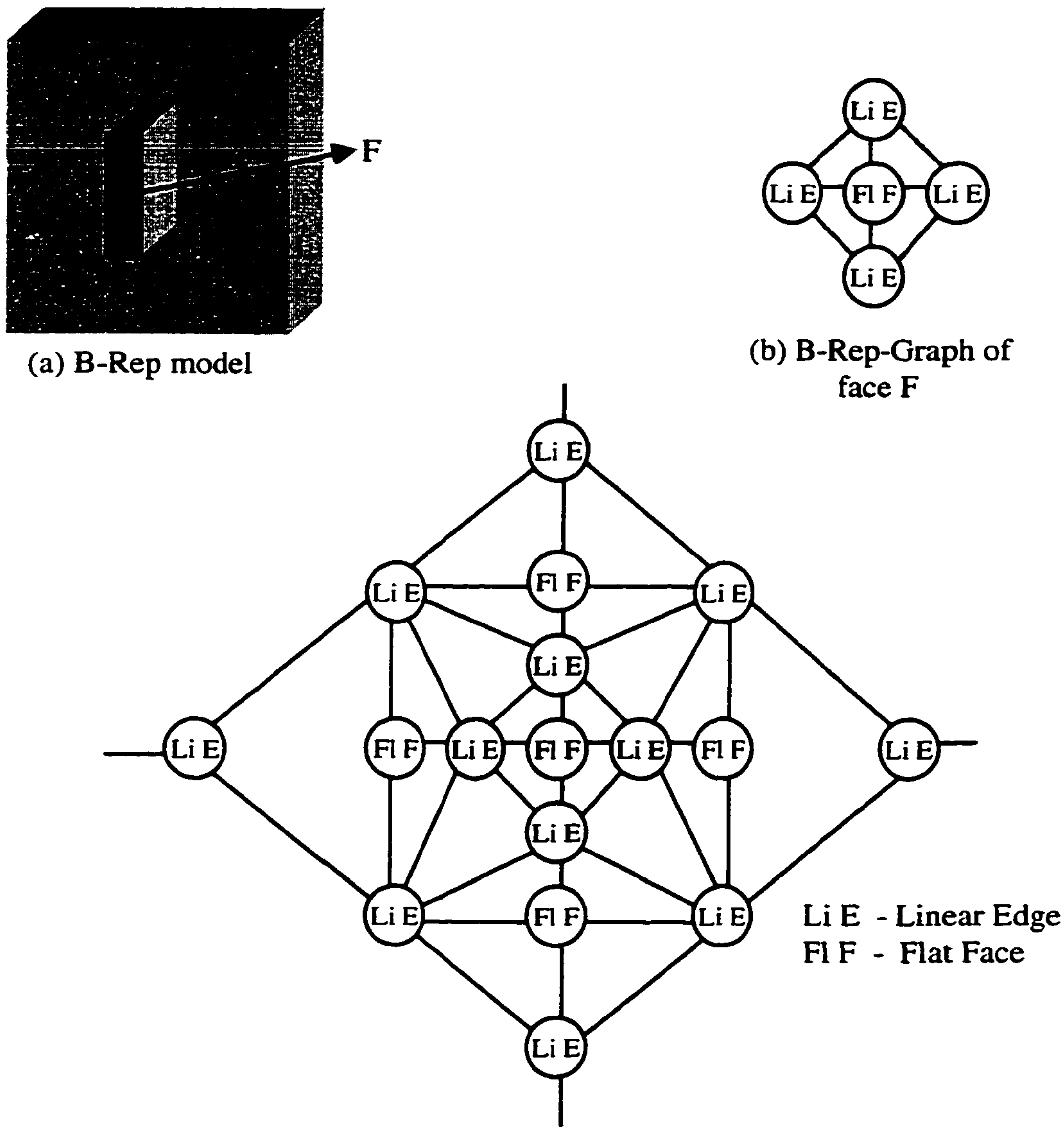
The Breadth-first matching algorithm that is described in Algorithm 5-1 is utilized in feature recognition algorithm, which is as follows:

Note: The part B-Rep-Graph is referred to as partBRGraph and the feature B-Rep-Graph is referred to as featBRGraph.

```
Filter out that nodes in the partBRGraph that are not of the same type as the nodes of featBRGraph
and create a new graph called filteredpartBRGraph.
For each node N in featBRGraph that is not already visited {
        For each node N1 in filteredpartBRGraph that is not already visited {
                // Two br-nodes match with each other if, the nodetype, geometry type,
                // connectivities (number of neighbors) of the two nodes are identical
                If (node N matches with node N1){
                        Evaluate the attribute values on node N1 if there are any attributes attached
                        to node N
                        Perform a constraint check on the evaluated attributes if there are any
                        constraints in terms of the attributes.
                        If (constraint check passes){
                                Set the visited flags of nodes N and N1
                                Do a breadth-first match in the filteredpartBRGraph starting from
                                nodes N and N1 in featBRGraph and filteredpartBRGraph
                                respectively. If there is a failure in the breadth-first match, go to the
                                filteredpartBRGraph for loop and start from another node.
                                Store all the matched nodes of filteredpartBRGraph in a new
                                BRGraph called recognizedfeatureBRGraph.
```

```
                    Reset the visited flags of nodes N and N1 if there is a failure in the
                    bread-first match.
                }
            } // End if statement
        } // End for loop
} // End for loop
```

Algorithm 5-3: B-Rep-Graph feature recognition



(a) B-Rep model

(b) B-Rep-Graph of face F

(c) Partial B-Rep-Graph

Figure 5-2: B-Rep-Graph match example

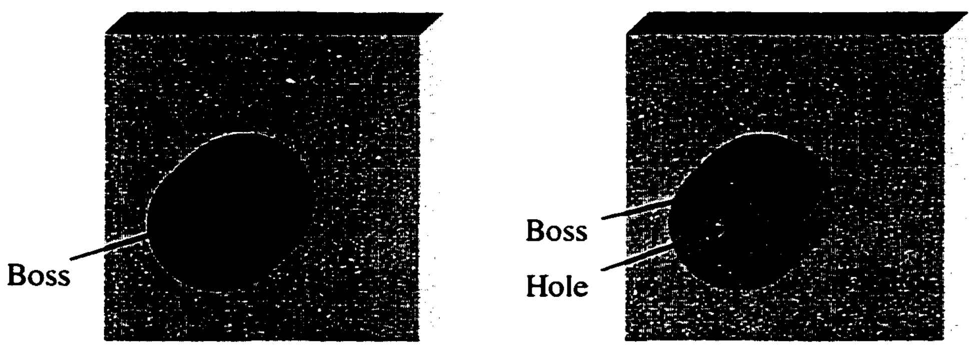
After Algorithm 5-3 completes, all the recognized features (represented as a BR-Graph called recognizedfeatureBRGraph in the above algorithm) are returned in an array. The required information relating to the attributes and the constraint checks of each recognized feature is also returned in an array of feature parameters.

The B-Rep-Graph match is illustrated pictorially in Figure 5-2. The graph in Figure 5-2(c) is the partial B-Rep-Graph of the model in Figure 5-2(a). Assume that the defined feature contains the single face F. The B-Rep-Graph of the feature (i.e., face F) is shown in Figure 5-2(b). One of the results obtained after the application of the graph match algorithm is shown in Figure 5-2(c).

5.2 Feature Recognition Algorithm for CR-Graph-match

The CR-Graph-match algorithm is used when a feature is defined in terms of a CR-Graph or a Primitive Shape. When a feature is defined in terms of a CR-Graph, depending on the type of sub-graph-matching algorithm that is specified in the definition, the CR-Graph of the feature is matched against the CR-Graph of the entire part or the CR-Graphs of the Primitives. When a feature is defined in terms of a Primitive Shape the CR-Graph of the feature is matched against the CR-Graph of the Protrusions/Depressions that are determined using the algorithms in Chapter 2. As in B-Rep-Graph-match, this matching algorithm is also based on the concept of a breadth-first search (BFS) of a graph [1]. Although, the basic recognition algorithm is identical to the B-Rep-Graph recognition algorithm, the node-matching algorithms are different. Depending on the type of feature that is defined, three different node-matching algorithms are utilized: 1) Exact Node Match, 2) Inexact Node Match, and, 3) Approximate Node Match.

The reason for having different type of node-matching algorithms is that features with variations in geometry and topology cannot be recognized using only the exact node-match algorithm. For example, if the definition is based on the boss feature that is shown in Figure 5-3(a), the exact node-matching determines that the Boss features in Figure 5-3(a) and Figure 5-3(b) are different. However, inexact matching algorithm determines that the boss feature (with the hole inside the boss top face) in Figure 5-3(b) is identical with the boss in Figure 5-3(a). Therefore, inexact node matching is utilized to determine composite features using a simple feature definition.

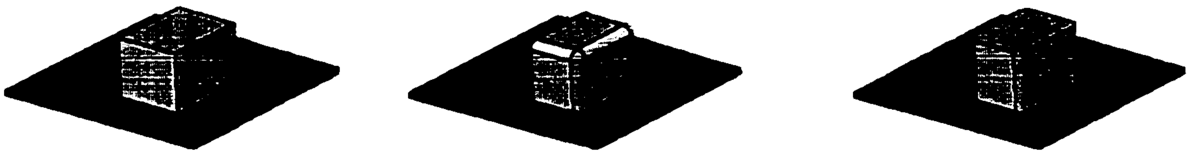


(a) Simple feature (b) Composite feature

Figure 5-3: Inexact node matching

Another example of three rib features that vary slightly in geometry and topology is shown in Figure 5-4. In this case, the feature recognition algorithm using exact node match recognizes the rib features in Figure 5-4(a) and Figure 5-4(b) using the same feature definition. However, the rib feature in Figure 5-4(c) is not recognized. The number and type of Curvature Regions for all the three rib features is identical. The reason the rib feature in

Figure 5-4(c) is not recognized is that, in the Curvature Region Representation of the rib in Figure 5-4(c), the top face is connected to eight nodes in contrast to four nodes for ribs in Figure 5-4(a) and Figure 5-4(b). If an approximate node-matching algorithm is used all the three rib features are recognized using a single feature definition.



(a) Rib feature (b) Rib feature with top and side edges filleted (c) Rib feature with side edges filleted

Figure 5-4: Approximate node matching

The exact, inexact and approximate node-matching algorithms are described in the following sections.

5.2.1 Exact Node Match

Exact node matching is utilized when the feature definition type is CRGRE. The CR-Graph of the feature is matched against the CR-Graph of the entire part. The node-matching algorithm that is used for matching two CR-nodes N_1 and N_2 exactly is as follows:

```
If (node type of N1 is not equal to node type of N2 OR number of neighbors of N1 is not equal to the
number of neighbors of N2){
        Exit the algorithm with a match failure
} // End if statement
Node-match algorithm returns success
```

Algorithm 5-4: Exact CR-Node matching

5.2.2 Inexact Node Match

Inexact node matching is utilized when the feature definition type is CRGRI. The CR-Graph of the feature is matched against the CR-Graph of the entire part. The algorithm for matching CR-nodes N_1 and N_2 in an inexact manner is as follows:

```
If (node type of N1 is not equal to node type of N2){
        Exit the algorithm with a match failure
} // End if statement
If (N2 corresponds to a face) { // N2 is a node in the part CR-Graph
        Remove the CR-Nodes of edges and vertices that lie in the interior of the face from the
        neighbor list of N2
} // End if statement
If (number of neighbors of N1 is not equal to the number of neighbors of N2){
        Exit the algorithm with a match failure
} // End if statement
Node-match algorithm returns success
```

Algorithm 5-5: Inexact CR-Node matching

5.2.3 Approximate Node Match

Approximate node matching is utilized when the feature definition type is CRGRA. The CR-Graph of the feature is matched against the CR-Graph of the entire part. Approximate node matching is also used when a feature is defined in terms of a Primitive Shape. In which case, the CR-Graph of the feature is matched against the CR-Graph of the Protrusions/Depressions. The algorithm for matching CR-nodes N_1 and N_2 approximately is as follows:

```
If (node type of N1 is not equal to node type of N2){
        Exit the algorithm with a match failure
} // End if statement
// N1 is a node in the feature CR-Graph and N2 is a node in the part CR-Graph
If (number of neighbors of N1 greater than the number of neighbors of N2){
```

```
        Exit the algorithm with a match failure
} // End if statement
Node-match algorithm returns success
```

Algorithm 5-6: Approximate CR-Node matching

This concludes the description of the feature recognition algorithms that are used in the current research. The results obtained on some sample parts after applying the Feature Definition and Recognition techniques described thus far are provided in the next Chapter.

6 Results

The results are described into two sections. The first section describes the results for feature definition and the second section described the results for feature recognition.

6.1 Results for Feature Definition

6.1.1 Non-template Feature Definition

Several features have been defined using the non-template user interface to the Feature Definition Language. The feature definitions are shown in Appendix B. A summarized table of the number of nodes, attributes and constraints of the features defined in Appendix B is shown in Table 6-1. The time taken to define the features is also shown in the table. It has been observed that the ease of determining the connectivities between the nodes and placement of the nodes in the graph decreases with the increase in the number of nodes in a definition.

Another factor influencing the ease of creation of a definition is the configuration of the graph. For example, the number of nodes in both the CORNERSLOT and the POCKET are 25. The time taken to define a CORNERSLOT is 460 seconds and the time taken to define a POCKET is 240 seconds. This discrepancy in the time taken to define the two features is due to the fact that the configuration of the graph of the CORNERSLOT is more complex than that of the POCKET. Moreover, for a novice user, determining the Curvature Regions corresponding to a B-Rep entity (face, edge or vertex) is a time consuming process.

It has been determined empirically that the non-template user interface is convenient to use only when the number of nodes in a model is less than 15. For example, the time taken to define the feature PROTRIB1 is 15 seconds. This feature has been defined as a Protrusion Shape that contains four [0,0] nodes. Therefore, the template feature definition interface is better suited to define features when compared to the non-template definition user interface.

Feature Name	N_{nodes}	$N_{attributes}$	$N_{constraints}$	T (seconds)
BLINDHOLE	6	4	0	128
BOSS	6	2	2	120
BOSS1	10	5	1	209
BUTTON	6	4	1	83
CORNERSLOT	25	3	0	460
HOLE	3	3	0	60
POCKET	25	7	0	240
PROTRIB	2	4	1	63
PROTRIB1	4	0	0	15
RIB1	21	7	2	323
SLOT1	17	4	0	144
SLOT2	11	3	0	107
SLOT3	21	3	0	225
WEB	15	4	0	124

N_{nodes} Number of Nodes in the Feature Definition

$N_{attributes}$ Number of attributes in the Feature Definition

$N_{constraints}$ Number of constraints in the Feature Definition

T Time taken to define the feature

Table 6-1: Example Feature Definitions

6.1.2 Template Feature Definition

In contrast to non-template feature definition, it is easier to create the above definitions using a template user interface. In a template definition, features are defined by selecting entities on the CAD model, attaching attributes to the entities and specifying constraints. Therefore, it is not required for a user to specify the nodes in a definition. A snapshot of the template definition tool that has been incorporated into the ProEngineer® CAD system is shown in Figure 6-1.

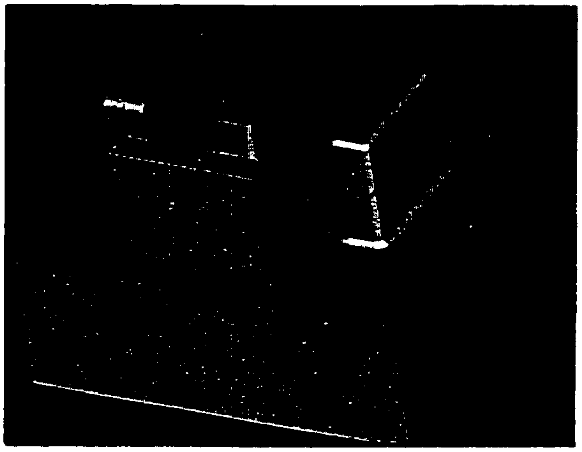


Figure 6-1: Template Definition Tool for ProEngineer® CAD System

In the above definition, the selection type is set to "Protrusion" and only the top face of the rib feature is selected. Subsequently, the entire protrusion shape that contains the selected face is highlighted. The highlighted feature is used as a template to define a rib feature. The CR-Graph of the template is generated automatically and the rib feature definition is then stored to a file. The definition that is output to a file is shown in Figure 6-2. The feature name is RIB and the definition type is CRGRA and there are 26 nodes in the graph. Although the rib feature that is selected in Figure 6-1 contains fillets, the definition of the rib feature with fillets and without fillets is the same due to the property of the Simplified CR-Graph (described in section on Simplified CR-Graph in Chapter 2).

```
RIB CRGRA
CRNODES{
    26
    2 4 9 13 14 10
    2 4 9 11 12 2
    5 1 1
    2 4 10 15 11 4
    5 1 3
    2 4 13 12 16 6
    5 1 5
    2 4 14 16 15 8
    5 1 7
    3 4 0 1 17 18
    3 4 0 3 19 17
    3 4 3 1 17 21
    3 4 1 5 18 23
    3 4 0 5 18 20
    3 4 0 7 20 19
    3 4 7 3 19 24
    3 4 5 7 20 25
    4 3 9 10 11
    4 3 9 12 13
    4 3 10 14 15
    4 3 14 13 16
    6 4 22 11 2 4
    2 1 21
    6 4 22 12 6 2
    6 4 22 15 4 8
    6 4 22 16 8 6
}

ATTRIBUTES{
    0
}
CONSTRAINTS{
    0
}
```

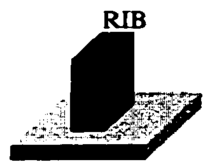


Figure 6-2: Template definition for a rib feature

Another example of a template definition is shown in Figure 6-3. In this example, a p_rib feature is defined. The selection type in template definition is set to "Protrusion" and only the top face of the p_rib feature is selected. Subsequently, the entire protrusion shape that contains the selected face highlighted. The highlighted feature is used as a template to define the p_rib feature. Using a non-template user interface for defining this p_rib feature is a time consuming task since the CRs at the edges must be determined manually, which involves considerable amount of time.

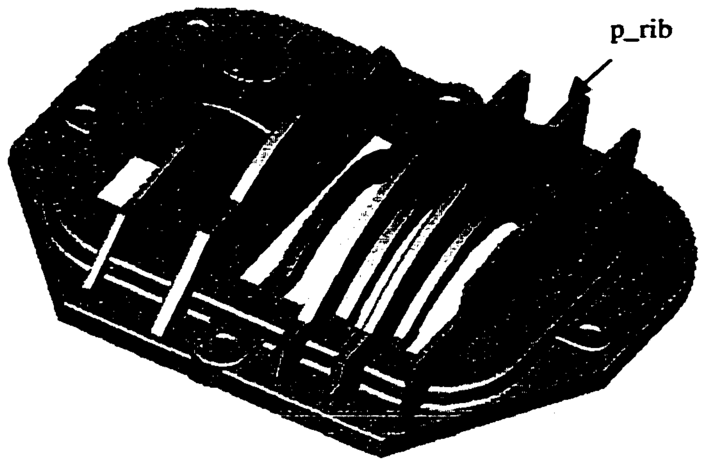


Figure 6-3: Template definition for a p_rib feature

Though the template definition tool is more convenient than the non-template definition tool, the non-template definition tool is still needed since the user may not have access to a CAD system to define the features. In which case, the non-template definition tool must be used to define the features. Moreover, the non-template feature definition tool can be used to locally edit features defined using the template interface so that local changes can be made to a definition without creating another template from a new feature.

6.2 Results for Feature Recognition

The feature recognition algorithm has been applied on several parts of varying complexity, of which 33 parts are shown in Appendix C. The time taken to compute the CR-Graph for the parts is plotted against total number of non-neutral edges and faces (N_{edges} + N_{faces}) and the resultant plot is shown in Figure 6-4. The time taken to compute the CR-Graph is of the order of the total number of non-neutral edges and faces ($O(N_{edges} + N_{faces})$). The neutral edges in the model do not contribute to the computation time of the CR-Graph.

This is due to the fact that when an edge is neutral the CRs on the edge and the CRs on the vertices of the edge are not evaluated.

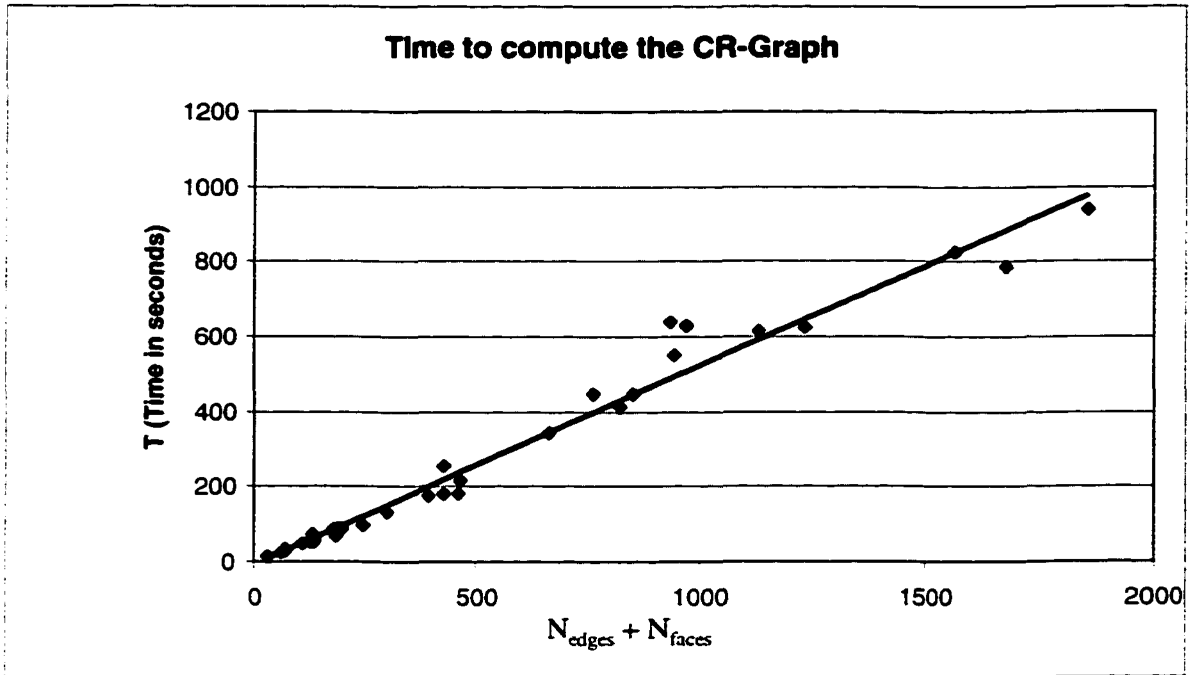
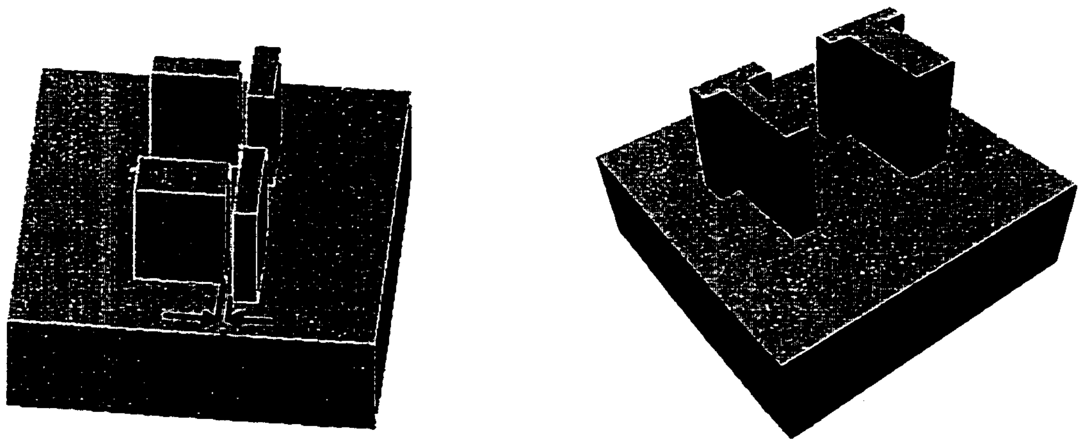


Figure 6-4: Time to compute the CR-Graph

The results of feature recognition are presented in the next few sections.

6.2.1 Recognizing features that do not exist in the design database

The primary reason for the requirement of feature recognition is that the required feature information is not available in the design database. To illustrate this, the result of feature recognition performed on an example part is presented in this section. Consider the part shown in Figure 6-5. The part in Figure 6-5(b) has been created in a Parametric Modeler by setting the distance “d” between the rib features shown in Figure 6-5(a) to zero. As a result, the modeling system contains information only about the four rib features and does not contain any information about the two T-Rib features.



(a) Part before T-Rib creation (b) Part after T-Rib creation

Figure 6-5: Object containing a T-Rib feature

To obtain the T-Rib feature information for the above part, first, a T-Rib feature is defined by selecting the top face of the T-Rib feature in the template definition tool. The template is subsequently saved to a file with the feature definition type set to Exact CR-Graph (CRGRE) as shown in Figure 6-6. Second, feature recognition is performed on the part by using the feature definition of the T-Rib features. After feature recognition, two T-Rib features are recognized and their bounding boxes are computed and the result is shown in Figure 6-7.

```
t-rib CRGRE
CRNODES{
    47
    2 8 19 22 23 25 27 28 30 20
    2 4 19 21 2 3
    5 1 1
    5 1 1
    2 4 21 32 20 5
    5 1 4
    2 4 24 22 7 2
    5 1 6
    2 4 26 23 24 9
    5 1 8
    2 4 26 25 11 12
    5 1 10
    5 1 10
    2 4 29 27 12 14
    5 1 13
    2 4 29 31 28 16
    5 1 15
    2 4 31 32 30 18
    5 1 17
    3 4 0 1 33 39
    3 4 0 4 33 38
    3 4 4 1 33 40
    3 4 0 6 34 39
    3 4 0 8 35 34
    3 4 6 8 34 41
    3 4 0 10 35 42
    3 4 10 8 35 43
    3 4 0 13 36 42
    3 4 0 15 37 36
    3 4 15 13 36 44
    3 4 0 17 38 37
    3 4 17 15 37 45
    3 4 4 17 38 46
    4 3 19 20 21
    4 3 22 23 24
    4 3 23 25 26
    4 3 27 28 29
    4 3 28 30 31
    4 3 30 20 32
    6 3 19 22 2
    6 3 21 3 5
    6 3 24 7 9
    6 3 25 27 12
    6 3 26 11 9
    6 3 29 16 14
    6 3 31 18 16
    6 3 32 5 18
}
ATTRIBUTES{
    0
}
CONSTRAINTS{
    0
}
```

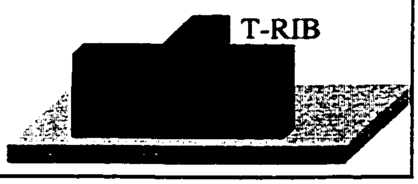


Figure 6-6: T-Rib feature definition

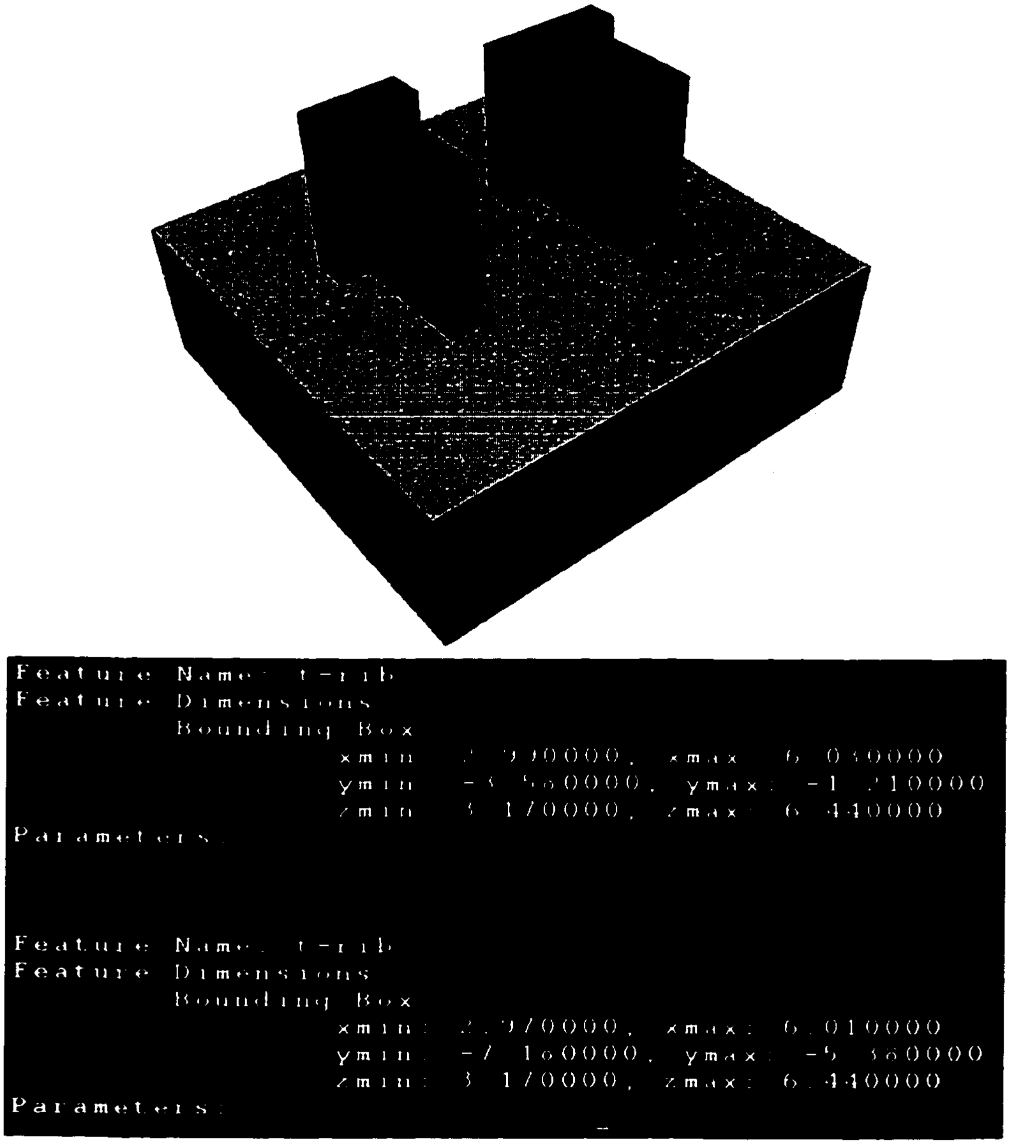


Figure 6-7: Recognized T-Rib features

The next section presents several example results of feature recognition on more complex parts.

6.2.2 Sample feature recognition results

As described earlier, the feature definitions that are in terms of Curvature Regions are more detailed than the definitions that are in terms of Primitive Shapes and feature definitions that are in terms of B-Rep elements are more detailed than the definitions that are in terms of Curvature Regions. The amount of feature information that is obtained through feature recognition is, therefore, dependent on the type of feature definition. The current section presents some feature recognition results that are obtained by utilizing the three different types of feature definition. An example result for the B-Rep based feature definition is shown in Figure 6-8. Feature recognition is performed on the part named Team in Appendix C. A boss feature definition in terms of a B-Rep-Graph is utilized in the recognition algorithm.

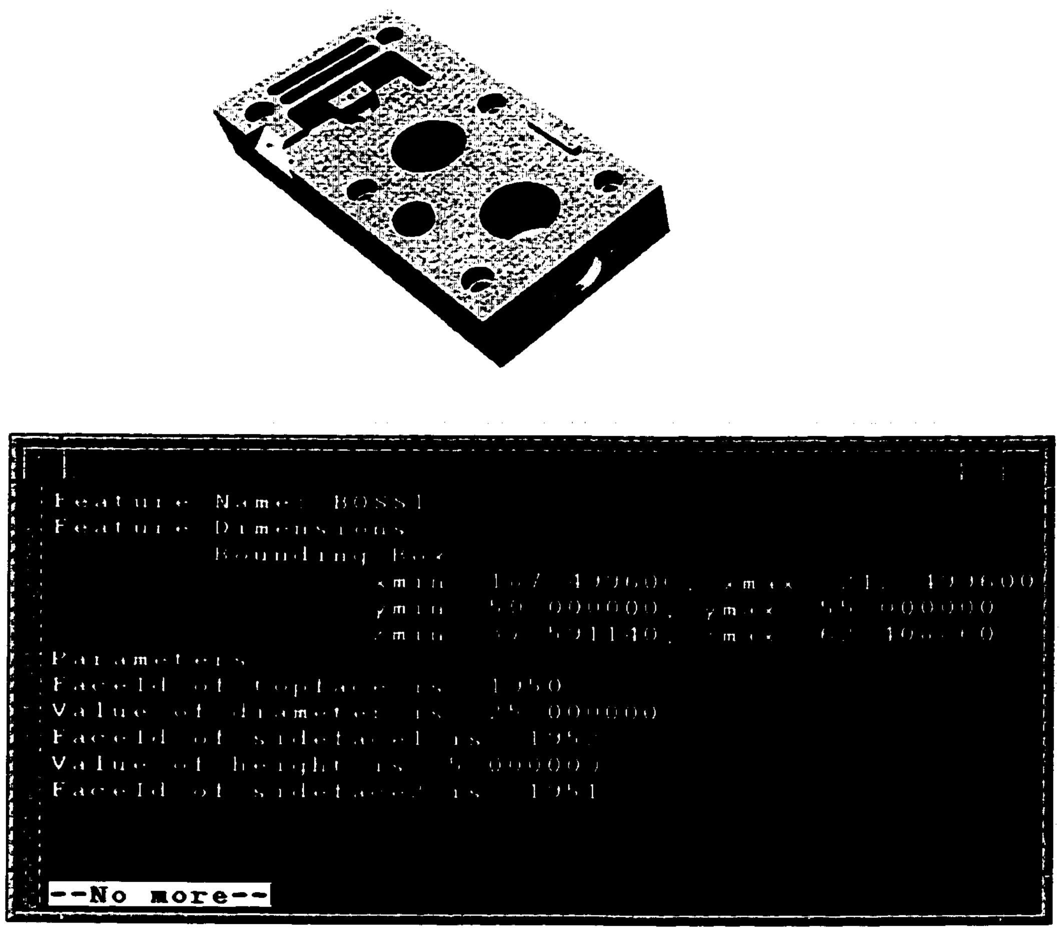
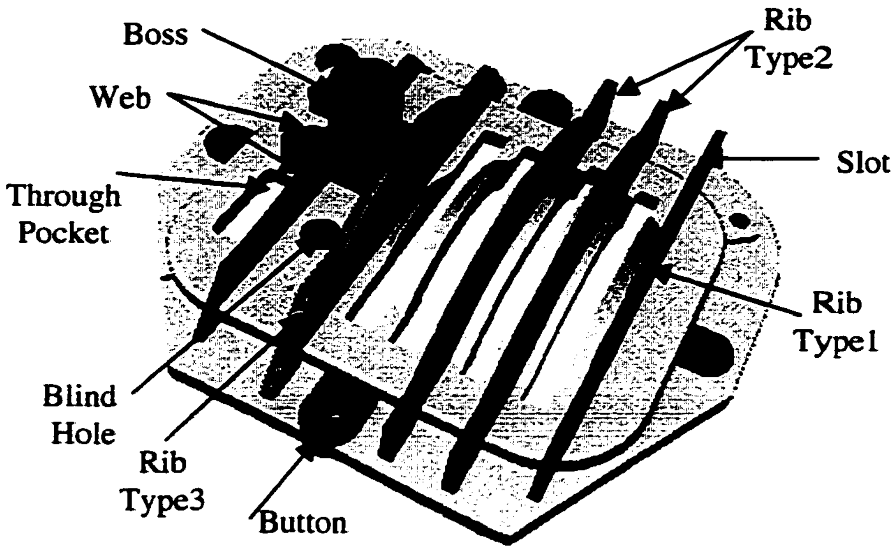
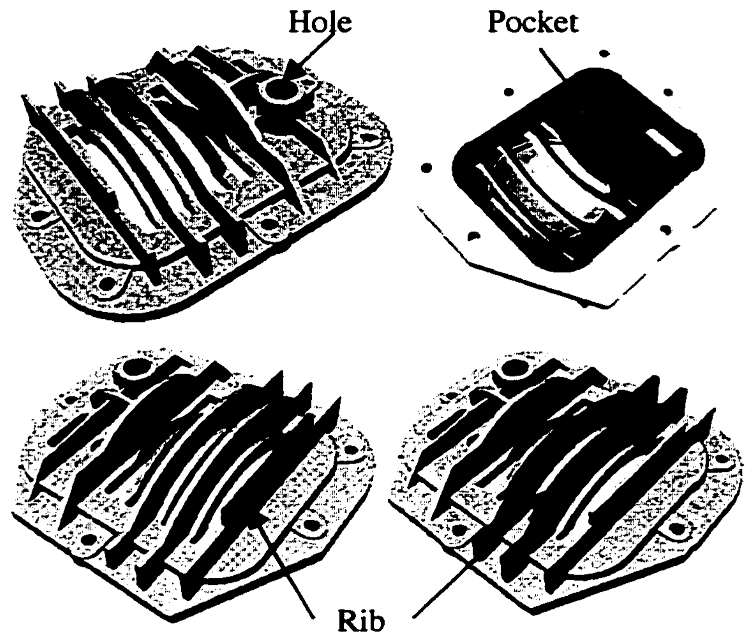


Figure 6-8: A test result for a B-Rep graph definition

Example results of feature recognition using feature definitions in terms of Curvature Regions and Primitive Shapes are shown in Figure 6-9. Several definitions of a rib feature are used in this example. Firstly, a rib feature is defined as a Primitive Shape (protrusion) containing two parallel flat faces (same as Feature 8 in Appendix B). Secondly, three types of rib features are defined in terms of Curvature Regions. The other feature definitions that are used in the current example are boss, web, blind-hole, button, slot and through-pocket.



(a)



(b)

Figure 6-9: Results for Curvature Region and Primitive Shape based definitions

The rib features obtained using Primitive Shape feature definition are shown in the Figure 6-9(b). The recognized features are complex in the sense that the rib features have varying topology and geometry and the base on which the rib features lie is not a single surface. The other features obtained using feature definitions in terms of Primitive Shapes are the hole and pocket features. The hole feature is defined as a depression containing a [0,-] region and the Pocket feature is defined as a depression containing five flat faces.

The features obtained using Curvature Region based feature definitions are shown in Figure 6-9(a). The button and boss features are differentiated based on the Constraint specification on the height to diameter ratio for boss and button features as in Appendix B.

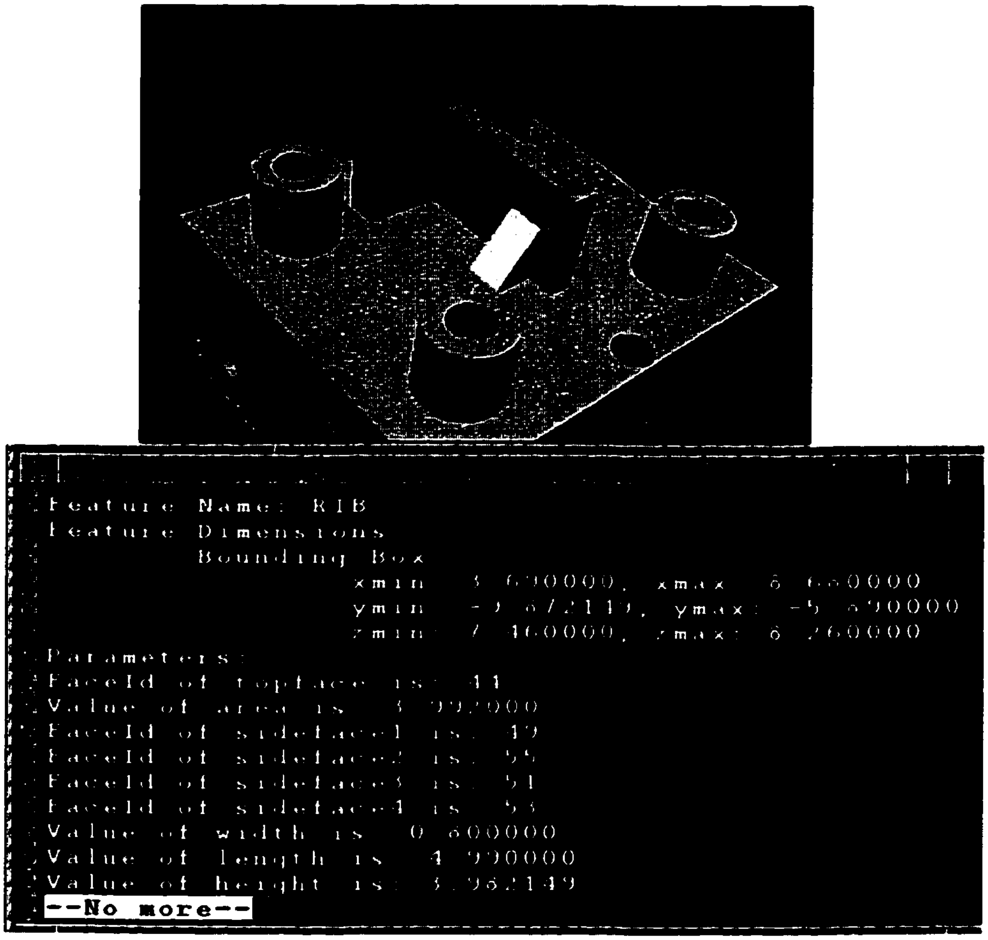


Figure 6-10: Rib feature

Figure 6-10 and Figure 6-11 show test results in which the dimensions of the features obtained are also shown. The dimensional attributes that are defined for the rib feature are the width, length and height. The dimensional attributes defined for the pocket feature are length, width and depth.

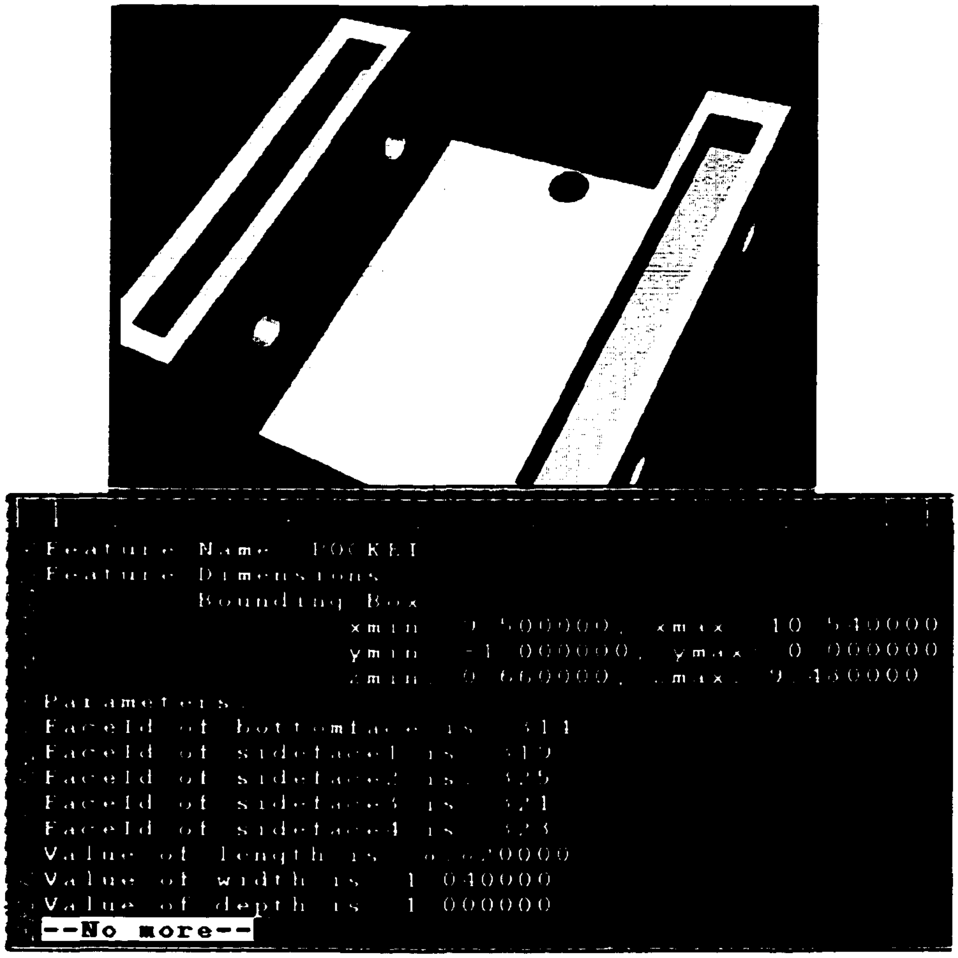


Figure 6-11: Test results

The next few sections present example results for the following claims that have been made about the current research:

a) Based on the type of feature definition, the feature recognition allows topological and geometric variations of a feature to be recognized with a single feature definition.

b) The user has a choice to define features any application domain. Subsequent to their definition, the features can be recognized from a part.

Example results are presented for the application domains of machining and molding.

6.2.3 Features varying in Geometry and Topology

As mentioned earlier, in the current research, features that vary slightly in geometry and topology are recognized using a single feature definition. Consider the Prt6 in Appendix C, an enlarged image of which is shown in Figure 6-12. This part contains 5 rib features that are slight variations of each other. All the 5 rib features are recognized using the definition for a rib feature that shown in Figure 6-2.

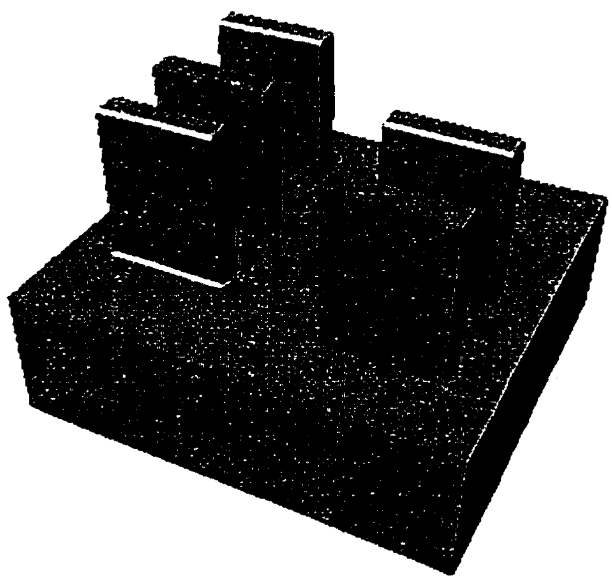
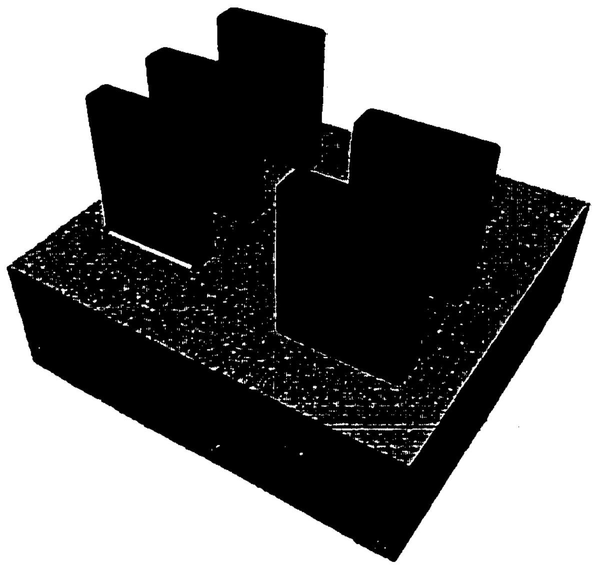


Figure 6-12: Part with features that vary in geometry and topology



Feature Name: RIB
Feature Dimensions
Bounding Box
xmin: 6.360000, xmax: 9.060000
ymin: -7.710000, ymax: -7.020000
zmin: 3.440000, zmax: 7.300647
Parameters:

Feature Name: RIB
Feature Dimensions
Bounding Box
xmin: 6.360000, xmax: 9.060000
ymin: -5.040000, ymax: -4.350000
zmin: 3.440000, zmax: 7.300647
Parameters:

Feature Name: RIB
Feature Dimensions
Bounding Box
xmin: 1.050000, xmax: 3.750000
ymin: -2.330000, ymax: -1.640000
zmin: 3.440000, zmax: 7.300647
Parameters:

Feature Name: RIB
Feature Dimensions
Bounding Box
xmin: 1.050000, xmax: 3.750000
ymin: -5.000000, ymax: -4.310000
zmin: 3.440000, zmax: 7.300647
Parameters:

Feature Name: RIB
Feature Dimensions
Bounding Box
xmin: 0.950000, xmax: 3.850000
ymin: -7.120000, ymax: -6.230000
zmin: 3.440000, zmax: 7.300647
Parameters:

Figure 6-13: Recognition of features varying in geometry and topology

The result of feature recognition using this single definition is shown in Figure 6-13. The bounding boxes of the 5 RIB features are computed during feature recognition. One of the filleted ribs does not contain the bottom edge fillets because the CRs corresponding to the bottom edge fillets are of the type [+,0]. The rib feature definition that is used in this example does not contain any [+,0] nodes. Therefore, the bottom edge fillets are not matched during feature recognition.

6.2.4 Machining Features

Machining features are features such as slot and pocket that are required to analyze a part for machinability. Machining features are created through volume removal during the machining process and hence are depression features. The machining features that are considered in the next few examples are Slot, Open Slot, Through Slot, Pocket, Through Pocket, Hole, Blind Hole and Countersink Hole (shown in Figure 6-14).

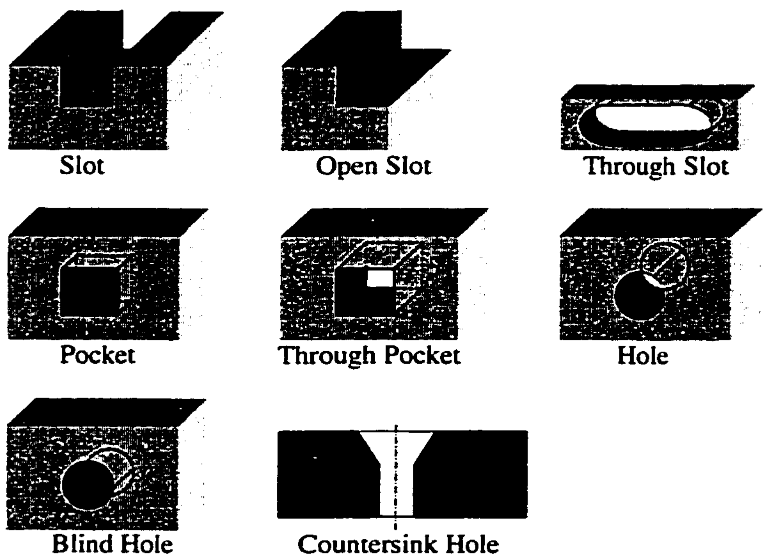


Figure 6-14: Machining Features

The above machining features have been defined via both the template user interface and the non-template user interface and the feature definitions are similar to those shown in Appendix B. The parts in Appendix C that have been tested for machining features are

Bulkhead, Conrod, Frame, Htoolbase, Parker, Piston, Team and Toolhold. The result of feature recognition on these parts is shown in Figure 6-15.

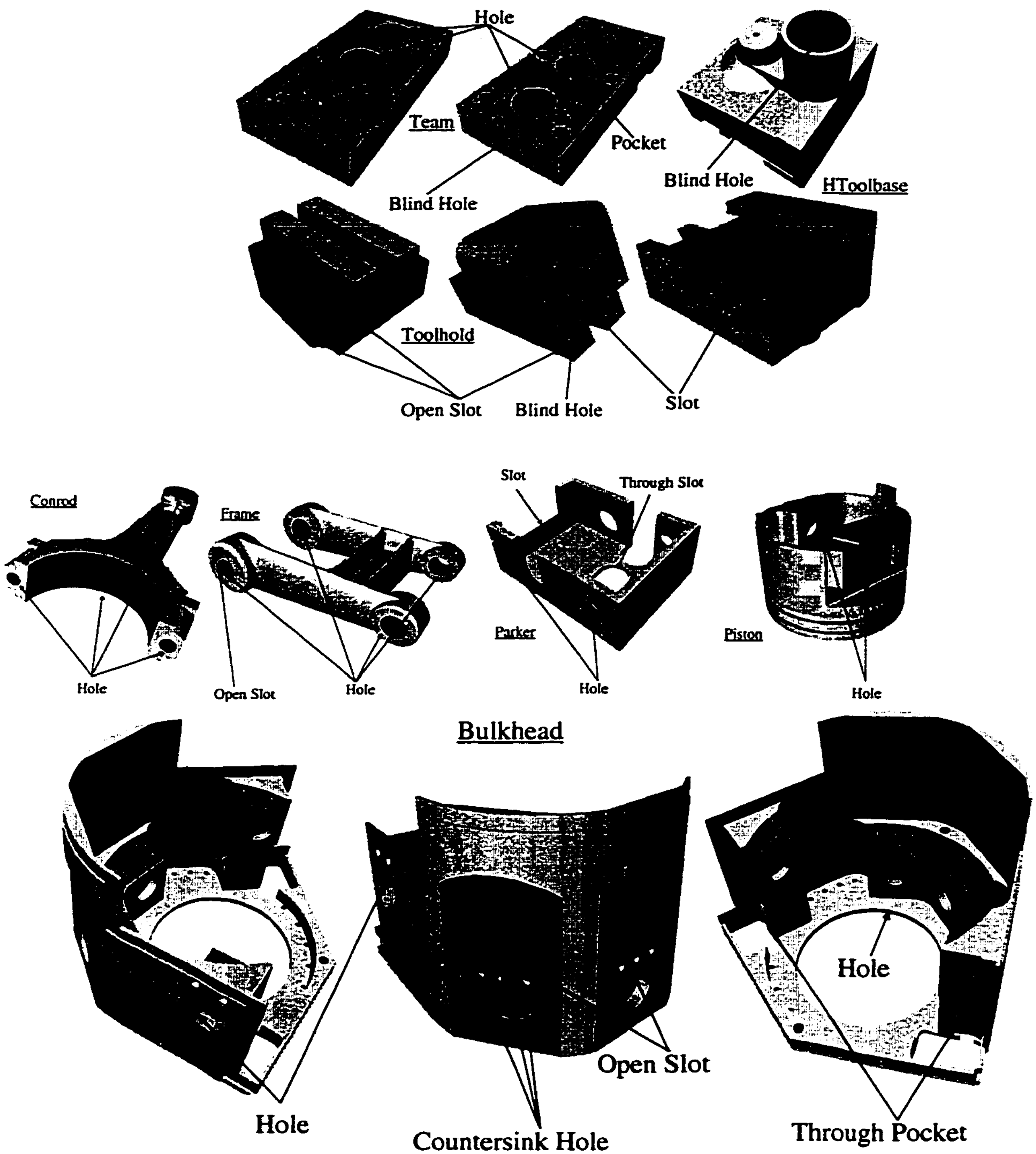


Figure 6-15: Recognized Machining Features

6.2.5 Molding/Casting Features

Molding features are features such as rib, boss and groove that are required to analyze a part for moldability. Unlike machining features injection molding features are both protrusion and depression features. However, only protrusion features are considered in the next few examples since examples for depression features have already been shown in the previous section. The molding features that are used in this section are Boss, Web, Button, Fin and several types of Rib features.

The molding features have been defined via both the template user interface and the non-template user interface and the feature definitions are similar to those shown in Appendix B. The parts in Appendix C that have been tested for molding features are Abscover, Approx, Coverc, Gadh1, Housing and Pmtest. The result of feature recognition on these parts is shown in Figure 6-16.

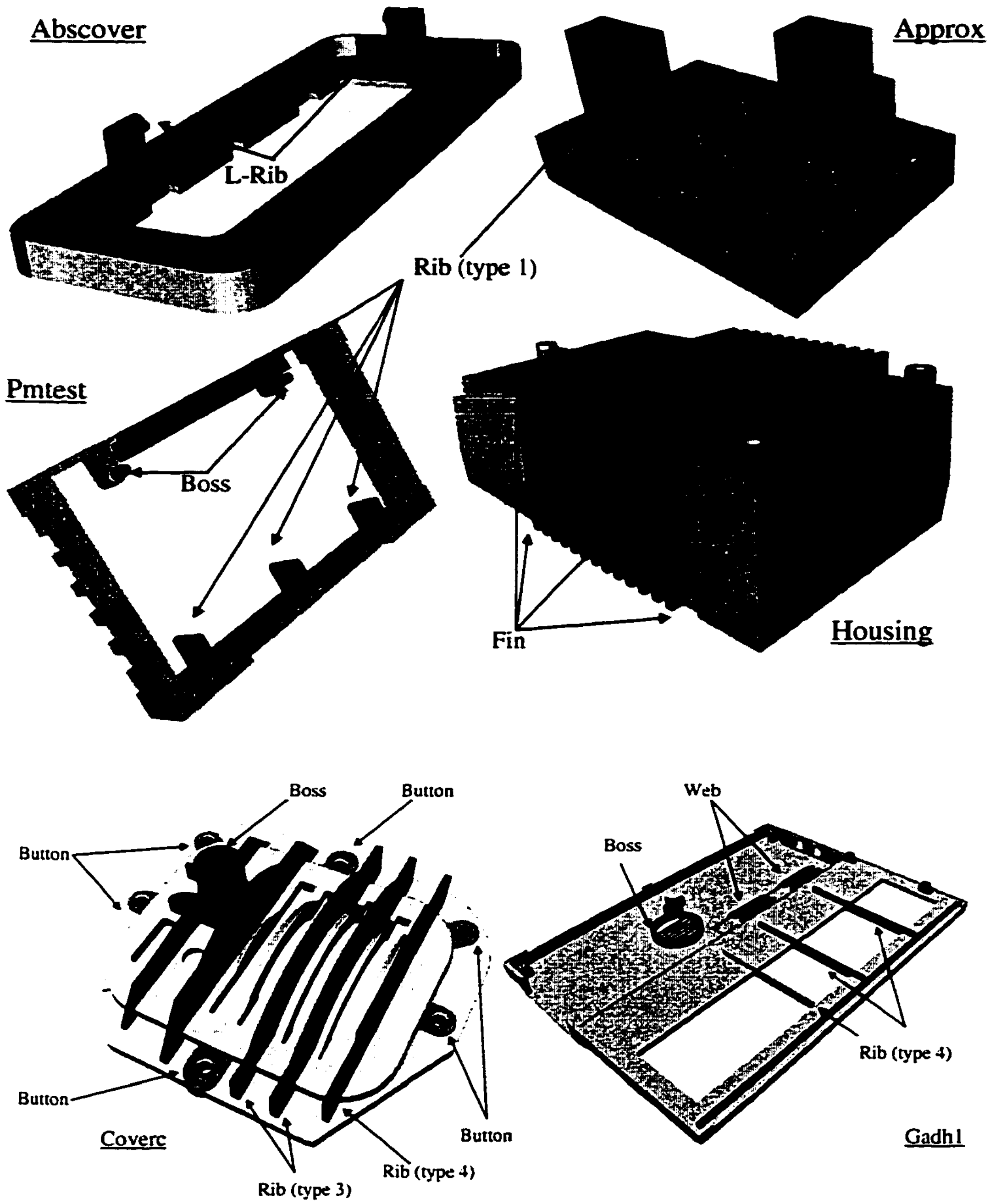


Figure 6-16: Recognized Molding Features

The current approach can also be used for blend feature recognition and feature definition and design rule integration. The next two sections present example results in these two categories.

6.2.6 Blend Feature Recognition

The native CAD model format in most commercial CAD systems is proprietary. As a result, a part designed in one CAD system (e.g., ProEngineer) cannot be used in another CAD system (e.g., SDRC IDEAS) unless the part is exported in a neutral file format (such as STEP and IGES) that both CAD systems understand. However, most neutral file formats allow only the B-Rep data to be saved and not the feature information. Therefore, whenever a model is created in one CAD system (System 1) and it is modified in another CAD system (System 2), the feature information has to be extracted from the model in System 2. During design modification, one of the operations that is performed is deletion/suppression of blends on the part. Subsequent to the blend deletion/suppression the rest of the model is modified. Therefore, to perform a design modification, the blend features on the model must be recognized. The current approach can be utilized to perform blend feature recognition on a part.

The blend features are defined using Curvature Regions. All the Curvature Regions except the flat regions may correspond to a blend face. Therefore, five types of blend features are defined -- each definition corresponding to one curvature type. The blend feature definitions for the five types are shown in Figure 6-17. Each blend feature is defined as a CR-Graph containing one single CR node and constraints are attached to the node. The constraints that are attached to the node are as follows:

a) the B-Rep entity corresponding to the node is a face

b) the diameter of the face is less than a threshold value.

```
blendfeature1 CRGRA
CRNODES{
    1
    3 0
}
ATTRIBUTES{
    2
    d FACEDIAMETER 0
    isface0 ISFACE 0
}
CONSTRAINTS{
    2
    isface0 equal-to 1.0
    d < 1.5
}
```

```
blendfeature2 CRGRA
CRNODES{
    1
    4 0
}
ATTRIBUTES{
    2
    d FACEDIAMETER 0
    isface0 ISFACE 0
}
CONSTRAINTS{
    2
    isface0 equal-to 1.0
    d < 1.5
}
```

```
blendfeature3 CRGRA
CRNODES{
    1
    5 0
}
ATTRIBUTES{
    2
    d FACEDIAMETER 0
    isface0 ISFACE 0
}
CONSTRAINTS{
    2
    isface0 equal-to 1.0
    d < 1.5
}
```

```
blendfeature4 CRGRA
CRNODES{
    1
    6 0
}
ATTRIBUTES{
    2
    d FACEDIAMETER 0
    isface0 ISFACE 0
}
CONSTRAINTS{
    2
    isface0 equal-to 1.0
    d < 1.5
}
```

```
blendfeature5 CRGRA
CRNODES{
    1
    8 0
}
ATTRIBUTES{
    2
    d FACEDIAMETER 0
    isface0 ISFACE 0
}
CONSTRAINTS{
    2
    isface0 equal-to 1.0
    d < 1.5
}
```

Figure 6-17: Blend feature definitions

The result obtained after using the above feature definitions for feature recognition on two sample parts is shown in Figure 6-18. A diameter threshold of 1.5 has been used for the parts shown in Figure 6-18. To use the above definitions on other parts, the diameter threshold value in the feature definitions must be altered based the size of the part.

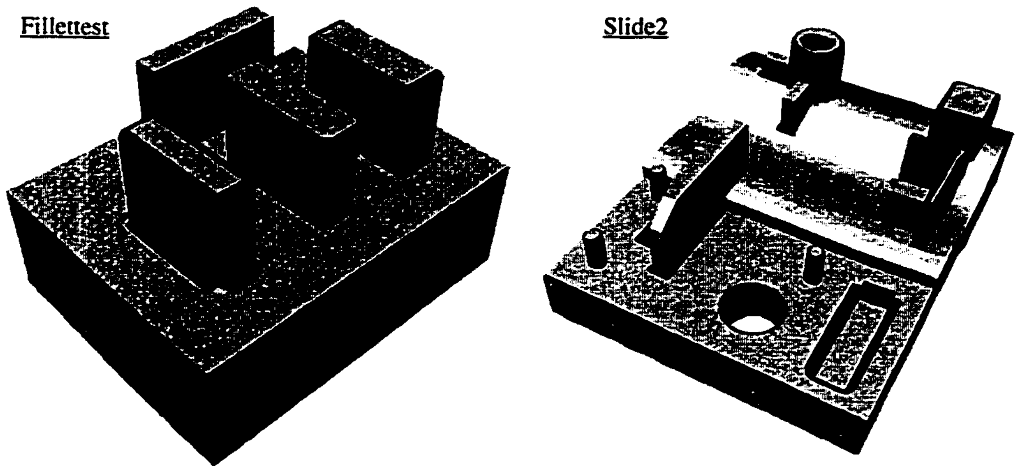


Figure 6-18: Blend feature recognition

6.2.7 Incorporating DFM Rules at Feature Definition Stage

Another advantage of the current approach is that the design-for-manufacturability rules can be incorporated at the feature definition stage. For example, as shown in Figure 6-19(a) and (b), using a design rule, acceptable and unacceptable ribs may be defined. The unacceptable ribs can be obtained right at the stage of feature recognition without the need of applying the Design Rule on the recognized features. The determination of unacceptable ribs provides the designer with an opportunity to modify the design so that the designed part is manufacturable. This capability of the Feature Definition Language to incorporate DFM rules during the definition stage allows a further integration of Design and Manufacturing phases of a product cycle.

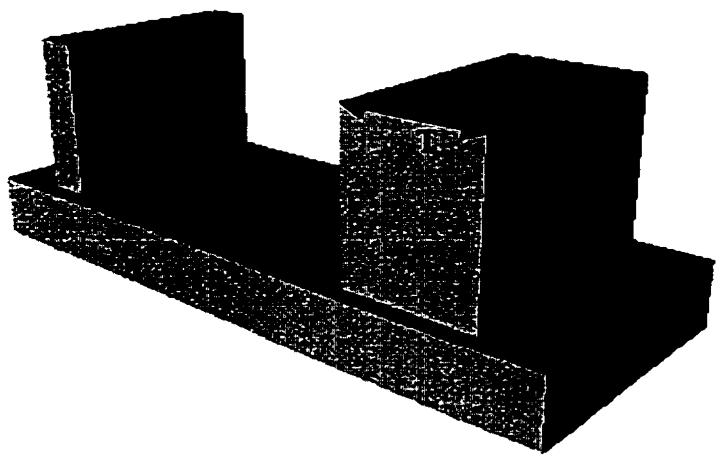
```
acceptable-rib CRGRE
CRNODES{
    26
    2 4 5 6 7 8
    2 4 10 9 8 17
    2 4 11 9 5 19
    2 4 12 10 7 20
    2 4 12 6 11 21
    3 4 0 2 13 14
    3 4 0 4 15 13
    3 4 0 3 16 15
    3 4 0 1 14 16
    3 4 2 1 14 22
    3 4 1 3 16 23
    3 4 2 4 13 24
    3 4 3 4 15 25
    4 3 5 6 11
    4 3 5 8 9
    4 3 6 7 12
    4 3 7 8 10
    5 4 18 1 22 23
    2 1 17
    5 4 18 2 24 22
    5 4 18 3 23 25
    5 4 18 4 25 24
    6 3 9 19 17
    6 3 10 17 20
    6 3 11 19 21
    6 3 12 20 21
}
ATTRIBUTES{
    1
    thickness    LENGTH 17
}
CONSTRAINTS{
    1
    thickness    <    0.800000
}
```

```
unacceptable-rib CRGRE
CRNODES{
    26
    2 4 5 6 7 8
    2 4 10 9 8 17
    2 4 11 9 5 19
    2 4 12 10 7 20
    2 4 12 6 11 21
    3 4 0 2 13 14
    3 4 0 4 15 13
    3 4 0 3 16 15
    3 4 0 1 14 16
    3 4 2 1 14 22
    3 4 1 3 16 23
    3 4 2 4 13 24
    3 4 3 4 15 25
    4 3 5 6 11
    4 3 5 8 9
    4 3 6 7 12
    4 3 7 8 10
    5 4 18 1 22 23
    2 1 17
    5 4 18 2 24 22
    5 4 18 3 23 25
    5 4 18 4 25 24
    6 3 9 19 17
    6 3 10 17 20
    6 3 11 19 21
    6 3 12 20 21
}
ATTRIBUTES{
    1
    thickness    LENGTH 17
}
CONSTRAINTS{
    1
    thickness    >=    0.800000
}
```

(a)

Design Rule

IF: The material is GE NORYL N190	
and	The input root thickness [T] is greater than 0.8
THEN:	possibility of bad sinkmark = 9/10
and	possibility of warpage = 8/10
and	Warning Message: Reduce [T] to be less than 0.8



(b)

Figure 6-19: Acceptable and unacceptable rib feature definitions

The result of feature recognition on the above part is shown in Figure 6-20.

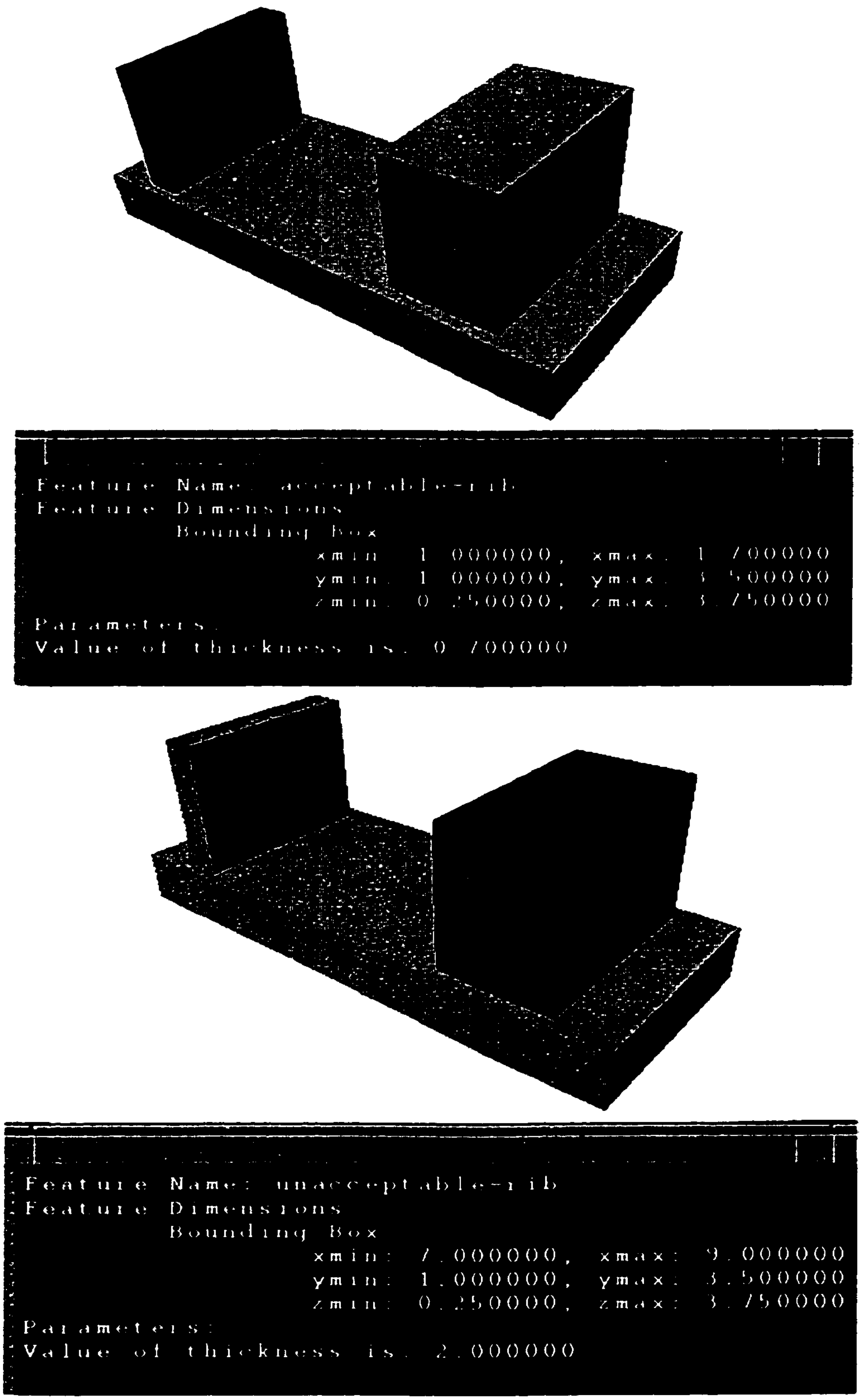


Figure 6-20: Recognized acceptable and unacceptable rib features

This concludes the Chapter on results for feature definition and feature recognition. Additional results of feature recognition from the parts in Appendix C are shown in Appendix D. Tables for the time to recognize the features using the different node-matching methods are also presented in Appendix D. The next chapter presents the Conclusion and some suggestions for future research directions.

7 Conclusion and Future Research

7.1 Conclusion

In the current research, three specific geometric abstractions, namely, B-Rep, Curvature Regions and Primitive Shapes are utilized to interactively define features and subsequently recognize features from a part.

A feature is represented using a Feature Definition Language that is in terms of the entities of the above three geometric abstractions. A front-end graphics tool is used to interactively define a feature and automatically generate the feature definition. Through the user interfaces, a user is provided with a choice to make the feature definition as detailed or as generic as required. Also, the user has a choice to define features for application domains, such as, machining or injection molding.

Subsequent to their definition, the features are recognized from a part. The algorithms that are used for feature recognition are based on the type of feature definition. The CR-Graph based feature recognition algorithm allows topological and geometric variations of a feature to be recognized with a single feature definition. Also, the features are extracted for multiple extraction domains without modifying the feature recognition algorithms.

In addition to the above advantages, using the current approach, manufacturability rules can be directly incorporated into a feature definition. However, a limitation of the current approach is that the geometric abstractions presented may not be sufficient for all CAD applications. Depending on the end application, additional geometric abstractions may be required for performing an analysis. For example, an assembly application analysis may require tolerances and mating information, which are not present in the three abstractions presented in this research.

In conclusion, the main contributions of the current approach are summarized as follows:

a) It provides a flexible and interactive means to define features

b) It represents topological variations of form consistently and in a compact manner.

c) It allows multiple interpretations of features making it applicable to multiple domains.

7.2 Implementation

The current approach has been implemented on the ProEngineer® CAD system. The code is implemented in the form of a layered architecture, as shown in Figure 7-1. The top-most layer is the application layer, which corresponds to feature definition and feature recognition. The features application uses the Wrapper layer as an interface to the CAD system. The Wrapper layer contains functions that perform low level queries on the CAD model, using the functions provided by the CAD system. The functions in the Wrapper layer include the functions for traversal of topology and querying geometric properties. The CAD System layer contains the functions provided by the CAD system, which allow topological and geometric queries on the CAD model.

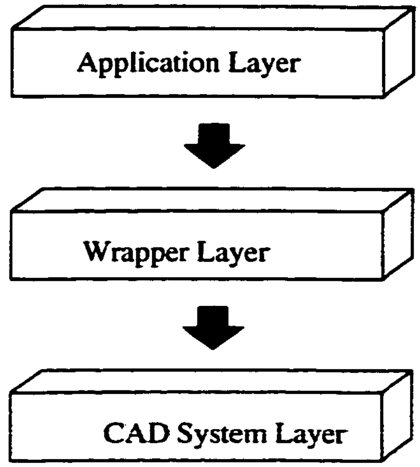


Figure 7-1: Layered Architecture

The purpose of the Wrapper is to isolate the CAD system layer from the application. The following are the benefits of the layered design:

1. If the application needs to be ported to another CAD system, only the Wrapper for that CAD system must be written
2. The same Wrapper can be used for different application programs.
3. Writing an application requires no knowledge of the data structures stored in the CAD system.

7.3 Limitations and Future Research Directions

Feature Interactions: When there is an interaction between two features in a part the topology and geometry of the two features changes. As a result, it is possible that some of the features are not recognized using the current approach. For example, for the part in Figure 7-2 there is an interaction between the rib and slot features. The rib feature can be recognized, however, the slot feature cannot be recognized if it is defined based on the slot as shown on the left. However, if the rib feature is removed after the recognition then the slot feature can be recognized.

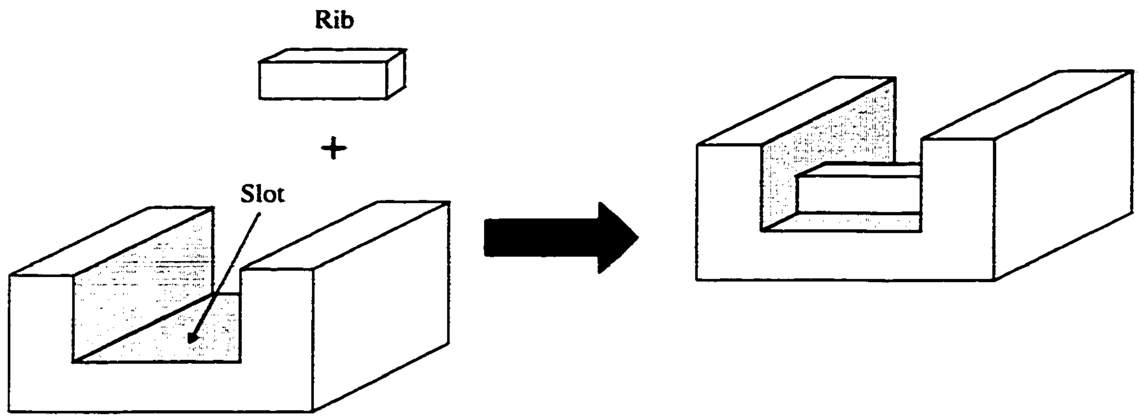


Figure 7-2: Interaction between Rib and Slot features

The modeling system (ProEngineer®) on which the current implementation has been done does not support feature removal. Therefore, if feature removal has to be performed, a modeling system like Acis® or Parasolid® that supports feature removal should be used. After a feature is removed from a model the topology and geometry of the part change locally. The geometric abstractions should be computed again for the entire model. However, this is a time consuming procedure and, instead, the geometric abstractions should also be modified locally. Subsequently, feature recognition can be performed on the new model to recognize the newly created features after feature removal.

<u>Incremental Feature Recognition</u>: One scenario where feature recognition should be performed incrementally is the above example. There are other situations also where this should be done. Consider a case where a user is performing DFM Analysis on a part using a rib feature on the part. Assume that after feature recognition and a subsequent design evaluation it is determined that the rib thickness is too large. The user has to modify the rib

thickness and subsequently perform feature recognition for re-analysis. When the rib feature is modified it is possible that the surrounding topology and geometry is altered. In which case, the geometric abstractions should be re-computed for the new model. Modifying the geometric abstractions incrementally, instead of computing them anew, allows for faster re-analysis. Therefore, it is necessary to develop a methodology for incremental geometric abstraction evaluation.

The following are some suggestions for extending the implementation of the current system.

Inter-feature Attribute and Constraint Specification: Currently, attributes that exist between two separate features cannot be specified during feature definition. For example, the distances l_1 and l_2 in Figure 7-3 cannot be specified in the hole and rib feature definitions. As a result, there is no provision in the current feature recognition algorithms to evaluate the distance between two features. The feature definition language should be improved so that inter-feature attributes and constraints can be specified.

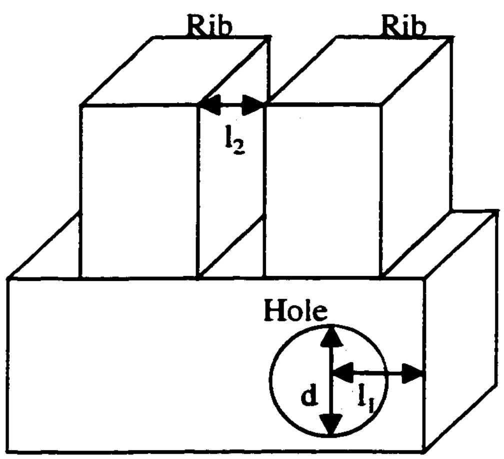


Figure 7-3: Inter-feature attributes

Intra-feature Attribute and Constraint Specification: Even if both the ribs in Figure 7-3 are defined as a single feature, the distance l_2 cannot be specified as an attribute in the current research. This is due to the fact that there is no facility to define a distance attribute between two nodes of a definition graph. Similarly, a constraint such as $l_1/d <$ THRESHOLD cannot be specified in the hole feature definition since there is no means to specify a ratio. Therefore, there is a need for the definition language to be expanded even for intra-feature attribute and constraint specification. The following are the options that should be added to the definition language:

a) Distance between two entities where an entity is a vertex, edge or face as an attribute of a node.

b) Arithmetic expressions in constraints.

Integration with Knowledge-base Expert System: Another improvement that can be done for faster DFM Analysis is the integration of the feature recognition system with a knowledge-base expert system. Information from the recognized features can be automatically used by the expert system to evaluate the design. There already exist commercial expert systems such as NExpert Object from Neuron Data that can be used for the integration.

References

1. Aho, A. V., Hopcroft, J. E., and Ullman, D., **"Data Structures and Algorithms"**, Addison-Wesley Publishing Company, 1987.

2. Ansaldi, S., DeFloriani, L., and Falcidienno, B., **"Geometric Modeling of Solid Objects by using a face adjacency graph representation"**, *Computer Graphics*, Vol. 19, No. 3, 1985, pp. 131-139.

3. Ballard, D. H., and Brown, M., **"Computer Vision"**, Prentice Hall Inc., 1982.

4. Bronsvoort, W. F., and Jansen, F. W., **"Feature Modelling and conversion – Key concepts to concurrent engineering"**, *Computers in Industry.*, Vol. 21, No. 1, pp. 61-86, 1993.

5. Choi, B. K., Barash, M. M., and Anderson, D. C., **"Automatic recognition of machined surfaces from a 3D solid model"**, *Computer Aided Design,* 16(2), pp. 81-86, 1984.

6. Chuang, S. H., and Henderson, M. R., **"Three-Dimensional Shape Pattern Recognition using Vertex Classification and Vertex-Edge Graph"**, *Computer Aided Design,* Vol. 22, July/August, 1990, pp. 377-387.

7. Corney, J., and Clark, D. E. R., **"Method for finding holes and pockets that connect multiple faces in 2.5D objects"**, *Computer Aided Design*, Vol. 23, No. 10, 1991, pp. 658-668.

8. Corney, J., and Clark, D. E. R., **"Efficient face-based feature recognition"**, *Proceedings of Second Symposium on Solid Modeling and Applications*, Sponsored by ACM SIGGRAPH, Montreal, Canada, pp. 313-322, May, 1993.

9. Cunningham, J. J., and Dixon, J. R., **"Designing with features: the origin of features"**, *Proceedings of the 1988 ASME International Computers in Engineering Conference and Exhibition*, Vol. 1, pp. 237-243, 1988.

10. Cutkosky, M. R., Tenenbaum, J. M., and Muller, D., **"Features in process-based design"**, In proceedings of *1988 ASME International Computers in Engineering Conference*, San Francisco, July 31 - August 4, 1988, pp. 557-562.

11. De Floriani, L., and Bruzzone, E., **"Building a feature-based object description from a boundary model"**, *Computer Aided Design*, 21(10), 1989, pp. 602-610.

12. De Martino, T., Falcidieno, B., and Giannini, F., **"Graph-based extraction of manufacturing features from Boundary Models"**, In *Proceedings of 3rd International Conference on Computer Integrated Manufacturing*, New York, May 20-22, 1992, pp. 449-455.

13. Dixon, J. R., **"Designing with features for component design"**, *Workshop on features in design and manufacturing, National Science Foundation*, Feb 1988.

14. Dong, X., and Wozny, M., **"A method for generating volumetric features from surface features"**, *Proceedings of First Symposium on Solid Modeling and Applications*, Sponsored by ACM SIGGRAPH, pp. 185-194, 1991.

15. Elber, G., and Cohen, E., **"Hidden curve removal for free-form surfaces"**, *Computer Graphics,* Vol. 24, No. 4, August, 1990.

16. Elber, G., **"Free form surface analysis using a hybrid of symbolic and numeric computation"**, *Ph.D. Thesis,* Department of Computer Science, The University of Utah, 1992.

17. Ferreira, J. C. E., and Hinduja, S., **"Convex hull-based feature recognition method for 2.5D components"**, *Computer Aided Design,* Vol. 21, No. 10, December, 1990, pp. 41-49.

18. Finger, S., and Safier, S. A., **"Representing and Recognizing Features in Mechanical Designs"**, In *Second International Conference on Design Theory and Methodology, DTM'90,* Chicago, 16-19 September, 1990.

19. Falcidieno, B., and Giannini, F., **"Extraction and organization of form features into a Structure Boundary Model"**, In *Proceedings of Eurographics'87,* North Holland, Netherlands, 1987, pp. 349-359.

20. Falcidieno, B., and Giannini, F., **"Automatic Recognition and Representation of Shape-based Features in a Geometric Modeling System"**, *Computer Vision, Graphics, and Image Processing,* Academic Press, Inc., **48**, 1989, pp. 93-123.

21. Farin, G. E., **"Curves and Surfaces for Computer Aided Geometric Design: A practical guide"**, *Computer Science and Scientific Computing,* 3rd Edition, Academic Press Inc., 1993.

22. Fu, Z., de Pennington, A., and Saia, A., **"Graph grammar approach to feature representation and transformation"**, *International Journal of Computer Integrated Manufacturing,* Vol. 6, Nos. 1 and 2, pp. 137-151, 1993.

23. Gadh, R., **"Abstraction of Manufacturing Features from Design"**, *Ph.D. Thesis,* Mechanical Engineering Department, Carnegie Mellon University, 1991.

24. Gadh, R., and Prinz, F. B., **"Recognition of Geometric Forms using the Differential Depth Filter"**, *Computer Aided Design*, Butterworth Heinemann Publishers, Vol. 24, No. 11, 1992, pp. 583-598.

25. Gadh, R., **"A Hybrid Approach to intelligent geometric design using features-based design and feature recognition"**, In *Proceedings of the 19th Design Automation Conference,* Albuquerque, New Mexico, September 19-22, 1993.

26. Gadh, R., and Prinz, F. B., **"A Computationally Efficient Approach to Feature Abstraction in Design-Manufacturing Integration"**, *Journal of Mechanical Design,* Transactions of the ASME, Vol. 117, February, 1995, pp. 16-27.

27. Gadh, R., and Prinz, F. B., **"Automatic Determination of Feature Interactions in Design-for-Manufacturing Analysis"**, *Journal of Mechanical Design,* Transactions of the ASME, Vol. 117, March, 1995, pp. 2-9.

28. Gadh, R., and Sonthi, R., **"Geometric Shape Abstractions for Internet-based Virtual Prototyping"**, *CAD Journal,* Vol. 30, No. 6, pp. 473-486, 1998.

29. Ganter, M. A., and Tuss, L. L., **"Computer-Assisted Parting Line Development for Cast Pattern Production"**, *Trans. of American Foundrymen's Society*, 1990.

30. Giunchiglia, F. and Walsh, T., **"A Theory of Abstraction"**, *Artificial Intelligence*, Vol. 57, No. 2-3, pp. 323-390, 1992.

31. Gupta, S. K., Regli, W. C., and Nau, D. S., **"Integrating DFM with CAD through Design Critiquing"**, *Concurrent Engineering - Research and Applications*, (2), 1994, pp. 85-95.

32. Gupta, S. K., and Nau, D. S., **"Systematic approach to analyzing the manufacturability of machined parts"**, *Computer Aided Design*, Vol. 27, No. 5, 1995, pp. 323-342.

33. Henderson, M. R., **"Extraction of feature information from three-dimensional CAD data"**, Ph.D. Thesis, 1984.

34. Henderson, M. R., **"Extraction and Organization of Form Features"**, *Software for Discrete Manufacturing*, Editors Crestin - McWaters, IFIP, Elsevier Science B.V, pp. 547-557, 1986.

35. Henderson, M. R., Srinath, G., Stage, R., Walker, K., Regli, W., **"Boundary Representation-based Feature Identification"**, *Advances in Feature Based Manufacturing*, Shah, J. J., Mäntylä, M., and Nau, D., (Editors), Manufacturing Research and Technology, Vol. 20, Elsevier Science B. V., 1994, pp. 15-38.

36. Hoover, P. S., and Rinderle, R. J., **"Abstractions, Design Views and Focusing"** *Design Theory and Methodology - ASME DE,* Vol. 68, pp. 115-129, 1994.

37. Hoschek, J., and Lasser, D., **"Fundamentals of Computer Aided Geometric Design"**, Translated by Schumaker, L., A K Peters Ltd., Wellesley, Massachusetts, 1993.

38. Huh, Y.-J., and Kim, S.-G., **"RIBBER: A knowledge-based synthesis system for ribbed injection molded parts"**, Proceedings of *Symposium on Concurrent Product and Process Design,* ASME Winter Annual Meeting, San Francisco, CA, Dec 1-15, 1989, pp. 195-204.

39. Huh, Y.-J., and Kim, S.-K., **"A knowledge-based CAD system for concurrent product design in injection molding"**, *International Journal of Computer Integrated Manufacturing,* Vol. 4, No. 4, 1991, pp. 209-218.

40. Jakubowski, R., **"Extraction of Shape Features for Syntactic Recognition of Mechanical Parts"**, *IEEE Transactions on Systems, Man and Cybernetics,* Vol. SMC-15, No. 5, September-October, 1985, pp. 642-651.

41. Joshi, S., and Chang, T. C., **"Graph-based heuristics for recognition of machining features from a 3D solid model"**, *Computer Aided Design,* Vol. 20, March, 1988.

42. Kim, Y.-S., **"Recognition of form features using Convex Decomposition"**, *Computer Aided Design,* Vol. 24, September, 1992, pp. 461-476.

43. Koenderink, J. J., **"Solid Shape"**, 3rd Edition, MIT Press, Cambridge, Massachusetts, 1990.

44. Krause, F.-L., Ulbrich, A., and Vosgerau, F., H., **"Feature-based approach for the integration of design and process planning systems"**, presented at *IFIP 5.2 Workshop on Geometric Modeling,* Rensellaerville, NY, June 17-21, 1990.

45. Kunjur, G., **"The CR Approach for Feature Recognition to Support Virtual Prototyping"**, MS Thesis, Department of Manufacturing Systems Engineering, University of Wisconsin - Madison, 1996.

46. Kyprianou, L. K., **"Shape Classification in Computer-Aided Design"**, *Ph.D. Thesis,* Christ's College, University of Cambridge, 1980.

47. Laakko, T., **"Incremental Feature Modeling: Methodology for Integrating Features and Solid Models"**, *Dr. Tech Thesis,* Helsinki University of Technology, Laboratory of Information Processing Science, 1993.

48. Laakko, T., and Mäntylä, M., **"A feature definition language for bridging solids and features"**, *Proceedings of Second Symposium on Solid Modeling and Applications,* Sponsored by ACM SIGGRAPH, Montreal, Canada, pp. 333-342, May, 1993.

49. Laakko, T., and Mäntylä, M., **"Incremental Feature Modeling"**, *Advances in Feature Based Manufacturing,* Shah, J. J., Mäntylä, M., and Nau, D., (Editors), Manufacturing Research and Technology, Vol. 20, Elsevier Science B. V., 1994, pp. 455-479.

50. Lentz, D. H., and Sowerby, R., **"Feature extraction of concave and convex regions and their intersections"**, *Computer Aided Design,* Vol. 25, No. 7, July, 1993, pp. 421-437.

51. Liu, S. S., **"The Definition and Extraction of Shape Abstractions for Automatic Finite Element Hexahedral Mesh Generation"**, *Ph.D. Thesis*, Department of Mechanical Engineering, University of Wisconsin-Madison, 1997.

52. Maekawa, T., and Patrikalakis, N. M., **"Interrogation of differential geometry properties for design and manufacture"**, *The Visual Computer*, Vol. 10, 1994, pp. 216-237.

53. Maekawa, T., Wolter, F.-W., and Patrikalakis, N. M., **"Umbilics and lines of Curvature for shape interrogation"**, *Computer Aided Geometric Design*, Vol. 13, 1996, pp. 133-161.

54. Mäntylä, M., and Opas, J., and Puhakka, J., **"A prototype system for generative process planning of prismatic parts"**, in Kusiak, A., ed., *Modern Production Management Systems - Proceedings of IFIP TC 5/WG 5.7 Working Conference on Advances in Production Management Systems (APMS '87)*, Winnipeg, Manitoba, Canada, 1987, pp. 599-611.

55. Mäntylä, M., **"An Introduction to Solid Modeling"**, Computer Science Press, Maryland, 1988.

56. Marefat, M., and Kashyap, R. L., **"Geometric Reasoning for Recognition of Three Dimensional Object Features"**, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, No. 10, 1990, pp. 949-965.

57. Margetis, G D., **"Robust Interrogation of Differential Properties for Design and Manufacture"**, *MS Thesis*, Naval Architecture, Marine Engineering and Mechanical Engineering, Massachusetts Institute of Technology, September, 1994.

58. Mäntylä, M., Opas, J., and Puhakka, J., **"A prototype system for generative process planning of prismatic parts"**, *Modern Production Management Systems, APMS'87*, Ed. Kusiak, A., Amsterdam, 1987, pp. 599-611.

59. Peters, T. J., **"Combinatorial Analysis of Feature Recognition"**, In *Proceedings of 4th International Conference on Design Theory and Methodology*, Scottsdale, AZ, September, 1992.

60. Ranta, M., Inui, M., Kimura, F., and Mäntylä, M., **"Cut and paste based modeling with boundary features"**, in Rossignac, J., Turner, J., and Allen, G., eds., *Proceedings of Second ACM Symposium on Solid Modeling and Applications*, Montreal, May 19-21, 1993, pp. 303-312.

61. Regli, W. C., and Nau, D. S., **"Building a general approach to feature recognition of material removal shape elements"**, In Proceedings of *2nd Symposium on Solid Modeling and Applications,* Montreal, Canada, May 19-21, 1993, pp. 293-302.

62. Regli, W. C., and Pratt, M. J., **"What are feature interactions?"**, In Proceeding of *1996 ASME Design Engineering Technical Conference and Computes in Engineering,* Irvine, California, August 18-22, 1996.

63. Requicha, A. A. G., and Vandenbrande, J. H., **"Form features for mechanical design and manufacturing"**, In proceedings of *1988 ASME International Computers in Engineering Conference,* Anaheim, 1989, pp. 47-52.

64. Rogers, M., **"Form feature modeling shell for design of mechanical parts"**, *M. S. Thesis,* Department of Mechanical Engineering, Arizona State University, 1987.

65. Roller, D., **"Design by features: An approach to high-level shape manipulations"**, *Computers in Industry,* Vol. 12, 1989, pp. 185-191.

66. Rosen, D. W., Dixon, J. R., and Finger, S., **"Conversions of Feature-Based Design Representations using Graph Grammar Parsing"**, *ASME Journal of Mechanical Design,* 116(3), 1994, pp. 785-792.

67. Sakurai, H., and Gossard, D. C., **"Recognizing Shape features in Solid Models"**, *IEEE Computer Graphics and Applications,* September, 1990, pp. 22-32.

68. Salomons, O. W, **"Computer support in the design of mechanical parts - Constraint specification and satisfaction in mechanical feature-based design and manufacturing"**, *Ph.D. Thesis,* University of Twente, 1995.

69. Shah, J. J., **"Conceptual Development of Form Features and Feature Modelers"**, *Research in Engineering Design,* 1991, 2:93-108.

70. Shah, J. J., **"Assessment of features technology"**, *Computer Aided Design,* Vol. 23, No. 5, 1991, pp. 331-343.

71. Shah, J. J., Mäntylä, M., and Nau, D., **"Introduction to Feature Based Manufacturing"**, *Advances in Feature Based Manufacturing*, Shah, J. J., Mäntylä, M., and Nau, D., (Editors), Manufacturing Research and Technology, Vol. 20, Elsevier Science B. V., 1994, pp. 1-11.

72. Shah, J. J., and Rogers, M. T., **"Expert form feature modeling shell"**, *Computer Aided Design*, Shah, Vol. 20, No. 9, November, 1988, pp. 515-524.

73. Shah, J. J., and Rogers, M. T., **"A Testbed for Rapid Prototyping of Feature-based Applications"**, *Advances in Feature Based Manufacturing*, Shah, J. J., Mäntylä, M., and Nau, D., (Editors), Manufacturing Research and Technology, Vol. 20, Elsevier Science B. V., 1994, pp. 423-453.

74. Shah, J. J., and Mäntylä, M., **"Parametric and Feature-based CAD/CAM: Concepts, Techniques and Applications"**, John-Wiley and Sons, Inc., 1995.

75. Sonthi, R., Harinarayan, K. R., and Gadh, R., **"Feature Extraction from Non-linear Geometric Models in Design For Manufacturing"**, *SAE International Off-Highway & Powerplant Congress & Exposition*, MECCA, Milwaukee, WI, September 12 - 14, 1994.

76. Sonthi, R., Dani, T. H., and Gadh, R., **"Concurrent Design of Sheet Metal Parts"**, *International Body Engineering Conference*, Detroit, MI, Sept 26-29, 1994.

77. Sonthi, R., Kunjur, G., and Gadh, R, **"Determination of Shape Abstractions from Geometric Models via the Curvature Region Representation (CR-REP)"**, *IFIP WG5.2 Workshop on Geometric Modeling in CAD*, Warrenton, VA, May 1996.

78. Sonthi, R., and Gadh, R., **"A Curvature Region Approach to Shape Feature Determination"**, *1997 NSF Design and Manufacturing Grantees Conference*, Seattle WA, January 7-10, 1997.

79. Sonthi, R., Kunjur, G., and Gadh, R., **"Feature Recognition using Curvature Regions for Design-For-Manufacturability Analysis"**, *North American Manufacturing Research Conference XXV*, Nebraska, Lincoln, May 20-23, 1997.

80. Sonthi, R., Kunjur, G., and Gadh, R., "**Role of Abstractions to Support Virtual Prototyping: Parting Line for Die Design**", *Manufacturing Systems*, Vol. 26(1997), No.4, pp. 5-11, 1997 (also at *28th CIRP International Seminar on Manufacturing Systems*, Johannesburg, South Africa, May 15-17, 1996.)

81. Sonthi, R., Kunjur, G., and Gadh, R., **"Shape Feature Determination using the Curvature Region Representation"**, *Proceedings of Fourth Symposium on Solid Modeling and Applications*, Sponsored by ACM SIGGRAPH, Atlanta, Georgia, May 14-16, 1997, pp. 285-296.

82. Sonthi, R., and Gadh, R., **"MMCs and PPCs as constructs of curvature regions for form feature determination"**, *CAD Journal*, Vol.30, No.13, pp. 997-1001, 1998 (also at *1997 ASME Computers In Engineering Conference*, Sacramento, CA, September 14-17, 1997).

83. Sonthi, R., Zhao, F., and Gadh, R., **"Feature Definition and Extraction using Curvature Region/Primitive Shape Abstractions along with Constraint**

Specification", *SIAM Workshop on Mathematical Foundations for Features in Computer-Aided Design, Engineering and Manufacturing*, Troy, MI, Oct 22-23, 1998.

84. Sreevalsan, P., **"An investigation into the unification of form feature definition methods"**, *M. S. Thesis*, Mechanical Engineering, Arizona State University, 1990.

85. Staley, S. M., Henderson, M. R., and Anderson, D. C., **"Using Syntactic Pattern Recognition to Extract Feature Information from a Solid Geometric Database"**, *Computers in Mechanical Engineering*, September, 1983.

86. Taylor, E. L., and Henderson, R. M., **"Role of features and abstraction in Mechanical Design"**, *Design Theory and Methodology - ASME DE*, Vol. 68, pp. 131-140, 1994.

87. Turner, G. P., and Anderson, D. C., **"An object oriented approach to interactive, feature based design for quick turnaround manufacturing"**, In proceedings of *1988 ASME International Computers in Engineering Conference*, San Francisco, July 31 - August 4, 1988, pp. 551-555.

88. Vandenbrande, J. H., and Requicha, A. A. G., **"Geometric computation for the recognition of spatially interacting machining features"**, *Advances in Feature Based Manufacturing*, Shah, J. J., Mäntylä, M., and Nau, D., (Editors), Manufacturing Research and Technology, Vol. 20, Elsevier Science B. V., 1994, pp. 83-106.

89. Van Emmerik, M. J. G. M., and Jansen, F. W., **"User interface for feature modeling"**, *Proceedings of CAPE*, pp. 625-632, 1989.

90. Van Emmerik, M. J. G. M., **"Interactive design of parameterized 3D models by direct manipulation"**, *Ph.D. Thesis,* Delft University of Technology, Netherlands, 1990.

91. Waco, D. L., and Kim, Y.-S., **"Geometric reasoning for machining features using convex decomposition"**, *Computer Aided Design,* Vol. 26, June, 1994, pp. 477-489.

92. Wang, M. T., **"A geometric reasoning methodology for manufacturing feature extraction from a 3D CAD model"**, *Ph.D. Dissertation,* Purdue University, 1990.

93. Wilson, P. R., and Pratt, M. J., **"A taxonomy of features in Solid Modeling"**, *Geometric Modeling for CAD Applications,* Wozny, M. J., McLaughlin, H. W., and Encarnacao, J. L., (Editors), Elsevier Science Publishers, 1988, pp. 125-137.

94. Woo, T. C., **"Feature extraction by volume decomposition"**, In *Proceedings of Conference on CAD/CAM in Mechanical Engineering,* MIT, Cambridge, Massachusetts, March 24-26, 1982, pp. 39-45.

95. Zeid, I., **"CAD/CAM Theory and Practice"**, 1st Edition, McGraw-Hill Inc., 1991.

96. Zhao, F., Sonthi, R., and Gadh, R., **"Feature Extraction from Filleted Geometric Models via a Virtual Edge Approach"**, *SIAM Workshop on Mathematical Foundations for Features in Computer-Aided Design, Engineering and Manufacturing,* Troy, MI, Oct 22-23, 1998.

97. Zhao, F., Sonthi, R., and Gadh, R., **"Virtual Edge/Vertex Creation from File in Geometric Models for Feature Extraction"**, *International Journal for Vehicle Design including CAD/ CAM Applications,* Vol. 21, No. 2/3, Special Issues, pp. 205-227, 1999.

8 Appendix A: Equations for Curvatures on a Surface

The general parametric equation of a surface is:

$$\bar{P}(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix} \qquad \text{Eq. 1}$$

where,

$u_{min} \leq u \leq u_{max}$

$v_{min} \leq v \leq v_{max}$

The normal curvature at a point $\bar{P}(u,v)$ on a surface is the curvature at $\bar{P}(u,v)$ on the normal section curve, which is a curve of intersection between a plane containing the normal $\bar{n}$ at point $\bar{P}(u,v)$ and the surface. There can exist a family of planes that contain $\bar{n}$ and, therefore, a family of normal section curves. The curvature at a point on a normal section curve that is represented in the form {u = u(t), v= v(t)} is:

$$\kappa = \frac{\left(Lu'^2 + 2Mu'v' + Nv'^2\right)}{\left(Eu'^2 + 2Fu'v' + Gv'^2\right)} \qquad \text{Eq. 2}$$

where,

$u' = \delta u/\delta t$

$v' = \delta v/\delta t$

$L(u,v) = \bar{n} \cdot \bar{P}_{uu}$

$M(u,v) = \bar{n} \cdot \bar{P}_{uv}$

$N(u,v) = \bar{n} \cdot \bar{P}_{vv}$

$E(u,v) = \bar{P}_u \cdot \bar{P}_u$

$F(u,v) = \bar{P}_u \cdot \bar{P}_v$

$G(u,v) = \bar{P}_v \cdot \bar{P}_v$

The radius of curvature at the point is $\rho = 1/\kappa$. Equation 2 gives the surface curvature in any direction at point $\bar{P}(u,v)$.

The Gaussian Curvature *K* and the mean curvature *H* are defined by:

$$K = \frac{(LN - M^2)}{(EG - F^2)} \qquad \text{Eq. 3}$$

$$H = \frac{(EN + GL - 2FM)}{2(EG - F^2)} \qquad \text{Eq. 4}$$

The principal curvatures at a point which are the maximum (κ_{max}) and minimum (κ_{min}) normal curvatures at the point, can be defined in terms of K and H.

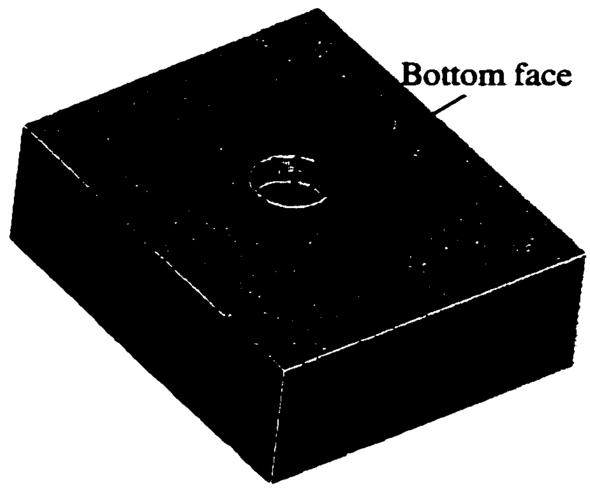
$$\kappa_{max} = H + \sqrt{H^2 - K} \qquad \text{Eq. 5}$$

$$\kappa_{min} = H - \sqrt{H^2 - K} \qquad \text{Eq. 6}$$

9 Appendix B: Example Feature Definitions

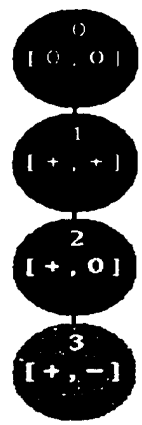
Example Feature 1:

Feature: Blind Hole Definition Type: CRGRP Parameters: Depth(h), Diameter(d) Constraints: None




```
BLINDHOLE CRGRP    ATTRIBUTES{
CRNODES{             4
  4                  bottomface  BOTTOMFACE 0
  2 1 1              sideface    SIDEFACE  2
  8 2 0 2            diameter    EDGEDIAMETER 1
  5 2 1 3            depth       FACELENGTH  2
  6 1 2            }
}                  CONSTRAINTS{
                     0
                   }
```

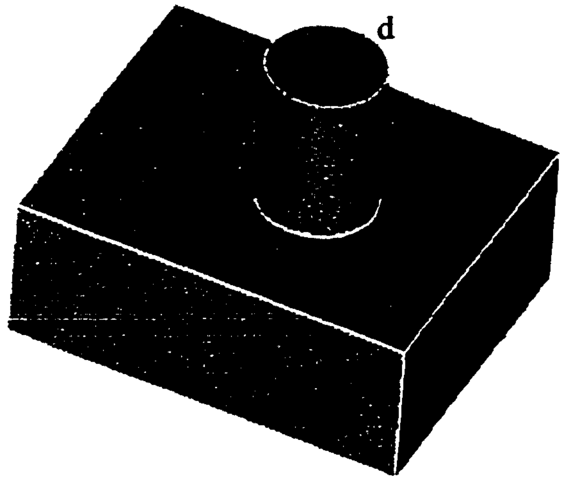
Feature Definition



Feature CR-Graph

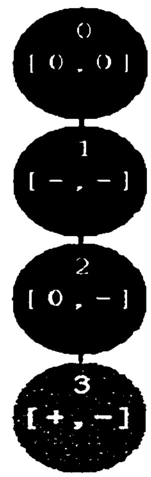
Example Feature 2:

Feature: Boss
Definition Type: CRGRP
Parameters: Height(h),
Diameter(d)
Constraints: d <= h
d > 2.0




```
BOSS CRGRP            ATTRIBUTES{
CRNODES{                  2
    4                     diameter DIAMETER 2
    2 1 1                 height LENGTH 2
    4 2 0 2           }
    3 2 1 3           CONSTRAINTS{
    6 1 2                 2
}                         diameter less-than-equal-to height
                          diameter greater-than 2.0
                      }
```

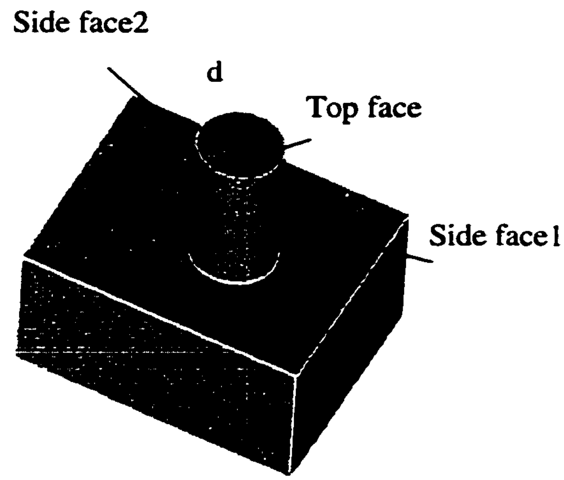
Feature Definition



Feature CR-Graph

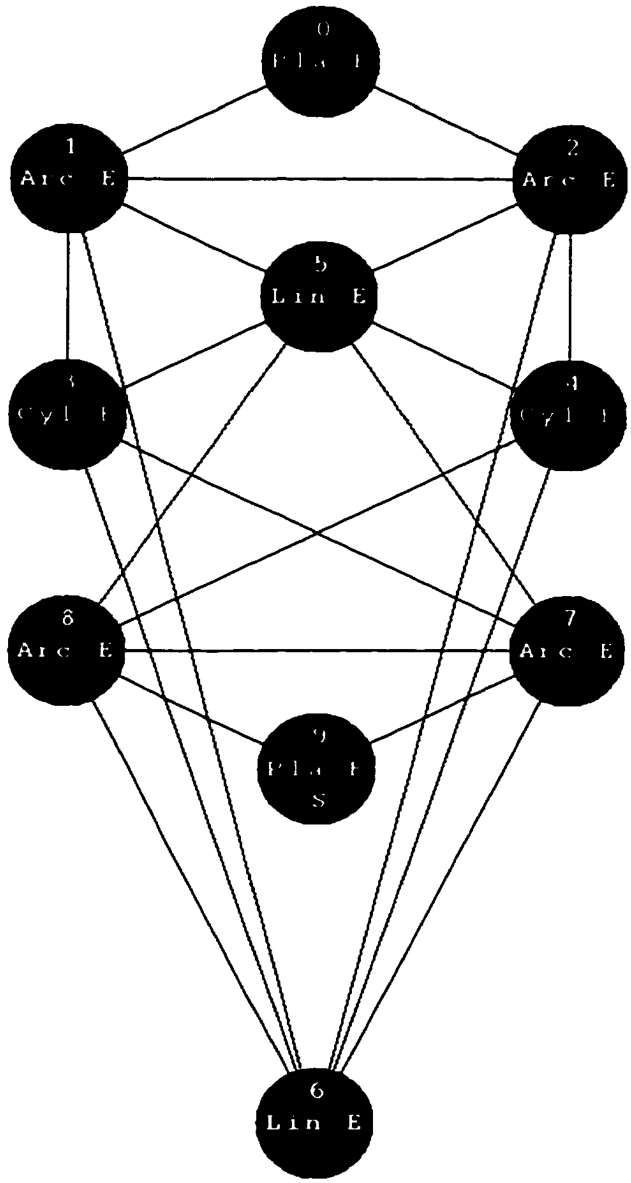
Example Feature 3:

Feature: Boss1
Definition Type: BRGR
Parameters: Height(h),
Diameter(d)
Topface, sideface1,
sideface2
Constraints: d <= h




```
BOSS1 BRGR              ATTRIBUTES{
BRNODES{                  5
  10                      topface   TOPFACE 0
  0 34 2 1 2              sideface1  SIDEFACE 3
  1 3 2 5 0 2 3 5 6       sideface2  SIDEFACE 4
  1 3 2 5 0 1 4 5 6       diameter  EDGEDIAMETER  1
  0 36 4 1 5 6 7          height   FACELENGTH  3
  0 36 4 2 5 6 8        }
  1 2 1 6 1 2 3 4 7 8   CONSTRAINTS{
  1 2 1 6 1 2 3 4 7 8     1
  1 3 4 5 3 5 6 8 9       diameter less-than-equal-to height
  1 3 4 5 4 5 6 7 9     }
  0 34 0
}
```

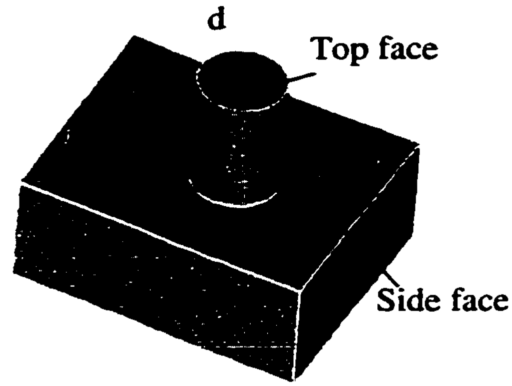
Feature Definition



<u>Feature CR-Graph</u>

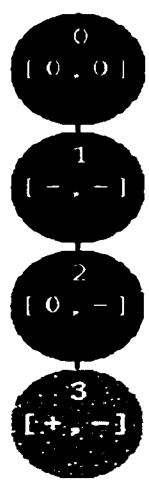
Example Feature 4:

Feature: Button
Definition Type: CRGRP
Parameters: Height(h),
Diameter(d)
sideface
topface
Constraints: d > h




```
BUTTON CRGRP          ATTRIBUTES{
CRNODES{                 4
     4                   topface    TOPFACE  0
     2 1 1               sideface   SIDEFACE 2
     4 2 0 2             diameter   FACEDIAMETER  2
     3 2 1 3             height     FACELENGTH  2
     6 1 2            }
}                     CONSTRAINTS{
                         1
                         diameter greater-than height
                      }
```

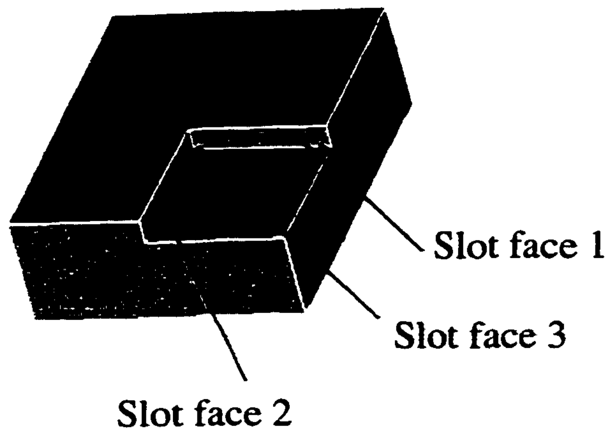
Feature Definition



Feature CR-Graph

Example Feature 5:

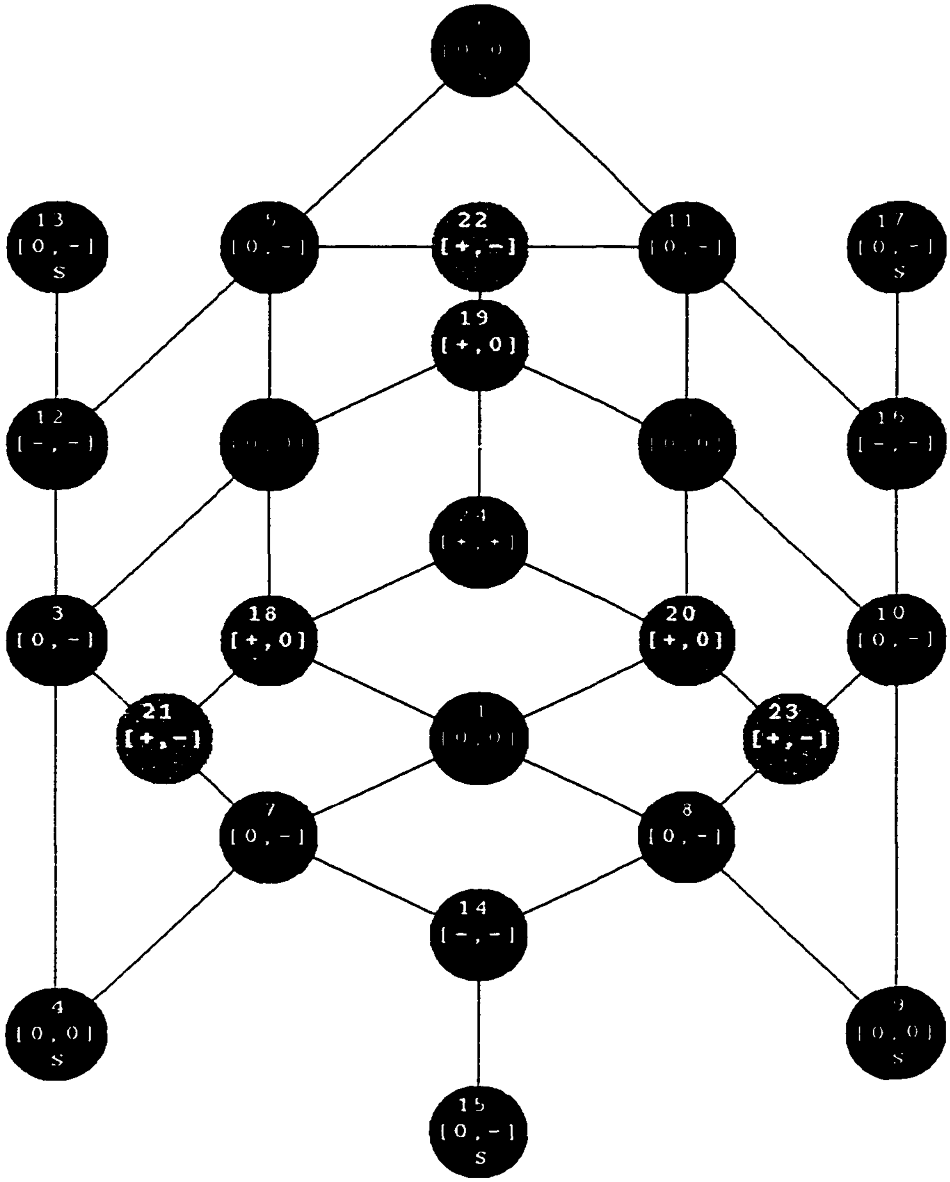
Feature: Corner Slot Definition Type: CRGRP Parameters: Height(h), Width(w) Depth(d) slotface 1, slotface 2, slotface 3 Constraints: None




```
CORNERSLOT CRGRP
CRNODES{
  25
  2 4 3 5 18 19
  2 4 7 8 18 20
  2 4 10 11 19 20
  3 4 4 0 12 21
  2 1 3
  3 4 6 0 12 22
  2 1 5
  3 4 4 1 14 21
  3 4 9 1 14 23
  2 1 8
  3 4 9 2 16 23
  3 4 6 2 16 22
  4 3 3 13 5
  3 1 12
  4 3 7 15 8
  3 1 14
  4 3 10 17 11
  3 1 16
  5 4 1 0 21 24
  5 4 2 0 22 24
  5 4 2 1 23 24
  6 3 7 3 18
  6 3 11 5 19
  6 3 10 8 20
  8 3 19 20 18
}

ATTRIBUTES{
  6
  slotface1  SIDEFACE 0
  slotface2  SIDEFACE 1
  slotface3  SIDEFACE 2
  length  EDGELENGTH  18
  width   EDGELENGTH  19
  depth   EDGELENGTH  20
}
CONSTRAINTS{
  0
}
```

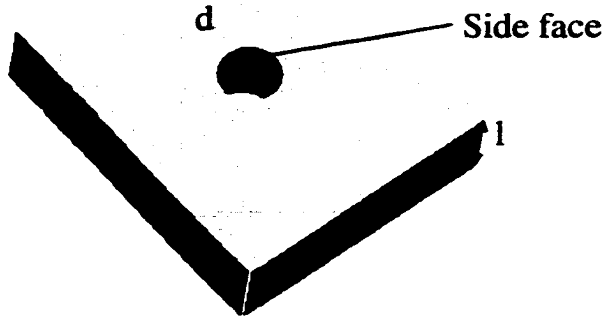
Feature Definition



<u>Feature CR-Graph</u>

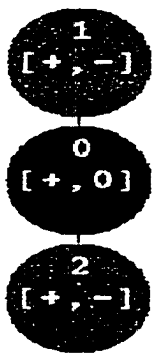
Example Feature 6:

Feature: Hole
Definition Type: CRGRP
Parameters: Diameter(d)
Length(l)
sideface
Constraints: None




```
HOLE CRGRP            ATTRIBUTES{
CRNODES{                 3
   3                     sideface   SIDEFACE   0
   1 5 2 1 2             diameter   FACEDIAMETER  0
   1 6 1 0               length     FACELENGTH  0
   1 6 1 0               }
}                        CONSTRAINTS{
                            0
                         }
```

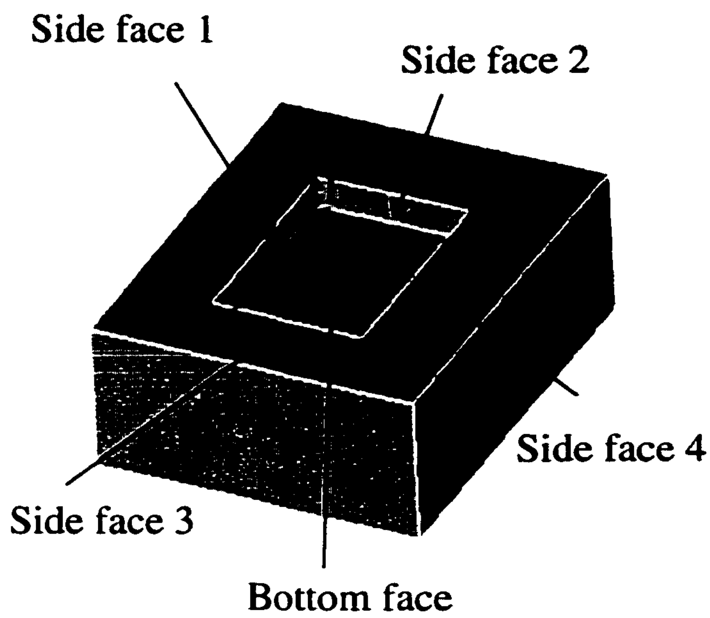
Feature Definition



Feature CR-Graph

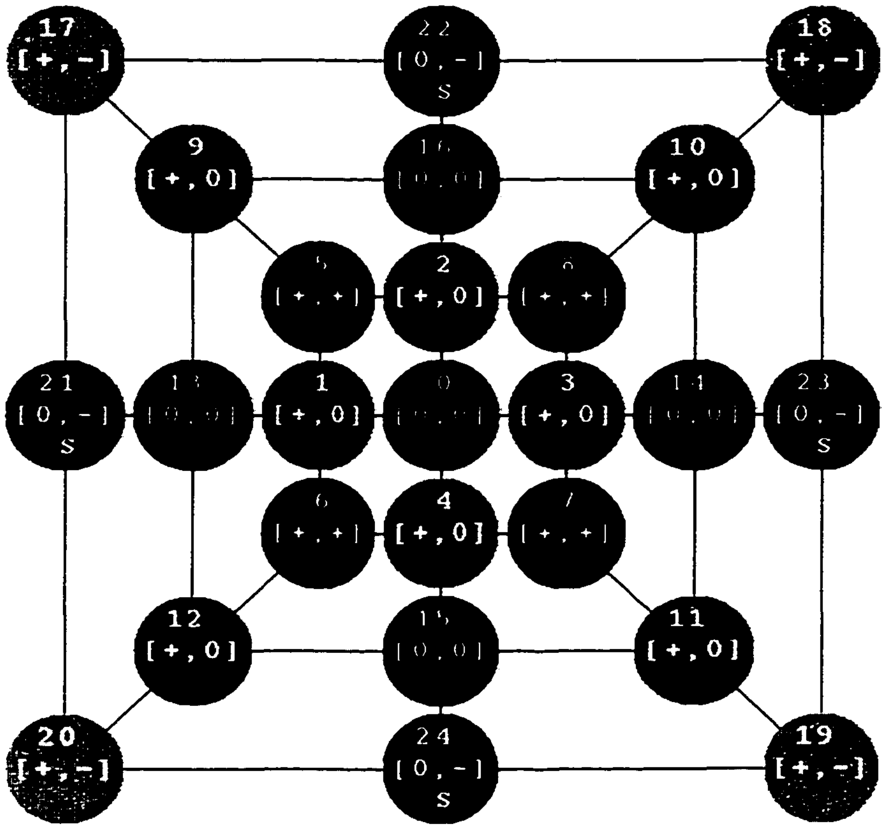
Example Feature 7:

Feature: Pocket
Definition Type: CRGRP
Parameters: Width(d)
Length(l)
Height(h)
bottomface,
sideface1,
sideface2,
sideface3,
sideface4
Constraints: None




```
POCKET CRGRP          ATTRIBUTES{
CRNODES{                8
  25                    bottomface  BOTTOMFACE 0
  2 4 3 2 1 4           sideface1   SIDEFACE 13
  5 4 0 13 6 5          sideface2   SIDEFACE 14
  5 4 0 16 5 8          sideface3   SIDEFACE 15
  5 4 0 14 8 7          sideface4   SIDEFACE 16
  5 4 0 15 6 7          length   LENGTH  1
  8 3 9 1 2             width    LENGTH  2
  8 3 12 4 1            depth    LENGTH  9
  8 3 11 3 4          }
  8 3 10 2 3          CONSTRAINTS{
  5 4 16 13 5 17        0
  5 4 14 16 8 18      }
  5 4 15 14 7 19
  5 4 15 13 6 20
  2 4 1 21 9 12
  2 4 3 23 11 10
  2 4 4 24 12 11
  2 4 2 22 10 9
  6 3 22 21 9
  6 3 23 22 10
  6 3 23 24 11
  6 3 21 24 12
  3 3 13 17 20
  3 3 16 18 17
  3 3 14 19 18
  3 3 15 20 19
}
```

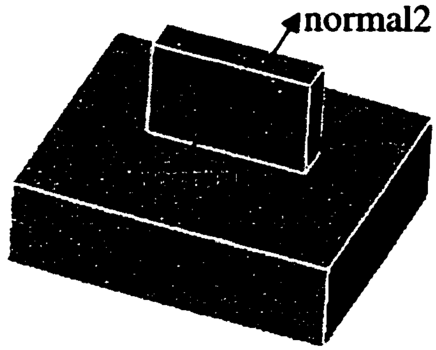
Feature Definition



Feature CR-Graph

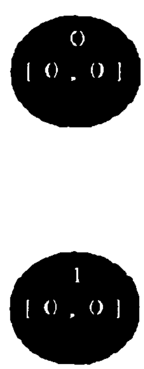
Example Feature 8:

Feature: ProtRib
Definition Type: PROT
Parameters: normal1,
normal2
Constraints:
normal1 anti-parallel-to normal2




```
PROTRIB PROT        ATTRIBUTES{
CRNODES{                 4
     2                   n1 NORMAL 0
     2 0                 isface0 ISFACE 0
     2 0                 n2 NORMAL 1
}                        isface1 ISFACE 1
                    }
                    CONSTRAINTS{
                         3
                         n1 antiparallel-to n2
                         isface0 equal-to 1.0
                         isface1 equal-to 1.0
                    }
```

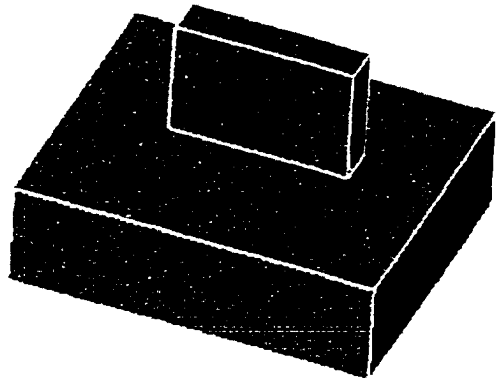
Feature Definition



Feature CR-Graph

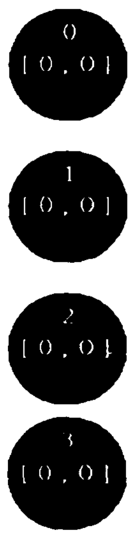
Example Feature 9:

Feature: ProtRib1
Definition Type: PROT
Parameters: None
Constraints: None



```
PROTRIB1 PROT  ATTRIBUTES{
CRNODES{            0
    4           }
     2 0        CONSTRAINTS{
     2 0            0
     2 0        }
     2 0
}
```

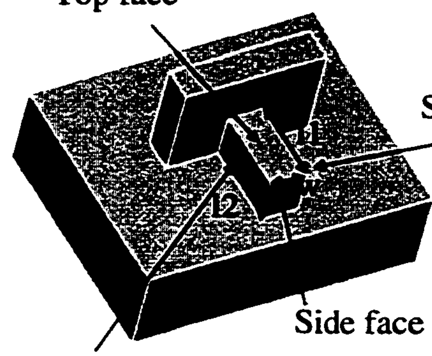
Feature Definition



Feature CR-Graph

Example Feature 10:

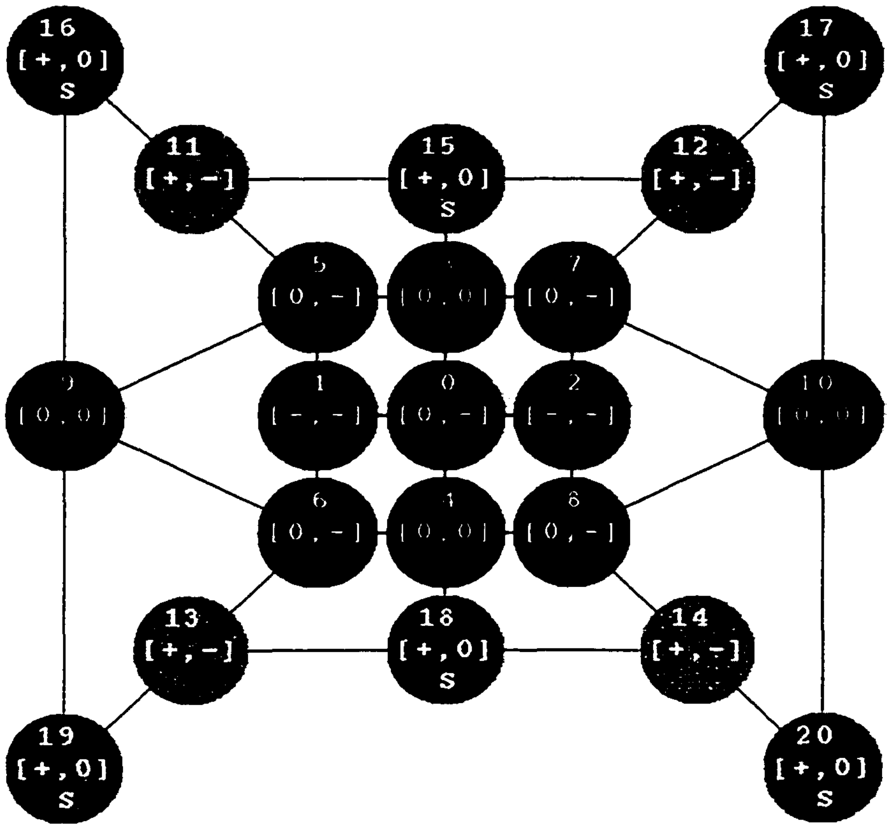
Feature: Rib1
Definition Type: CRGRP
Parameters: Length1(l1),
Length2(l2),
Width(w),
Topface,
Sideface1,
Sideface2,
Sideface3
Constraints: w < l1
w < l2




```
RIB1 CRGRP
CRNODES{
   21
   3 4 1 2 3 4
   4 3 0 5 6
   4 3 0 7 8
   2 4 0 5 7 15
   2 4 0 6 8 18
   3 4 3 1 11 9
   3 4 4 1 13 9
   3 4 3 2 12 10
   3 4 4 2 14 10
   2 4 5 6 16 19
   2 4 7 8 17 20
   6 3 5 15 16
   6 3 7 15 17
   6 3 6 19 18
   6 3 8 18 20
   5 3 11 12 3
   5 2 11 9
   5 2 12 10
   5 3 13 14 4
   5 2 13 9
   5 2 14 10
}

ATTRIBUTES{
   7
   topface1   TOPFACE      3
   topface2   SIDEFACE     4
   sideface1  SIDEFACE     9
   sideface2  SIDEFACE     10
   length1    EDGELENGTH   7
   length2    EDGELENGTH   8
   width      EDGELENGTH   0
}
CONSTRAINTS{
   2
   width less-than length1
   width less-than length2
}
```

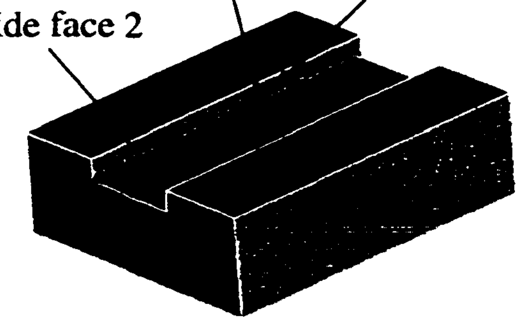
Feature Definition



<u>Feature CR-Graph</u>

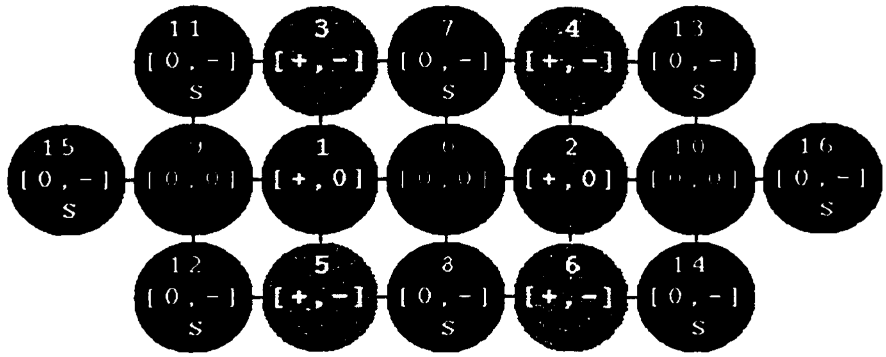
Example Feature 11:

Feature: Slot1
Definition Type: CRGRP
Parameters: Length (l),
Bottomface,
Sideface1,
Sideface2,
Constraints: None




```
SLOT1 CRGRP
CRNODES{
   17
   2 4 2 1 8 7
   5 4 0 3 5 9
   5 4 0 10 6 4
   6 3 1 11 7
   6 3 2 13 7
   6 3 1 12 8
   6 3 2 8 14
   3 3 0 3 4
   3 3 0 5 6
   2 4 1 11 12 15
   2 4 2 14 16 13
   3 2 9 3
   3 2 9 5
   3 2 10 4
   3 2 10 6
   3 1 9
   3 1 10
}
ATTRIBUTES{
   4
   bottomface   BOTTOMFACE 0
   sideface1   SIDEFACE  9
   sideface2   SIDEFACE  10
   length   EDGELENGTH  1
}
CONSTRAINTS{
   0
}
```

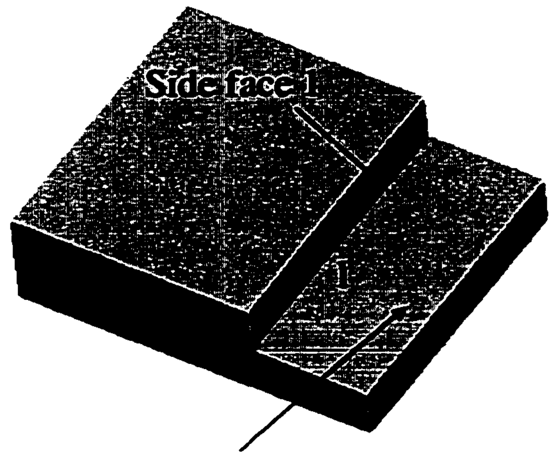
Feature Definition



Feature CR-Graph

Example Feature 12:

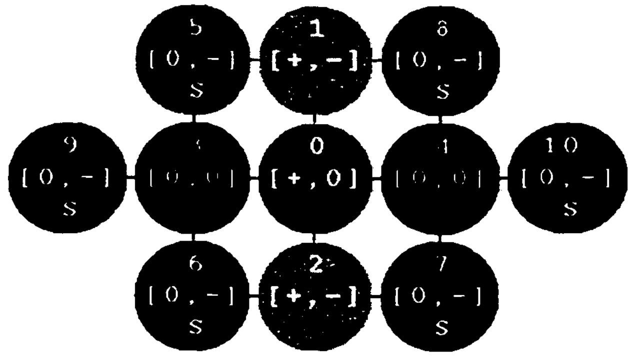
Feature: Slot2
Definition Type: CRGRP
Parameters: Length (l),
Sideface1,
Sideface2,
Constraints: None




```
SLOT2 CRGRP
CRNODES{
  11
  5 4 2 1 3 4
  6 3 0 5 8
  6 3 0 6 7
  2 4 0 5 6 9
  2 4 0 8 7 10
  3 2 1 3
  3 2 2 3
  3 2 2 4
  3 2 1 4
  3 1 3
  3 1 4
}

ATTRIBUTES{
  3
  slotface1   SIDEFACE  3
  slotface2   SIDEFACE  4
  depth       EDGELENGTH  0
}
CONSTRAINTS{
  0
}
```

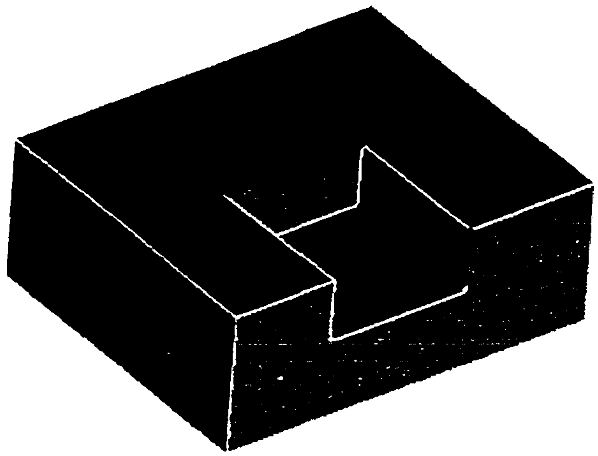
Feature Definition



<u>**Feature CR-Graph**</u>

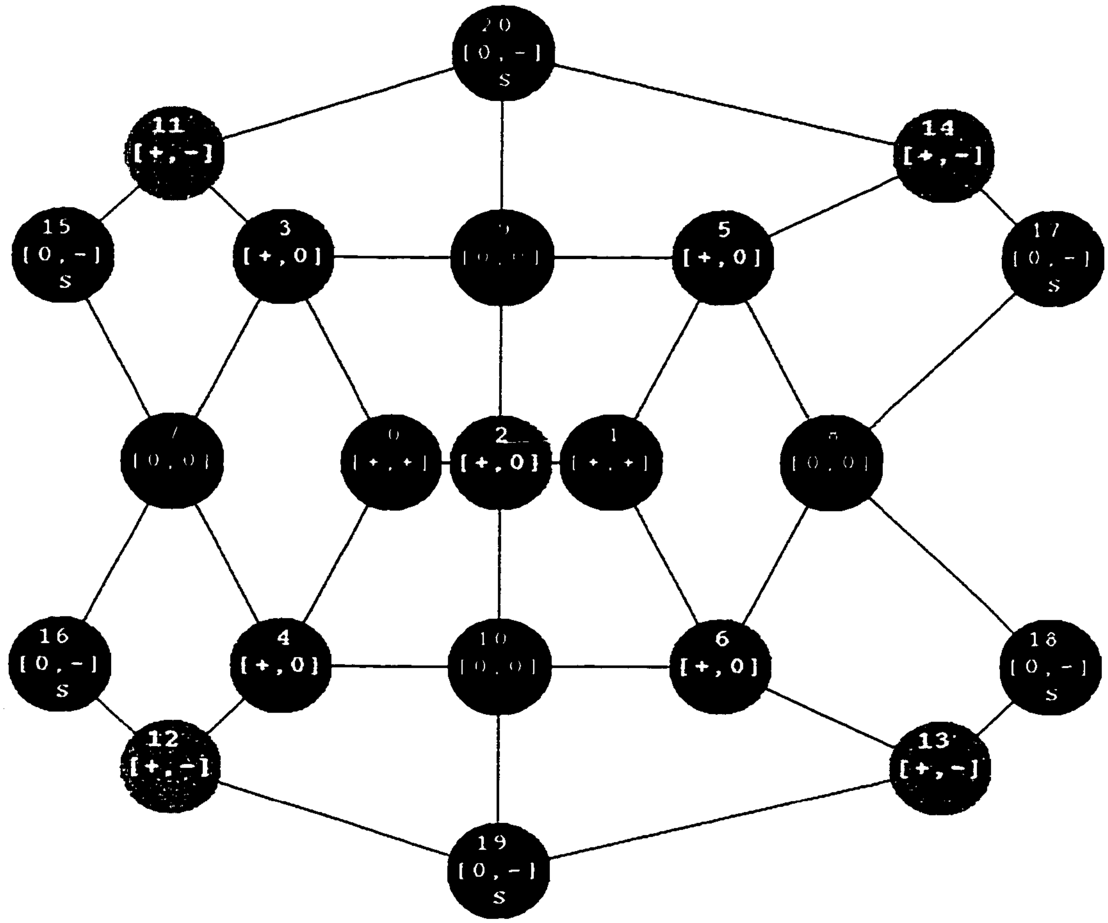
Example Feature 13:

Feature: Slot3
Definition Type: CRGRP
Parameters: Length1 (l1),
Lenght2 (l2),
Width (w),
Constraints: None



```
SLOT3 CRGRP
CRNODES{
   21
   8 3 2 3 4
   8 3 2 6 5
   5 4 0 1 9 10
   5 4 0 9 7 11
   5 4 0 10 7 12
   5 4 1 9 8 14
   5 4 1 10 8 13
   2 4 3 4 15 16
   2 4 5 6 17 18
   2 4 20 2 5 3
   2 4 19 6 4 2
   6 3 3 15 20
   6 3 4 16 19
   6 3 6 18 19
   6 3 5 20 17
   3 2 7 11
   3 2 7 12
   3 2 8 14
   3 2 8 13
   3 3 10 12 13
   3 3 9 11 14
}
ATTRIBUTES{
   3
   width    EDGELENGTH  2
   length1  EDGELENGTH  3
   length2  EDGELENGTH  4
}
CONSTRAINTS{
   0
}
```

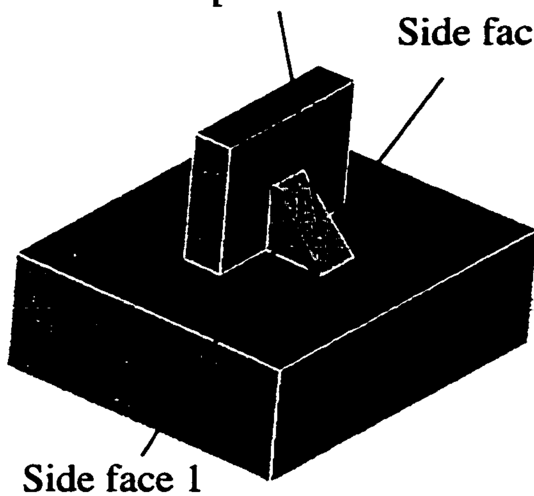
Feature Definition



<u>Feature CR-Graph</u>

Example Feature 14:

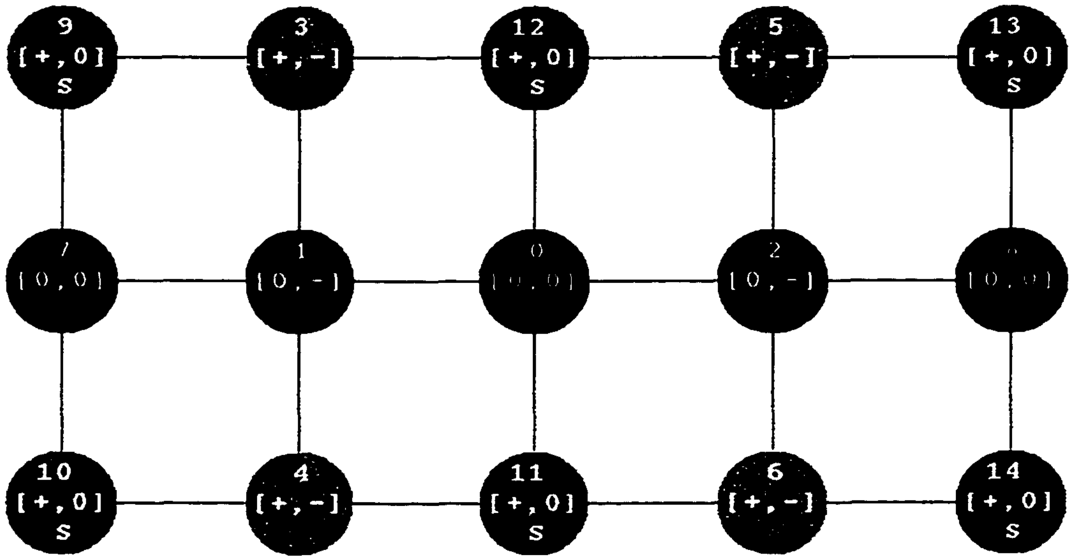
Feature: Web
Definition Type: CRGRP
Parameters: Length (l), Topface, Sideface 1, Sideface 2
Constraints: None




```
WEB CRGRP
CRNODES{
   15
   2 4 12 11 2 1
   3 4 0 3 4 7
   3 4 0 6 5 8
   6 3 1 12 9
   6 3 1 11 10
   6 3 2 12 13
   6 3 11 2 14
   2 3 1 9 10
   2 3 13 14 2
   5 2 7 3
   5 2 7 4
   5 3 0 4 6
   5 3 0 5 3
   5 2 8 5
   5 2 8 6
}

ATTRIBUTES{
   4
   topface  TOPFACE    0
   sideface1  SIDEFACE   7
   sideface2  SIDEFACE   8
   length   EDGELENGTH  1
}
CONSTRAINTS{
   0
}
```

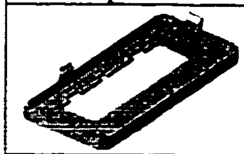
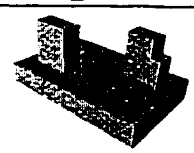
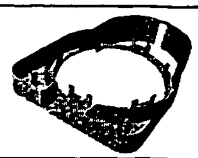
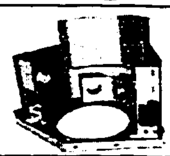
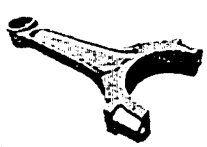
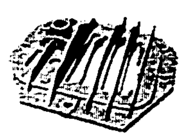
Feature Definition

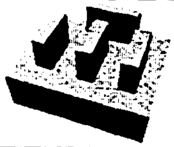
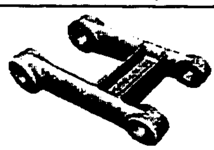
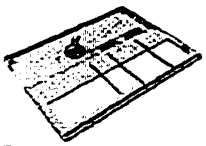
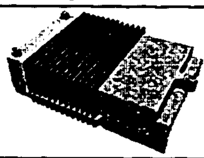
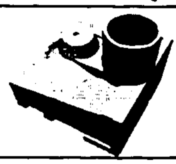
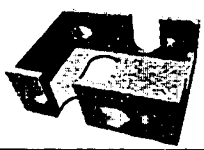
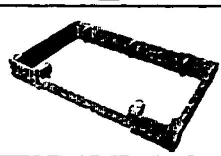
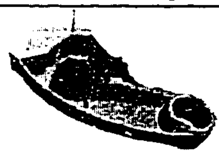
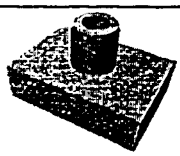


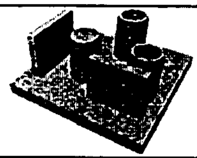
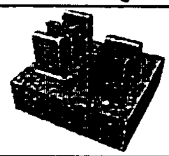
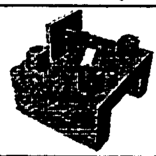
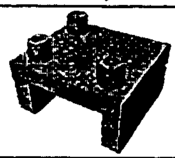
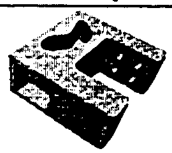
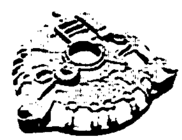
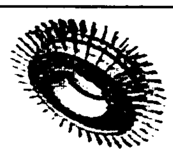
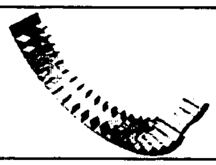
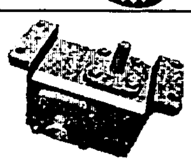
Feature CR-Graph

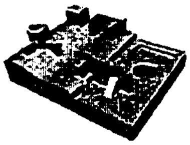
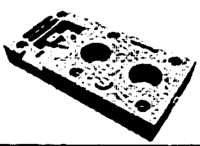
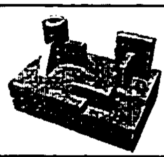
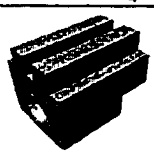
10 Appendix C: Example Parts used in Feature Recognition

In the table given below, N_{faces} is the number of faces, N_{edges} is the number of non-neutral edges, N_{total} is equal to Nf_{aces} + N_{edges} and T is the time taken to compute the CR-Graph of the part in seconds.

Picture of the part	Part Name	N_{faces}	N_{edges}	N_{total}	T (seconds)
	Abscover	52	126	178	87
	Approx	18	42	60	23
	Base	126	338	464	214
	Bulkhead	302	641	943	553
	Conrod	146	96	242	97
	Coverrear	710	858	1568	828
	Coverb	331	432	763	450
	Coverc	365	605	970	628

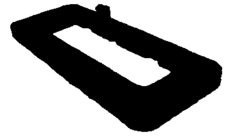
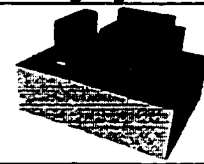
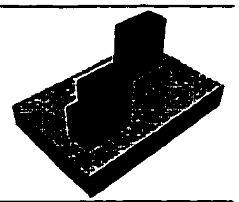
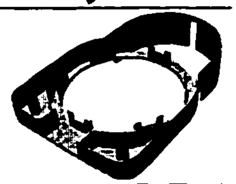
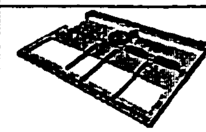
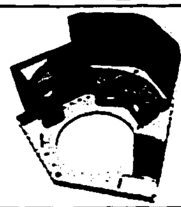
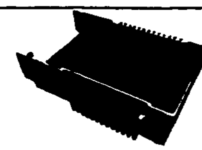
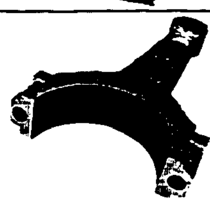
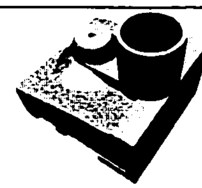
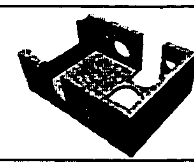
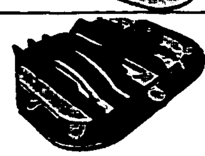
	Fillettest	52	54	106	48
	Frame	132	61	193	87
	Gadh1	381	853	1234	626
	Housing	296	835	1131	616
	Htoolbase	38	87	125	53
	Parker	38	90	128	74
	Piston	296	638	934	638
	Pmtest	117	308	425	184
	Proemdl	108	283	391	175
	Prt1	12	20	32	14

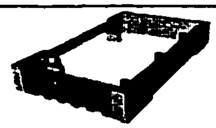
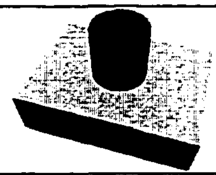
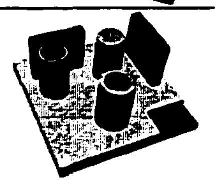
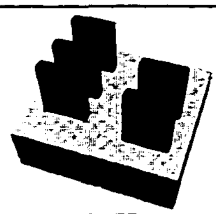
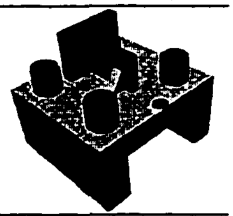
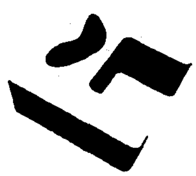
	Prt5	48	59	107	49
	Prt6	75	58	133	55
	Prt7	60	125	185	89
	Prt9	25	44	69	35
	Regli	44	90	134	66
	Rfx2u	1477	380	1857	938
	Rjf20	231	232	463	182
	Rjf22	741	942	1683	787
	Rjf6	272	552	824	411
	S718	269	585	854	447

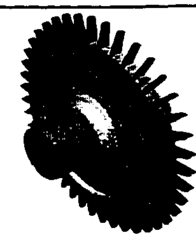
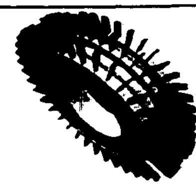
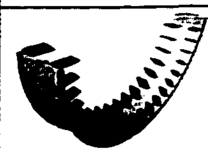
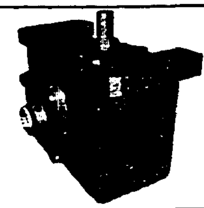
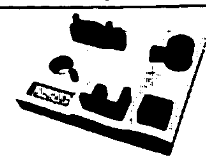
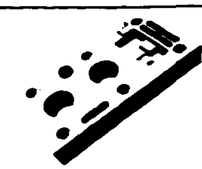
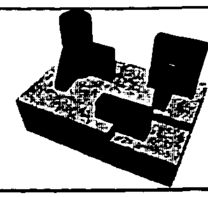
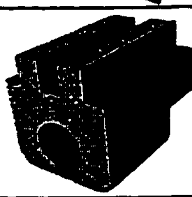
	Slide2	147	151	298	132
	Team	155	271	426	257
	Test1	198	467	665	345
	Test2	128	53	181	67
	Toolhold	21	48	69	29

11 Appendix D: Feature Recognition Results

The results of feature recognition on the parts in Appendix C are presented below. The feature recognition has been performed on a 180MHz, R5000, SGI O2 Machine with 96MB RAM.

Part Name	Features Found	Picture of Part with features highlighted	Part Name	Features Found	Picture of Part with features highlighted
Abscover	L-Rib, Pocket, Slot		Fillettest	Rib	
Approx	Rib, Slot		Frame	Hole, Slot	
Base	Boss, Hole, Through Pocket, Blind Hole		Gadh1	Hole, Pocket, Slot, Rib, Boss, Blind Hole	
Bulkhead	Hole, Slot, Through Pocket, Countersink Hole, Blind Hole		Housing	Fin, Pocket, Slot, Boss, Hole, Blind Hole	
Conrod	Hole		Htoolbase	Blind Hole, Slot	
Coverb	Through Pocket, Rib, Pocket, Slot		Parker	Hole, Slot, Through Slot	
Coverc	Through Pocket, Rib, Pocket, Slot		Piston	Hole, Slot	

Pmtest	Slot, Rib, Boss, Hole	
Proemdl	Slot, Hole, Boss, Open Slot	
Prt1	Blind Hole, Boss	
Prt5	Blind Hole, Boss, Rib, Corner Slot	
Prt6	Rib	
Prt7	Hole, Pocket, Rib, Web, Slot, Blind Hole	
Prt9	Boss, Slot, Blind Hole	
Regli	Slot, Pocket, Blind Hole	

Rjf20	Blade, Hole	
Rjf22	Blade, Hole	
Rjf6	Rib	
S718	Hole, Blind Hole, Rib, Web, Boss, Slot	
Slide2	Hole, Web, Boss, Pin, Rib, Pocket	
Team	Hole, Boss, Pocket, Blind Hole, Blind Slot	
Test2	Boss, Web, Rib, Blind Hole	
Toolhold	Hole, Slot, Open Slot	

Computation time for feature recognition

The time to recognize features using the different node matching algorithms is shown in the following tables. In each table,

Column 1 corresponds to the number of nodes in the feature graph,

Column 2 corresponds to the time in seconds to perform the graph match,

Column 3 corresponds to the number of features found after the graph match and

Column 4 corresponds to the number of nodes in the part/primitive graph in which the graph match is performed.

a) Time for feature recognition using CR-Graph Inexact match on the part

Num. Feature Nodes	Time (s) for recognition	Num. Features Found	Num. Part Nodes
10	0.03	2	1745
25	0.05	2	1289
25	0.06	2	1289
25	0.06	2	1745
32	0.02	1	130
4	0.01	1	1289
4	0.01	1	2586
4	0.01	12	1764
4	0.01	16	1713
4	0.01	2	188
4	0.01	2	2266
4	0.01	3	1305
4	0.01	3	2498
4	0.01	3	364
4	0.02	1	779
4	0.02	3	2570
4	0.03	3	3103
45	0.2	1	1764
8	0.01	1	1289
9	0.04	1	1745

Table 11-1: Computation time for CR-Graph Inexact match

b) Time for feature recognition using CR-Graph Approximate match on the part

Num. Feature Nodes	Time (s) for recognition	Num. Features Found	Num. Part Nodes
10	0.04	1	1764
10	0.05	1	2570
10	0.07	4	1745
14	0.02	1	491
14	0.02	2	253
19	0.04	1	491
21	0.01	2	188
21	0.02	1	364
21	0.02	2	188
21	0.02	4	180
21	0.02	5	209
21	0.03	1	130
21	0.03	1	364
21	0.03	2	253
21	0.03	4	180
21	0.03	5	209
21	0.09	1	1896
21	0.12	1	356
21	0.16	1	907
21	0.21	1	1764
21	0.23	13	1713
21	0.38	7	2586
21	0.81	2	364
23	0.09	1	1764
29	0.05	1	491
30	0.08	3	722
31	0.14	2	1289
37	0.26	1	1764
37	0.57	1	2586
39	0.07	1	722
39	0.49	2	2586
4	0.01	1	1289
4	0.01	1	209
4	0.01	1	2570
4	0.03	8	2266
43	0.38	1	2586
45	0.51	2	2586
69	0.7	1	2586
7	0.01	1	253
8	0.01	1	1305
8	0.01	1	356
8	0.01	1	996
8	0.01	4	1305
8	0.02	2	1289
8	0.02	3	1764
8	0.03	2	1745
8	0.07	4	1764
8	0.13	1	2266
9	0.01	1	491

Table 11-2: Computation time for CR-Graph Approximate match

c) Time for feature recognition using CR-Graph Exact match on the part

Num. Feature Nodes	Time (s) for recognition	Num. Features Found	Num. Part Nodes
10	0.02	2	1745
11	0.01	1	722
11	0.02	2	722
11	0.95	2	3103
12	0.01	1	1289
127	37.7	26	2498
14	0.03	2	1289
15	0.01	1	364
17	0.06	3	907
21	0.02	1	364
249	76.67	13	2498
25	0.01	1	254
26	0.05	2	1289
26	0.05	2	1745
26	0.37	41	1736
26	2.01	41	1736
3	0.15	37	3103
3	2.75	39	3103
3	3.41	39	3103
3	3.7	41	779
3	4.56	41	779
31	0.09	1	1289
31	0.12	1	1745
31	0.12	2	1745
39	0.37	2	2586
4	0.01	1	722
4	0.01	1	805
4	0.01	2	2498
4	0.04	41	779
4	0.14	39	3103
43	0.36	1	2586
47	0.06	2	356
5	0.01	10	2570
5	0.01	5	1289
5	0.01	5	1745
5	0.01	8	2266
5	0.02	10	1764
5	0.08	41	3103
53	0.13	1	1289
53	0.18	1	1745
7	0.01	1	1713
7	0.01	2	1896
9	0.01	1	1289
9	0.03	1	1745

Table 11-3: Computation time for CR-Graph Exact match

d) Time for feature recognition using CR-Graph Approximate match on Primitive Shapes

Num. Feature Nodes	Time (s) for recognition	Num. Features Found	Num. Primitive Nodes
15	0.06	1	15
21	0.07	1	21
21	0.23	1	21
21	0.26	1	21
21	0.34	1	21
21	0.36	1	21
3	0.1	1	3
3	0.11	1	3
3	0.12	1	5
3	0.13	1	5
3	0.14	1	3
3	0.15	1	3
3	0.16	1	3
3	0.17	1	3
3	0.18	1	3
3	0.19	1	3
3	0.2	1	3
3	0.21	1	3
3	0.23	1	3
3	0.25	1	3
3	0.29	1	3
3	0.35	1	3
3	0.36	1	3
3	0.37	1	3
3	0.38	1	3
3	0.39	1	3
3	0.41	1	3
3	0.42	1	3
3	0.43	1	3
3	0.44	1	3
3	0.55	1	3
3	0.6	1	8
3	0.63	1	3
3	0.67	1	8
3	0.68	1	7
3	1.15	1	3
3	1.19	1	26
3	1.2	1	3
5	0.01	1	5
5	0.02	1	5
5	0.03	1	5
5	0.14	1	5
9	0.01	1	9
9	0.02	1	9
9	0.03	1	9
9	0.13	1	9
9	0.14	1	9
9	0.15	1	9
9	0.16	1	9

Table 11-4: Computation time for CR-Graph Approximate match on Primitive Shapes