INFERS: An intelligent feature recognition system for rotational parts.

Dengzhou. Qi
University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation
NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.
INFERS: AN INTELLIGENT FEATURE RECOGNITION SYSTEM
FOR ROTATIONAL PARTS

by
Dengzhou Qi

A Thesis
submitted to the
Faculty of Graduate Studies and Research
through the Department of
Industrial Engineering in Partial Fulfillment
of the requirements for the Degree
of Master of Applied Science at
the University of Windsor

Windsor, Ontario, Canada
1991
The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

I hereby declare that I am the sole author of this thesis. I authorize the University of Windsor to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Dengzhou Qi

I further authorize the University of Windsor to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Dengzhou Qi
The University of Windsor requires the signatures of all persons using or photocopying this thesis. Please sign below, and give an address and a date.
ABSTRACT

Most existing CAD systems do not provide part feature information which is essential for process planning. A fully automatic part feature recognition system serves as an efficient link between CAD and CAPP. A part feature recognition system for rotational parts is proposed in this research work. The system, which is named as INFERS, consists of three modules, capable of recognizing rotational parts with multiple parallel axes. Human interpretation for feature recognition is eliminated by INFERS during the whole process. Except for the data extraction section, INFERS is independent of CAD systems by which the part is drawn. The output of the system can be used as part feature input for a CAPP system. INFERS serves as a valuable link between CAD and CAPP systems.
DEDICATION

To my wife and son

Xiaomei and Tian
ACKNOWLEDGEMENTS

I would like to take this opportunity to thank my supervisor Dr. N. Singh for his encouragement, support and fruitful guidances. I would like to thank the committee members, Dr. R.S. Lashkari, Dr. S.P. Dutta and Dr. V.M. Huynh, for their suggestions. To Ms. Jacque Mummyy I extend my gratitude for her assistance and friendship. I would like to thank Mr. Tom Williams for his assistance in using computer facilities. Also, I would like to thank Miss Nancy Peel.
TABLE OF CONTENTS

ABSTRACT vi
DEDICATION vii
ACKNOWLEDGEMENTS viii
LIST OF TABLES xi
LIST OF FIGURES xii

CHAPTER 1: INTRODUCTION 1
1.1 Background 1
1.2 The Problem of Part Feature Recognition 2
  1.2.1 Statement of the Problem 2
  1.2.2 The Role of Feature Recognition 3
1.3 Organization of the Proposed Research 3

CHAPTER 2: LITERATURE REVIEW 5
2.1 Geometric Modeling Techniques 5
  2.1.1 Wire Frame Modeling Scheme 5
  2.1.2 Surface Modeling Scheme 6
  2.1.3 Solid Modeling Scheme 6
2.2 Literature Review 8
  2.2.1 Classification of Feature Recognition Systems 8
  2.2.2 Approaches to Feature Recognition 8
  2.2.3 Representative Part Feature Recognition Systems 14
2.3 General Comments of Literature Review 20
2.4 Motivations for the Proposed Research 20
2.5 Objectives of the Proposed Research 21

CHAPTER 3: THE INFERS SYSTEM 22
3.1 Overview of INFERS 22
  3.1.1 Geometric Representation of a Part 22
  3.1.2 Feature Recognition Approach 23
  3.1.3 Programming Language 23
  3.1.4 Structure of INFERS 23
3.2 Module I: Data Extraction and Preprocess 24
3.3 Module II: Identify Surfaces of a Part Drawn 28
3.4 Module III: Recognize Part Features 30

CHAPTER 4: IMPLEMENTATION OF INFERS 34
4.1 Example 1: A part with through hole 34
4.2 Example 2: A stepped part 37
4.3 Example 3: A complex part I with multiple axes 40
4.4 Example 4: A complex part II with multiple axes 43

CHAPTER 5: EVALUATION AND PROSPECTIVE APPLICATIONS OF INFERS 46

5.1 Evaluation 46
5.2 Prospective Applications 47
  5.2.1 Generating Process Plans 47
  5.2.2 Classifying and Coding for GT 49
  5.2.3 Creating NC Tape Images 49
  5.2.4 Design for Assembly Analysis 49

CHAPTER 6: CONCLUSIONS AND SUGGESTIONS FOR FUTURE RESEARCH 50

6.1 Contributions of the Research 50
6.2 Suggestions for Future Research 50

REFERENCES 52
APPENDIX 59
Program Listing 60
VITA AUCIRIS 114
## LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE NO.</th>
<th>PAGE NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>9</td>
</tr>
<tr>
<td>4.1-1</td>
<td>36</td>
</tr>
<tr>
<td>4.1-2</td>
<td>36</td>
</tr>
<tr>
<td>4.2-1</td>
<td>39</td>
</tr>
<tr>
<td>4.2-2</td>
<td>39</td>
</tr>
<tr>
<td>4.3-1</td>
<td>42</td>
</tr>
<tr>
<td>4.3-2</td>
<td>42</td>
</tr>
<tr>
<td>4.4-1</td>
<td>44</td>
</tr>
<tr>
<td>4.4-2</td>
<td>45</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE NO.</th>
<th>PAGE NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>9</td>
</tr>
<tr>
<td>4.1-1</td>
<td>36</td>
</tr>
<tr>
<td>4.1-2</td>
<td>36</td>
</tr>
<tr>
<td>4.2-1</td>
<td>39</td>
</tr>
<tr>
<td>4.2-2</td>
<td>39</td>
</tr>
<tr>
<td>4.3-1</td>
<td>42</td>
</tr>
<tr>
<td>4.3-2</td>
<td>42</td>
</tr>
<tr>
<td>4.4-1</td>
<td>44</td>
</tr>
<tr>
<td>4.4-2</td>
<td>45</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE NO.</th>
<th>DESCRIPTION</th>
<th>PAGE NO.</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Functional structure of INFERS</td>
<td>25</td>
</tr>
<tr>
<td>3.2</td>
<td>Flow chart for Module I</td>
<td>27</td>
</tr>
<tr>
<td>3.3</td>
<td>Flow chart for Module II</td>
<td>29</td>
</tr>
<tr>
<td>3.4</td>
<td>Flow chart for Module III</td>
<td>33</td>
</tr>
<tr>
<td>4.1</td>
<td>A part with through hole (example 1)</td>
<td>34</td>
</tr>
<tr>
<td>4.2</td>
<td>A stepped part (example 2)</td>
<td>37</td>
</tr>
<tr>
<td>4.3</td>
<td>A complex part I with multiple axes (example 3)</td>
<td>40</td>
</tr>
<tr>
<td>4.4</td>
<td>A complex part II with multiple axes (example 3)</td>
<td>43</td>
</tr>
<tr>
<td>5</td>
<td>Interfacing INFERS with CAD and CAPP</td>
<td>48</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

1.1 Background

Automation is the source of many technological improvements in the manufacturing area. Especially, with the rising importance of industrial competitiveness, the need for these improvements in the area of manufacturing is becoming acute. Automation in manufacturing can speed product turnaround, reduce the need for retooling, and lead to a more efficient allocation of resources. This automation can be achieved by implementing an integrated Computer Aided Design (CAD) system and Computer Aided Manufacturing (CAM) system.

CAD/CAM implies an integrated process continuum where computer technology is incorporated in the design and manufacture of products. However, CAD/CAM is not yet an integrated continuum, but rather a series of 'islands of automation' [Wang and Wysk 1988]. There are a number of obstacles to the implementation of a fully integrated, and automated manufacturing system. One of the major obstacles is the lack of smooth information flow between CAD and CAM. These two functions have traditionally developed as independent activities, with no feedback from CAM to CAD to reflect manufacturability considerations [Ssemakula and Satsangi 1990].

CAPP (Computer Aided Process Planning) systems usually serve as a link
between CAD and CAM. However, it is only a partial link, because most of the existing CAD/Drafting systems do not provide part feature information which is essential data for CAPP. To solve the CAD/CAPP interface problem, feature recognition is one of the efficient approaches. Even for the present feature-based design systems, feature recognition is still necessary [Sakurai and Gossard 1990]. Therefore, the proposed research focuses on part feature recognition.

1.2 The Problem of Part Feature Recognition

1.2.1 Statement of the Problem

Feature recognition is a CAD interface problem which requires the matching/extraction of embedded manufacturing features and their parameters/properties [Wang and Chang 1990]. In essence, feature recognition is to extract design data from a CAD/drafting database and then interpret the design information in terms of features. Formally, feature recognition converts a general CAD model into an application specific feature model. A generic part feature recognition system should be able to:

- extract design information of a part drawn from a CAD database,
- identify all surfaces of the part, and
- recognize, reason and/or interpret these surfaces in terms of part features.

All of the functions should be performed automatically, without any human interpretation or intervention. The main objective of such systems is to bridge the gap between CAD data base and automated
process planning systems by automatically distinguishing the features of a part from the design information stored in a CAD/drafting system. Once the features are recognized, CAPP or APP (Automated Process Planning) systems could develop process plans to make the part without human interpretation [Bedworth et al. 1991].

1.2.2 The Role of Feature Recognition

Part feature recognition plays an important role in the integration of CAD and CAM. It is clear that, based on the discussion in the previous section, part feature recognition systems take CAD database as input and produce output about design information in terms of part features, while, the feature information will be essentially used as one of the inputs to a Computer Aided Process Planning system. It is shown that part feature recognition system serves as a direct link between CAD and CAPP, even the first step toward the integration of CAD/CAPP/CAM [Choi, Barach and Anderson 1984, Li and Agida 1987, Graves et al. 1989].

1.3 Organization of the Proposed Research

The research is organized as follows. A study of geometric modeling techniques and a comprehensive review of part feature recognition approaches are conducted in chapter 2. Motivations and objectives of the proposed research are also included in chapter 2. In chapter 3, the proposed part feature recognition system named INFERS (INtelligent FEature Recognition System for rotational parts) is described in detail. The implementation of INFERS is demonstrated in chapter 4. In chapter 5, the evaluation of the system INFERS is given. The
prospective applications of INFERS are discussed in chapter 5.
Finally, conclusions and suggestions for future research are provided in chapter 6.
CHAPTER 2

LITERATURE REVIEW

2.1 Geometric Modeling Techniques

As a preliminary step of the proposed research, the geometric modeling techniques used by most CAD systems are studied. In general, there are three types of representation schemes being used: wire frame modeling scheme, surface modeling scheme and solid modeling scheme.

2.1.1 Wire Frame Modeling Scheme

Wire frame modeling technique is a representation scheme. Many CAD/drafting systems use 3-D wire frame modeling technique to represent components [Bedworth et al. 1991]. The wire frame modeling scheme uses points, curves (i.e. lines, circles, arcs), etc. to define objects. For example, a user may, with 3-D wire frame models, enter 3-D vertices, say (x, y, z), and then join the vertices to form a 3-D object. However, a user may not obtain complete and unambiguous information about an object from the wire frame models. The main reason is that the wire frame representation scheme uses only points, lines, curves to define objects without surface definition. As a result, it is short of representation knowledge in some cases, for instance, an object with sides sculptured to bulge outward or inward may not be defined completely by this model. Further, it is hard to tell the interior and exterior of an arbitrarily given part image from monitor.
2.1.2. Surface Modeling Scheme

Surface modeling scheme is another widely used modeling technique, which defines objects by their bounding faces. Surface modeling systems contain definitions of surfaces, edges, vertices. In such systems, a user may input the vertices, edges of a given work piece in such a manner that outlines or bounds one face at a time. Surface modeling systems offer better graphic interaction, although they are more difficult to create than wire frame scheme [Chang and Wysk 1985]. In the sense of surface modeler, surface modeling scheme is better than wire frame modeling scheme. However, it still has some drawbacks. It does not provide the information of the topology of the entities such as the concept of the component "inside" and "outside". As a result, with surface models, a user may still not be able to distinguish the interior and exterior of an object on the monitor. As a matter of fact, a surface modeling system may not guarantee that the user has designed a realizable object, that is, the collection of surfaces may not define a physical part [Bedworth et al. 1991].

2.1.3 Solid Modeling Scheme

In a solid modeling system, objects are defined by primitive shapes called building blocks. There are at least six methods for constructing a 3D object within a CAD solid modeling system. They are [Bedworth et al. 1991]:
- Pure Primitive Instancing (PPI),
- Spatial Occupancy Enumeration (SOE),
- Cell Decomposition (CD),
- Sweeping (S),
- Constructive Solid Geometry (CSG), and
Boundary Representation (BREP).
The PPI method recalls already-stored descriptions of primitive solids, such as blocks, spheres, and cylinders, and applies a scaling transform to the primitive. SOE subdivides 3D space into small volumes and classifies these volumes as either empty, full, or partially full of the solid. CD is similar to SOE, but CD starts with the object, not the 3D space, and subdivides that object into small volumes, all of which are full of the part. Sweeping method involves the notion that a polygon or polyhedron moving through space sweeps out a solid described by the polygon or polyhedron and the trajectory.

The last two methods, CSG and BREP are the most popular methods. The CSG or set-theoretic [Woodwark 1988] modeling method uses Boolean set operators to construct the solid geometry. The database of CSG contains the primitive volumes with their respective Boolean operators used. The BREP representation contains the boundary elements of the resultant object. The BREP database stores primitives such as faces, edges and vertices and their connective relationships [Chuang and Henderson 1990]. Although solid modeling technique has some advantages over the other two methods, it still has some shortcomings. For instance, BREP provides implicit feature representation although explicit model representation, the CSG tree is very complex. The CSG representation scheme is non-unique, for example, a slot may be constructed by adding two small blocks to a larger block or by subtracting two small blocks from a large block. The implicit information provided by solid modelers implies that interpretation of the geometry and/or topology may still be necessary.
2.2 Literature Review

The integration of CAD and CAM has long been a goal of researchers. During the proposed research, about sixty technical papers related to part feature recognition were reviewed. About thirty representative part feature recognition systems were studied and listed in Table 2-1 with necessary information such as developers, name or title of feature recognition system, main recognizable features, etc.

2.2.1 Classification of Feature Recognition Systems

The existing part feature recognition systems may be classified in two ways:

(i) According to the CAD system environment, classify them as:
   - 2-D part feature recognition systems,
   - 2½ D part feature recognition systems, and
   - 3 D part feature recognition systems.

(ii) Based on the types of recognizable parts, classify the part feature recognition systems as:
   - prismatic part feature recognition systems, and
   - rotational part feature recognition systems.

2.2.2 Approaches to Feature Recognition

Down the stream of feature recognition, the approaches or techniques used by existing systems may be summarized as follows [Alting and Zhang 1989, Wang and Chang 1990].
<table>
<thead>
<tr>
<th>Developer</th>
<th>Part Feature Recognition System</th>
<th>Main Recognizable Features</th>
<th>Geometric Modeling Schemes</th>
<th>Ref.&amp; Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. Chunag</td>
<td>3D Shape pattern recognition using Ver.Class.and VEG</td>
<td>Prismatic</td>
<td>BREP</td>
<td>[8] 90</td>
</tr>
<tr>
<td>4. Dong</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wozny</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Henderson</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. Henderson</td>
<td>FEATURES</td>
<td>Holes, slots, pockets</td>
<td>BREP</td>
<td>[17] 84</td>
</tr>
<tr>
<td>Anderson</td>
<td></td>
<td></td>
<td></td>
<td>[18]</td>
</tr>
<tr>
<td>Musti</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. Henderson</td>
<td>FRAPP</td>
<td>Holes, slots etc.</td>
<td>BREP</td>
<td>[20] 88</td>
</tr>
<tr>
<td>Chang</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10. Joshi</td>
<td>Graph-based heristic for recog.mach.surf. from 3D</td>
<td>Polychreral features such as slots, steps pockets</td>
<td>BREP</td>
<td>[25] 88</td>
</tr>
<tr>
<td>Chang</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11. Kakino</td>
<td>A new method of parts descrip. for CAPP</td>
<td>Grooves, steps flages</td>
<td>BREP &amp; CSG</td>
<td>[27] 77</td>
</tr>
<tr>
<td>14. Kyprianou</td>
<td>Shape Class. in CAD</td>
<td>Rotational part family</td>
<td>BREP</td>
<td>[29] 80</td>
</tr>
<tr>
<td>Developer</td>
<td>Part Feature Recognition System</td>
<td>Main Recognizable Features</td>
<td>Geometric Modeling Schemes</td>
<td>Ref.&amp; Date</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------------------</td>
<td>----------------------------</td>
<td>---------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>15. Li</td>
<td>A Part-Feature recog. system</td>
<td>Rotational part family</td>
<td>2D wire frame</td>
<td>[34] 88</td>
</tr>
<tr>
<td>16. Lee</td>
<td>Integration of SM &amp; DB manage. for CAD/CAM</td>
<td>Holes, slots, etc.</td>
<td>CSG</td>
<td>[30] 84</td>
</tr>
<tr>
<td>17. Lee</td>
<td>FEXCAPP</td>
<td>Prismatic</td>
<td>BREP</td>
<td>[32] 90</td>
</tr>
<tr>
<td>18. Lee &amp; Fu</td>
<td>Extraction &amp; unification of manufacturing features</td>
<td>Fillet, round chamfer, necks etc.</td>
<td>CSG</td>
<td>[31] 87</td>
</tr>
<tr>
<td>19. Liu Srinivasan</td>
<td>Gen. PP using syntactic pat. recognition</td>
<td>Holes, etc.</td>
<td>BREP</td>
<td>[37] 84</td>
</tr>
<tr>
<td>20. Perng,</td>
<td>Automatic 3D mach. feat. extract.</td>
<td>Holes, steps pockets, etc.</td>
<td>CSG</td>
<td>[43] 90</td>
</tr>
<tr>
<td>21. Rosario</td>
<td>Extract.of geom. features from CAD</td>
<td>Rotational</td>
<td>3D wire frame</td>
<td>[45] 90</td>
</tr>
<tr>
<td>22. Sakurai Gossard</td>
<td>Recog. shape features in solid</td>
<td>Steps, slots holes, etc.</td>
<td>Incorp. CSG&amp;BREP</td>
<td>[47] 90</td>
</tr>
<tr>
<td>23. Sahay</td>
<td>A method for recog. feat. in 2D cylind. parts</td>
<td>symmetric rotational part family</td>
<td>2D wire frame</td>
<td>[46] 90</td>
</tr>
<tr>
<td>25. Staley</td>
<td>Using syntactic pattern recog. to extract feat.</td>
<td>holes</td>
<td>BREP</td>
<td>[51] 83</td>
</tr>
<tr>
<td>Henderson Anderson</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27. Wang</td>
<td>Intelligent Reason for PP</td>
<td>Symmetric Rotational Parts</td>
<td>Wire frame</td>
<td>[52] 86</td>
</tr>
<tr>
<td>Developer</td>
<td>Part Feature Recognition System</td>
<td>Main Recognizable Features</td>
<td>Geometric Modeling Schemes</td>
<td>Ref. &amp; Date</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------------</td>
<td>---------------------------</td>
<td>---------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>28. Woo</td>
<td>Interfacing</td>
<td>Holes, slots</td>
<td>CSG</td>
<td>[59] 84</td>
</tr>
<tr>
<td></td>
<td>Solid modeling to CAD/CAM</td>
<td>etc.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29. Woo</td>
<td>Computer-aided rec. of volumetric designs</td>
<td>Cavities such as holes, slots etc.</td>
<td>CSG</td>
<td>[57] 77</td>
</tr>
<tr>
<td>Chang</td>
<td>For APP</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(1) Syntactic Pattern Recognition Approach

This approach is based on the theory of syntactic pattern recognition. Syntactic pattern recognition borrows most of its analysis methods from formal language theory, and the formulation of the basic problem statement is also made by analogy. In syntactic pattern recognition, complex patterns to be recognized are decomposed into simple sub patterns and relations among sub patterns. This decomposition is recursive until the patterns primitives (the simplest sub patterns) are obtained. Patterns primitives can be easily recognized because of their simple descriptions [Fu 1982]. The structural information about a pattern (i.e. the relations among sub patterns) is analogous to the syntax or grammar of a formal language. Sequences of sub patterns are concatenated to form more complex patterns, just as sentences are formed as collections of simper elements (words), and words in turn are formed from simpler elements (characters). [Staley et al. 1983].

Some researchers have employed syntactic pattern recognition approach in part feature recognition [Srinivasan et al. 1985, Staley et al. 1983, Kyprianou 1980, Zhang et al. 1989, Choi and Barash 1985, Dong and Wozny 1988, etc.]. It should be noticed that the syntactic pattern recognition approach has long been chosen to extract features from a CAD database; however its nature limits the application to 2D and 2½D at most.

(2) Geometry Decomposition Approach

The main point of this approach is that features, as primitives, are viewed as volumes to be removed by a unit-machine operation. A minimum-volume convex body (for stock) and its features (for material to be removed) is derived from a given representation of a 3D object.
Woo [1984] and Perng et al. [1990] used the geometry decomposition approach to part feature recognition.

(3) Expert System Rule/Logic Approach
An expert system is such a system that uses knowledge, facts and reasoning techniques, with the aid of computers, to solve problems that normally require the abilities of human experts. The knowledge the expert system uses consist of either rules or experience information about the behavior of the elements of a particular subject domain [Martin and Oxman 1987]. Researchers such as Sakurai and Gossard (1990), Liu (1984), Wang and Wysk (1990), Wang and Wysk (1987), Kung (1984), Mortensen and Belnap (1989), etc., used the expert system rule/logic approach for part feature recognition.

(4) Graph-Based Approach
Graph-based approach is another technique for part feature recognition. This approach is mainly based on the theory of graphs. A graph \( G \) consists of a finite nonempty set \( V=V(G) \) of points, say \( p \), together with a prescribed set \( X \) of \( q \) unordered pairs of distinct points of \( V \). A graph is connected if all pairs of points are joined by a path. A maximal connected subgraph of \( G \) is called a connected component of \( G \). The connectivity \( K=K(G) \) of a graph \( G \) is the minimum number of points whose removal results in a disconnected or trivial graph [Gavankar 1990]. A graph-based model can represent a solid model with data pointers linking together an object's faces, edges, and vertices. Edge-vertex graphs and edge-face graphs are two types of commonly used graphs in the representation of solid models [Gavankar 1990]. Several researchers used this approach in the field of part
feature recognition [Joshi and Chang 1987, Joshi and Chang 1988 and Gavankar and Henderson 1990].

(5) Combination Approach

The approach used for feature recognition by some researchers can not be classified exactly as one of the approaches mentioned here. They used the combination of the techniques cited above [Zhang et al. 1989].

2.2.3 Representative Part Feature Recognition Systems

(1) Prismatic Part Feature Recognition Systems

The techniques used for feature recognition in prismatic part feature recognition systems are feature classification by heuristic rules, and feature recognition by pattern matching or graph matching methods [Perng, Chen and Li 1990, Reingold, Nievergelt and Deo 1977]. There are quite a few publications dealing with prismatic part feature recognition. Choi (1984), Chuang and Henderson (1990), Gavankar and Henderson (1990), Henderson and Chang (1988), Kung (1984), Lee (1990), etc., developed systems or methodologies for recognizing form features such as slots, pockets, holes and so on. In the following section, seven systems are briefly introduced.

Gavankar and Henderson (1990) developed an algorithm for extracting features such as protrusions and blind holes. Features extracted by the system are from boundary representations. Only the topological information of a solid model is needed for feature extraction in this system, which implies that one graph theoretic algorithm will enable the extraction of features belonging to a certain connectivity class.
The feature extracted by the system is limited to protrusions and blind holes. Furthermore, the system only considered the feature extraction phase.

Henderson and Anderson (1984) and Henderson (1984) developed a system named FEATURES for feature extraction and recognition from CAD database. The system consists of a feature recognizer, extractor and organizer, which recognize manufacturing features from a CAD database and organize them hierarchically according to their position in the modeled part. The system uses the BREP representation of a part, while the BREP data is created by ROMULUS solid modeler. In this system, the feature rules are written and applied in PROLOG language. A 3D model in BREP representation is first converted into facts in Prolog language, and then features are recognized by the Prolog's pattern matching mechanism. The recognizable parts by the system are ones with cavities made up of various types of features such as slots, pockets and holes. The rules written are only for swept features, and the recognized holes are not classified into more specific types of holes.

Perng et al. (1990) proposed a system of automatic 3D machining feature extraction from 3D CSG solid input. This system converts the CSG tree representation of a given part into its equivalent DSG (Destructive Solid Geometry) tree representation first, and then recognize the types of machinable features from DSG tree. A DSG tree is a special case of CSG tree in which all geometric operations are of 'difference' type. A hierarchical machining feature definition is used in this system. The system can automatically recognize 18 machining features of prismatic parts, such as blind holes, through
holes, counter bores, pockets, steps, etc. The output of the system contains feature attributes including the dimensions and the possible tooling entrance faces. The limitations of the system are of being incapable of recognizing intersecting features and complex features such as tori and spheres.

In order to generate a tool path algorithmically, Grayer (1977) described a method for recognizing cavities which are represented by BUILD - a solid modeler storing solid models in their BREP form. The program was written in FORTRAN consisting of about 2500 lines. This system is limited to 2&1/2D parts, it can handle straight lines and circular areas.

Choi et al. (1984) represented a methodology for automatic recognition of machined surfaces from a 3D solid model in BREP representation. In order to store the information about the machined surfaces of the part, a set of surface data structure is defined in this methodology. The technique, used for finding the machined surface from the BREP data, is the syntactic pattern recognition method. Pattern matching rules considered both geometry and topology of features. The definition of features and rules are written in PASCAL language. The recognizable features are slots, pockets and holes. The limitations of the system is that it may not recognize surfaces of the work piece completely as machined surfaces and may not identify global constraints such as clamping requirements, etc.

Chuang and Henderson (1990) represented an approach to 3D feature recognition using vertex classification and vertex-edge graphs. The
solid modeler ROMULUS is used to store solid models in BREP form. The
PROLOG language is employed to convert the internal BREP of ROMULUS
into facts. Four types of vertices are defined: (a) three convex
dges, (b) two convex and one concave edges, (c) one convex and two
concave edges and (d) three concave edges. The two types of features
defined are pocket and slot in the system. However the system is not
valid for the following situation: surfaces having an order higher
than two, and having edges which change convexity and/or concavity
from one end to the other.
Sakurai and Gossard (1990) proposed a system for recognizing shape
features in solid models. The solid modeler used in this system
incorporated CSG tree and boundary representation. Graph matching
techniques is employed for feature recognition. In this system a shape
feature is defined as a single face or a set of contiguous faces
possessing certain characteristic facts in topology and geometry. The
system recognizes a shape feature by searching the solid model for
BREP sub graphs with the same properties. When a feature is
recognized, the geometry associated with it is removed from the
original solid model to produce a simpler solid model. Then the system
checks the simpler solid model to determine whether additional
features are revealed. If no additional features are found, the
process stops. The recognizable features are steps, slots and holes.
The limitations of the system are: (i) generic features can not be
recognized, (ii) some cases of intersecting features can not
recognized either.

(2) Rotational Part Feature Recognition Systems

The feature extraction techniques employed in rotational part feature

Srinivasan et al. (1985) developed a methodology for the extraction of manufacturing details from geometric models. In this system, a part is represented as a syntactic pattern made up of geometric primitives in 3D. Based on the syntactic pattern recognition technique, the recognizers were developed in terms of grammatical inference procedure and they took some known classified samples as training samples. The applications of this methodology are (i) feature classification for group technology and process selection and (ii) procedure for representation and machining of excess material. The implementation of the system is on axis symmetric rotational parts.

Li (1988) proposed a part feature recognition system for rotational parts. The feature recognition algorithm is based on the syntactic pattern recognition techniques. Twenty-four pattern primitives are defined in this system. Parts recognized by the system are represented with 2D wire frame models. A Part Definition Data Structure (PDDS) is designed in order to store recognizable part-feature data for the system. Part-definition data from CAD systems are extracted via IGES standard data format. The main limitation of the system is that parts recognized by the system have to be symmetrical around the center line.
Wang and Wysk (1988) developed a system named AIMS1—An Intelligent Machined Surface Identifier for rotational parts. The system consists of three parts: a preprocessor— which extracts all drawing entities of a given part from a CAD data base, a main processor— which searches and identifies all machined surfaces of the part, and post processor— which writes all machined surfaces identified into an output file in a certain format. This system deals with parts represented by 2D wire frame models. The process of part feature recognition with AIMS1 is fully automatic without any human interpretation, and it is independent of the CAD system used for drawing the part except the section of entity extraction. The limitation of the system is that it does not support non-symmetric rotational parts such as with multiple axes, keyways, etc.

Sahay et al. (1990) presented a methodology for recognizing features in two-dimensional cylindrical part designs. The algorithm proposed is, with feature definitions, based on the concept of features satisfying some generic, geometric properties, therefore, without resorting to heuristics, features can be recognized. The system deals with the parts defined by using a 2D wire frame representation scheme. Parts recognized by the system are symmetric rotational ones. Again, processing non-symmetric parts is beyond its capabilities. Two limitations are: (i) no duplicate points allowed in the input geometric data, and (ii) the center line of the part being collinear with the X-axis of the coordinate frame in which the geometry is defined.
2.3 General Comments of Literature Review

It is obvious, based on the review of the part feature recognition systems in the previous section, that the types of recognizable features are limited. For example, the three representative feature recognition systems for rotational parts [Li 1988, Wang and Wysk 1988, Sahay et al. 1990] are limited to center line symmetric parts. A part with multiple axes can not be recognized by their methodologies.

2.4 Motivations for the Proposed Research

(1) Feature Recognition Plays an Important Role in the Integration of CAD/CAM

The ability to automatically interpret an object from a CAD system is paramount to integrated CAD/CAM and CAPP. Fully integrated CAD/CAM systems have not come true because of the missing link between CAD and CAPP.

(2) The Need of Part Feature Recognition System for Rotational Parts

The integration of CAD and CAPP has long been a goal of researchers. Considerable achievements have been gained in part-feature recognition in the previous work, however, some limitations still remain. As pointed out earlier, the recognizable features by existing feature recognition systems such as AIMS [Wang and Wysk 1988], A Part-Feature Recognition System for Rotational Parts [Li 1988], etc., are limited to the parts which are symmetrical around the center line, which implies that a part with multi-axis may not be recognized by those approaches. This limitation blocks their practical applications.
2.5 Objectives of the Proposed Research

The main objective of the research is to develop a methodology or a system for recognizing features for rotational parts with multiple parallel axes. The system

1. can automatically extract the design information about a part drawn from a CAD database,

2. can automatically identify surfaces of the part, and

3. can intelligently recognize or reason the part features.

All of the functions will be performed without any human intervention or interpretation. The system can serve as a direct link between CAD and CAPP.
CHAPTER 3

THE INFERS SYSTEM

In this chapter, a part feature recognition system named INFERS (INtelligent part FEature Recognition System) is introduced. INFERS is capable of recognizing rotational parts with multiple axes. This chapter is organized as follows. In section 3.1 we overview the system from the view points of geometric representation, recognition technique employed and the structure of INFERS. The modules of INFERS are described in detail in sections 3.2, 3.3, and 3.4, respectively.

3.1 Overview of INFERS

3.1.1 Geometric Representation of a Part

Based on the geometric features of an object, a center line symmetric rotational part can be completely represented by using a one-view drawing in 2-D space. As described in previous works [Li 1988, Wang and Wysk 1988, Sahay et al. 1990], each point represents a circular edge, each line segment a cylindrical surface. For a nonsymmetric and/or multi-axes rotational part, however, one-view may not be enough to provide complete information. Drawings in 3-D space may be required to represent a part. Notice that a straight cylindrical surface may be represented by its two end faces (represented by two circles), a planar surface can be fully represented by its two edges (two line segments). The proposed feature recognition system is based on the above simplified boundary representation knowledge.
3.1.2 Feature Recognition Approach

A simple part or form feature is viewed by INFERS as a collection of primitives (the simplest patterns) such as end faces (circles), edges (line segments). For a complex part, it is decomposed into simple ones or form features. The feature recognition is performed by using syntactic pattern recognition, geometry decomposition and artificial intelligence techniques. INFERS is a kind of combination approach to feature recognition.

3.1.3 Programming Language

The C language is chosen as the programming language in the system. C language offers great flexibilities in programming. The system is written in modules, each module consists of pointer blocks and diverse functional blocks, which result in shortening computing time and reducing storage space.

3.1.4 Structure of INFERS

(1) Six Main Functions

From the function point of view, INFERS consists of six main functions. They are:

(a) Data extraction from CAD database.
(b) Preprocess data extracted.
(c) Identify surfaces of the workpiece.
(d) Determine surfaces relationship.
(e) Recognize part features.
(f) Write the output into files in a desired format.
(2) Four Knowledge Bases (rules and logic)

There are four knowledge bases (rule and logic) in the system.

(a) Data preprocess;
(b) Surfaces identification;
(c) Surfaces relationship determination;
(d) Feature recognition.

Knowledge base (a) is used in Module I, (b) in Module II, and (c) and (d) in Module III.

(3) Three Modules

INFERS consists of three modules, as shown in Figure 3.1:

Module I Data extraction and preprocess.
Module II Workpiece surface identification.
Module III Part feature recognition.

The output of each module will form the input of the following module as shown in Figure 3.1. Functions (1) and (2) are performed by Module I, function (3) by Module II and functions (4), (5) and (6) by Module III.

3.2 Module I: Data extraction and preprocess

First, Module I extracts all entities such as lines, curves from a CAD data file. Then, the entities extracted are preprocessed in a desired manner. The preprocessed data (or newly formed database) will meet the requirements of Module II, that is, the output of Module I ensures that no confusing or erroneous information is sent to Module II. The flow chart for Module I is shown in Figure 3.2.
The algorithm for Module I is described as follows:

Step 1: Access the database of CAD, extract all lines, circles and other curve entities.

Step 2: Preprocess data extracted, that is, rearrange lines, circles, etc. in a appropriate manner to be used by the following modules.

Step 2.1: Pick up lines which represent part axes.

Step 2.2: According to their relationship relative to the three coordinate planes, arrange lines into the following groups:

(i) xy-group in which lines are perpendicular to x-y coordinate plane,
(ii) xz-group in which lines are perpendicular to x-z coordinate plane,
(iii) yz-group in which lines are perpendicular to y-z coordinate plane, etc.
(iv) rearrange each line group according to the value of x or y or z coordinates.

Step 2.3: Rearrange curves:

(i) Rearrange curves with respect to each of the part axes to obtain group one, say $G^1_i$, (i=1,2,...ncl) (ncl: total number of axes of the part).

(ii) Rearrange $G^1_i$ with respect to coordinate along with one of the part axes, obtain group two, say $G^2_i$.

Step 3: Write the data preprocessed into new files to obtain preprocessed database ready to be used by module II.
FIGURE 3.2 FLOW CHART FOR MODULE I
Actually, the preprocess function is performed based on a set of rules and logic expressions. The rules are of the type:

\[
\text{IF conditions,} \\
\text{THEN conclusions.}
\]

For example, a rule for grouping a line into the group in which lines are perpendicular to yz_coordinate plane is stated as (assuming the starting and ending points are \(\{x_1, y_1, z_1\}\) and \(\{x_2, y_2, z_2\}\) respectively):

\[
\text{If } y_1 = y_2, \ z_1 = z_2 \text{ and } x_1 \neq x_2, \\
\text{then } \text{this line belongs to the group.}
\]

A rule for grouping a circle into \(G^1_i\) is (assuming the center of the circle is \(\{x'_i, y'_i, z'_i\}\), one of the axes of the given part along z-coordinated axis is \(\{x'_j, y'_j, z'_j\}\)):

\[
\text{If } x'_i = x'_j, \ y'_i = y'_j \\
\text{then } \text{this circle belongs to the group with respect to this part axis.}
\]

3.3 Module II: Identify surfaces of workpiece image

This module performs a search and identification of all surfaces of a part drawn. Figure 3.3 shows a functional flow chart of the second module.
FIGURE 3.3 FLOW CHART FOR MODULE II
To achieve the identification of all surfaces, the algorithm is developed as follows:

*Step 1:* Access the newly formed database.

*Step 2:* Select an entity from a certain group, such as a line, a circle, etc.

*Step 3:* Move the selected entity by increasing its coordinate along one axis direction chosen.

*Step 4:* Examine the trace of the movement and identify surfaces based on sets of rules. Then give it a label and write it into a file.

*Step 5:* Remove the checked entities from the database.

*Step 6:* Check whether all entities are selected and examined.

If yes, then go to Module III. Otherwise, go to *Step 1.*

There are two sets of rules for identifying surfaces, one for planar surfaces, and another for cylindrical surfaces. A rule for identifying a straight cylindrical surface is expressed as (suppose the center coordinates and radius of two circles are circle1 \{x1, y1, z1, r1\}, circle2 \{x2, y2, z2, r2\}):

If \(x1=x2\) and \(y1=y2\) and \(z2=z1+c\) and \(r1=r2\), (c is any constant), then the two circles are the two end faces of a straight cylindrical surface.

3.4 Module III: Recognize part features

Based on the identification of the surfaces in Module II, all
features of a given part will be recognized by this Module. It is shown, in Figure 3.4, that the part feature recognition is achieved in two main steps: surface relationship determination and feature recognition. The algorithm is stated as follows:

*Step 1:* Access the database newly obtained by Module II.
*Step 2:* Select surface groups.
*Step 3:* Determine the relationship between planar surfaces and give specific labels.
*Step 4:* Determine the relationships between cylindrical surfaces with the same axis and give specific labels.
*Step 5:* Determine the relationships between cylindrical surfaces with different part axes and give specific labels.
*Step 6:* Are all surfaces checked? If not, go back to Step 2.
*Step 7:* Recognize part features based on surface relations, rules and surfaces equations.
*Step 8:* Are all features recognized? If not, go back to step 7.
*Step 9:* Write the output into files in a desired format.
*Step 10:* End.

Sets of rules for determining relationships and ones for recognizing part features are developed. They are the type of:

\[
\text{IF conditions,} \quad \text{THEN conclusions.}
\]

For example, a rule for determining the relationship between two cylindrical surfaces with the same axis is stated as follows:
suppose two cylindrical surfaces $c_l \{x_l, y_l, z_l, x'_l, y'_l, z'_l, r_l\}$,
c2 \{x'_2, y'_2, z'_2, x'_1, y'_1, z'_1, r_2\}

If \(x'_1 = x'_2, y'_1 = y'_2, z'_1 = z'_2\), then \texttt{c1_is_below_c2}.

If \(x'_2 = x'_1, y'_2 = y'_1, z'_2 = z'_1\), then \texttt{c1_is_above_c2}.

A rule for recognizing a through hole is expressed as (assume that the relationship between surface c1 with length L1 and surface c2 with length L2 is c1 within c2):

If \(L_1=L_2\), then \texttt{c1_is_a_through_hole}.

Note that surface c1 and surface c2 may or may not share the same part axis.

Finally, the output of recognition will be written into files in a certain format.
**FIGURE 3.4 FLOW CHART FOR MODULE III**
CHAPTER 4

IMPLEMENTATION OF INFERS

In this chapter, four examples are given to demonstrate the capabilities of the system. It will be shown that INFERS can recognize form features such as 'through hole', 'countersink hole', 'step', 'taper', etc., and also can recognize complex rotational parts with multiple parallel axes. The parts shown as examples were drawn by using AutoCAD.

4.1 Example 1: A part with through hole

The first part chosen as an example is a rotational part with a through hole, as shown in Figure 4.1.

![Figure 4.1 Through Hole (Example 1)](image)

As soon as a user inputs the file name to be processed, INFERS will perform an intelligent feature recognition. The basic steps of the
feature recognition process executed by INFERS are described below:

Module I -- Data extraction and preprocess:

step 1: Extracting circles: four circles (C1, C2, C3, C4) are extracted.

step 2: Extracting lines: a line is extracted and it represents the part axis.

step 3: Preprocessing circles: the four circles are grouped together since they share the same axis.

step 4: Rearranging the four circles according to their center coordinates along this part axis.

step 5: Write the preprocessed data into a data file--DB1.

Module II -- Surface identification

step 1: Accesses DB1.

step 2: Select an entity, say C1, move it along the part axis.

step 3: Result in C1 matching C3. Based on rules, C1 and C3 are identified as the two end faces of a straight cylindrical surface, labeled as cysurf1.

step 4: Write cysurf1 into new database--DB2, in a format of starting edge, ending edge and L/D ratio, etc., as shown in table 4.1-1.

step 5: Remove C1 and C3 from DB1.

step 6: Repeat step 2 to step 5 until all entities checked.

Module III Feature recognition

step 1: Accesses DB2.

step 2: Determine the relationship between surfaces (cylindrical
<table>
<thead>
<tr>
<th>No.</th>
<th>surface label</th>
<th>starting face(edge)</th>
<th>ending face(edge)</th>
<th>L/D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>8.00 6.00 0.00 1.00</td>
<td>8.00 6.00 4.00 1.00</td>
<td>2.00</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>8.00 6.00 0.00 2.00</td>
<td>8.00 6.00 4.00 2.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Table 4.1-1 Output-Surface Information for Example 1

<table>
<thead>
<tr>
<th>Feature No.</th>
<th>Feature type</th>
<th>Consisting surface ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>external(diameter)</td>
<td>cysurf2</td>
</tr>
<tr>
<td>2</td>
<td>through_hole</td>
<td>cysurf1</td>
</tr>
</tbody>
</table>

Table 4.1-2 Output-Feature Information for Example 1
surfaces cysurf1 and cysurf2 are identified in Module II. cysurf2 is inside of cysurf1. Give a certain label to note this relationship.

step 3: Based on the relationship, further examine the lengths of the two surfaces, they are equal. Based on rules and logic, cysurf2 is recognized as a through hole.

step 4: All features recognized? Yes, go to step 3.

step 5: Write features into output files in a certain format as shown in Table 4.1-2.

4.2 Example 2: A stepped part

The second example is a stepped part as shown in Figure 4.2.

![Figure 4.2 A Stepped Part (Example 2)](image)

The procedures are the same as in example (1) till cysurf1 and cysurf2
are identified. Then, the relationship between cysurf1 and cysurf2 is determined as cysurf1 is on cysurf2 because the ending face of cysurf1 is located at the same point as the starting face of cysurf2 along the axis. Since there is no other surfaces surrounding cysurf1 and cysurf2, and the radius of cysurf1 is bigger than that of cysurf2, then cysurf1 is the external diameter and with cysurf2 forms a step from the right hand side direction as shown in Figure 4.2. Table 4.2-1 and Table 4.2-2 show the outputs in the form of surface information and feature information, respectively.
<table>
<thead>
<tr>
<th>No.</th>
<th>surface label</th>
<th>starting face(edge)</th>
<th>ending face(edge)</th>
<th>L/D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>x1      y1  z1  r1</td>
<td>x2      y2  z2  r2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>5.07  5.02  0.00  2.00</td>
<td>5.07  5.02  2.00  2.00</td>
<td>0.50</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>5.07  5.02  2.00  1.00</td>
<td>5.07  5.02  3.50  1.00</td>
<td>0.75</td>
</tr>
</tbody>
</table>

Table 4.2-1 Output-Surface Information for Example 2

<table>
<thead>
<tr>
<th>Feature No.</th>
<th>Feature type</th>
<th>Consisting surface ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>step</td>
<td>cysurfl and cysurf2</td>
</tr>
</tbody>
</table>

Table 4.2-2 Output-Feature Information for Example 2
The two examples above showed the capabilities of INFERS for recognizing simple centerline symmetric rotational parts. The main objective of INFERS is, however, to recognize rotational parts with multiple parallel axes. The following two examples will show the capability of INFERS for rotational parts with multiple axes.

4.3 Example 3: A complex part I with multiple axes

The complex part I is shown in Figure 4.3.

![Diagram of a complex part I with multiple axes]

FIGURE 4.3 A COMPLEX PART I (EXAMPLE 3)

The feature recognition steps are briefly described as follows: until the end of Module II, the procedures are similar to what was described for the two examples above, and we obtain the results: surfaces information shown in Table 4.3-1 and surface relationships between the surfaces with the same part axis, such as cysurf3 on cysurf4, cysurf5 on cysurf6, cysurf7 on cysurf8 and cysurf2 in cysurf1 (these
relationships are stored in buffer files temporarily, say RELA 1).

The following steps are added:
- Determine the relationships between the surfaces with different part axes, based on rules and surface equations. Obtain relations RELA2.
- Feature recognition, based on RELA1, RELA2, rules and calculations of their lengths and rules.
- Write feature into a file in a desired format. Table 4.3-2 shows the final result in the form of feature information.
<table>
<thead>
<tr>
<th>No.</th>
<th>surface label</th>
<th>starting face(edge)</th>
<th>ending face(edge)</th>
<th>L/D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>7.08 6.00 0.00 3.00</td>
<td>7.08 6.00 3.00 3.00</td>
<td>0.50</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>7.08 6.00 0.00 0.80</td>
<td>7.08 6.00 2.00 0.80</td>
<td>1.25</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>7.08 7.80 0.00 0.50</td>
<td>7.08 7.80 1.00 0.50</td>
<td>1.00</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>7.08 7.80 1.00 0.30</td>
<td>7.08 7.80 3.00 0.30</td>
<td>3.33</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>5.27 6.00 0.00 0.50</td>
<td>5.27 6.00 1.00 0.50</td>
<td>1.00</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>5.27 6.00 1.00 0.30</td>
<td>5.27 6.00 3.00 0.30</td>
<td>3.33</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>7.08 4.20 0.00 0.50</td>
<td>7.08 4.20 1.00 0.50</td>
<td>1.00</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>7.08 4.20 1.00 0.30</td>
<td>7.08 4.20 3.00 0.30</td>
<td>3.33</td>
</tr>
</tbody>
</table>

Table 4.3-1 Output-Surface Information for Example 3

<table>
<thead>
<tr>
<th>Feature No.</th>
<th>Feature type</th>
<th>Consisting surface ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>external(diameter)</td>
<td>cysurf1</td>
</tr>
<tr>
<td>2</td>
<td>blind_hole</td>
<td>cysurf2</td>
</tr>
<tr>
<td>3</td>
<td>countersink hole</td>
<td>cysurf3 and cysurf4</td>
</tr>
<tr>
<td>4</td>
<td>countersink hole</td>
<td>cysurf5 and cysurf6</td>
</tr>
<tr>
<td>5</td>
<td>countersink hole</td>
<td>cysurf7 and cysurf8</td>
</tr>
</tbody>
</table>

Table 4.3-2 Output-Feature Information for Example 3
4.4 Example 4: A complex part II with multiple axes

This part is shown in Figure 4.4. Though not an actual machine component, it includes five axes and several commonly found shapes, such as steps, holes and tapered surfaces. Thus the profile of the part tests the capabilities of the system. The final outputs of INFERS are given in Table 4.4-1 and 4.4-2.
<table>
<thead>
<tr>
<th>No.</th>
<th>surface label</th>
<th>starting face(edge)</th>
<th>ending face(edge)</th>
<th>L/D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>x₁ y₁ z₁ r₁</td>
<td>x₂ y₂ z₂ r₂</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>200</td>
<td>5.00 5.00 0.00 0.70</td>
<td>5.00 5.00 0.70 1.00</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>5.00 5.00 0.70 1.00</td>
<td>5.00 5.00 3.00 1.00</td>
<td>1.15</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>5.00 5.00 3.00 2.50</td>
<td>5.00 5.00 4.20 2.50</td>
<td>0.24</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>5.00 6.60 3.00 0.50</td>
<td>5.00 6.60 3.50 0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>5.00 6.60 3.50 0.30</td>
<td>5.00 6.60 4.20 0.30</td>
<td>1.17</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>5.00 3.40 3.00 0.50</td>
<td>5.00 3.40 3.50 0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>5.00 3.40 3.50 0.30</td>
<td>5.00 3.40 4.20 0.30</td>
<td>1.17</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>6.60 5.00 3.00 0.50</td>
<td>6.60 5.00 3.50 0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>6.60 5.00 3.50 0.30</td>
<td>6.60 5.00 4.20 0.30</td>
<td>1.17</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>3.40 5.00 3.00 0.50</td>
<td>3.40 5.00 3.50 0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>3.40 5.00 3.50 0.30</td>
<td>3.40 5.00 4.20 0.30</td>
<td>1.17</td>
</tr>
</tbody>
</table>

Table 4.4-1 Output-Surface Information for Example 4
<table>
<thead>
<tr>
<th>Feature No.</th>
<th>Feature type</th>
<th>Consisting surface ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>tapered surface</td>
<td>cysurf1</td>
</tr>
<tr>
<td>2</td>
<td>external (diameter)</td>
<td>cysurf3</td>
</tr>
<tr>
<td>3</td>
<td>step</td>
<td>cysurf2 and cysurf3</td>
</tr>
<tr>
<td>4</td>
<td>countersink hole</td>
<td>cysurf4 and cysurf5</td>
</tr>
<tr>
<td>5</td>
<td>countersink hole</td>
<td>cysurf6 and cysurf7</td>
</tr>
<tr>
<td>6</td>
<td>countersink hole</td>
<td>cysurf8 and cysurf9</td>
</tr>
<tr>
<td>7</td>
<td>countersink hole</td>
<td>cysurf10 and cysurf11</td>
</tr>
</tbody>
</table>

Table 4.4-2  Output-Feature Information for Example 4
CHAPTER 5

EVALUATION AND PROSPECTIVE APPLICATIONS OF INFERS

5.1 Evaluation

In comparison with previous systems such as the ones developed by Wang and Wysk [1988], Li [1988], Sahay et al. [1990], etc., INFERS has one significant improvement, that is, INFERS is capable of recognizing rotational parts with multiple parallel axes, which offers more practical applications. It should be noticed that, through the four examples, INFERS is also valid for center line symmetric rotational parts. Further, INFERS has the following two properties:

- interpretation is eliminated (fully automated) during the whole process of feature recognition;
- independent of CAD systems used for drawing parts except data extraction from CAD data base.

The knowledge bases (rules and logic) are developed according to part representations scheme (the simplified BREP) and their corresponding equations as well as coordinates. It should be pointed out that the algorithms developed potentially suit rotational parts with key ways although it is beyond the main objectives of INFERS.

INFERS is synthesized in a modular and hierarchical form. It is flexible and easy to be modified and enriched because of the decomposition of the methodology into sets of well-defined functions. For instance, at present, geometric data is read from .DXF files. To
read an IGES file, or an output file from a CAD system, only one routing needs to be added. An assembly part drawing in three dimensional space is required. INFERS currently supports rotational parts with multiple parallel axes consisting of cylindrical surfaces.

5.2 Prospective Applications

Surface and feature information about parts are indispensable for many manufacturing applications, such as process planning, classifying and coding parts, creating NC tape images, etc., while INFERS just provides those surface and feature information. All of the information is stored in files which are easily accessed; this implies that INFERS could be easily interfaced. Figure 5 shows a diagram of INFERS interfaced with CAD systems and process planning systems. Actually, in a sense of data extraction and feature recognition, INFERS is an ideal preprocessor to other manufacturing applications.

5.2.1 Generating Process Plans

Process planning is usually defined as the systematic determination of the methods by which a product can be manufactured economically and competitively [Chang and Wysk 1985]. It bridges the product design and the product fabrication functions [Wang and Wysk 1988]. A detailed plan usually contains the route, processes, process parameters, and machine and tool selection. The part features is basic information required for process planning. Other information such as tolerance, surface finish, material, quantity and the manufacturing system itself also contribute to the selection of operations and their sequencing in a process plan.
FIGURE 5 Interfacing INFERS with CAD and CAPP
For the generation of process plans, human interpretation of design drawings is necessary if without an intelligent data extraction and feature recognition system. The INFERS is an ideal preprocessor to a fully automated process planning system in which rotational parts with multiple parallel axes are produced.

5.2.2 Classifying and Coding Parts for GT

Group technology is a manufacturing philosophy in which similar parts are identified and grouped together to take advantage of their similarities in manufacturing and design. The basis of most group technology applications is a classification and coding system. The proposed methodology -- INFERS may be used in conjunction with such systems or it can be extended for part coding.

5.2.3 Creating NC Tape Images

In order to generate NC tape images, geometric data of a given part have to be extracted and identified. Geometric data extraction and identification is one of the functions of INFERS.

5.2.4 Design for Assembly Analysis

Design for assembly analysis procedures requires certain geometric properties for each component part, while INFERS can easily extract geometric information from CAD data base. INFERS may be used for integrating CAD and design for assembly analysis.
CHAPTER 6

CONCLUSIONS AND SUGGESTIONS FOR FUTURE RESEARCH

In this research, a part feature recognition system named INFERS has been developed and described in the previous chapters. The contributions of the research are given in the next section followed by a brief discussion on the suggestions for future research.

6.1 Contributions of the Research

In this work, an attempt was made to fill the existing gap between CAD and CAM. A part-feature recognition system named INFERS has been developed for rotational parts. INFERS is capable of recognizing rotational parts with multiple parallel axes, which is a significant improvement over previous systems for rotational parts. INFERS has many prospective applications as mentioned in the previous chapter. By using INFERS, human interpretation of design information from a CAD database is eliminated. INFERS is independent of a CAD system used for drawing parts except data extraction. It is a valuable link between CAD and CAPP.

6.2 Suggestions for Future research

Translating the CAD language to the language of a CAPP system is one of the major challenges of the integration of CAD and CAM (Gupta 1990). Because of the diverse techniques being used in feature recognition systems and the rapid development of computer aided technology, it may not be easy to predict the future research
directions. However, according to our survey of part feature recognition systems, the following issues need further efforts.

6.2.1 Interface with CAD/Drafting Systems
Most of the known feature recognition systems are dependent on CAD systems used for drawing objects which limit the application of the feature recognition system. A powerful feature recognition system, as an important link between the CAD system and APP or CAPP systems, should be able to be integrated with any CAD systems which are used to draw a work piece.

6.2.2 Generic Feature Recognition System
Most of the known feature recognition systems are limited to simple features or a limited number of form features. Certain feature recognition systems recognize specific part features or part families. Some systems may conceptually recognize any parts but their full scale involvement in application is yet to be seen. Therefore, a general feature recognition system is desirable.

6.2.3 Part Attributes Recognition System
From a practical point of view, a part feature recognition system should recognize automatically not only dimensional geometric and topological attributes of a part, but also all other attributes such as tolerance, surface finish, etc. This information is important for process planning. Few known feature recognition systems can automatically recognize/extract tolerance information about a part. This task is still performed through human interaction, which seems to block the fully integrated CAD and APP or CAPP to come true.
REFERENCES


30. Lee, Y. C. Integration of solid modeling and Database Management for CAD/CAM, PhD. Dissertation, Purdue Univ. USA, 1984


36. Liu, D., Utilization of Artificial Intelligence In Manufacturing,


59. Woo, T. C., Interfacing solid modeling to CAD and CAM data structures and algorithms for decomposing a solid, *Computer, Vol.17 No.12*, pp 44-49, 1984


APPENDIX
#include<stdio.h>
#include<string.h>

char filename[15];
char m[15], mcode[15], cont[15], cent[15], s[15], s1[15], s2[15];
char buff[15], data[20];
/* mcode[], cont[] and cent[] used for distinguish line type*/
char line[15], pline[15], circle[15], arcs[15];
char s_line[15], s_pline[15], c_vertex[15], s_circle[15];
char s_arcs[15], s_end[15];
char buffer[15];
char x_co[15], y_co[15], z_co[15];

FILE *crl1, *crl2, *crl3, *crl4, *crl5;
FILE *x_coor, *y_coor, *z_coor;
/* hl: line group which is perpendicular to yz plane. */
/* vl: line group which is perpendicular to xz plane. */
/* dl: line group which is perpendicular to xy plane. */
/* oxyl: oblique line group which is parallel to xy plane. */
/* oyzl: oblique line group which is parallel to yz plane. */
/* oxzl: oblique line group which is parallel to xz plane. */
/* cl : center line group. */

char hline[15], vline[15], dline[15], oxyline[15];
char oyzline[15], oxzline[15], cline[15];
char circ1[15], circ2[15], circ3[15], circ4[15], circ5[15];
int ncl=0, nc2=0, nc3=0, nc4=0, nc5=0;
/* No. of circles in each group. Grouped in center line(axis). */
int nhl=0, nvl=0, ndl=0, noxyl=0, noyzl=0, noxzl=0, ncl=0;
/* n--l: No. of lines in the corresponding line group */
int flag1=0, flag2=0, flag=10, contflag=0, centflag=0;
int lineflag=10, plineflag=10, pflag=10, vertexflag=10, circleflag=10;
```c
int arcflag=10, endflag=10;
int n=1, nl=0, nc=0, na=0;  /* nl: No. of lines extracted */
int i, j, k;               /* nc: No. of circles extracted */
                          /* na: No. of arcs extracted */
int nxco=0, nyco=0, nzco=0; /* nxco: No. of different x coordinates. */

float x[60], y[60], z[60], r[60], x2[60], y2[60], z2[60], r2[60], ratio[60];
float buf[60], buf2[60], buf3[60], br[60];
float minx, maxx, miny, maxy, minz, maxz, minbuf, bufm, bufy, bufz, bufr;
char cysurf[15], id_string[15];
FILE *cysur;
int cysf=0;
int no=0, code=0;
int id_code[60], number1[60], number2[60];
/* these three ints are used for part feature recog. */
FILE *relation; /* write down the relationships between surfaces */
char rela[15];
float len1, len2, length1[50], length2[50];
int re_code;    /* for relationship of surfaces */
int nrela=0;
int ceng=0, geshu=0;
int zhong;
char feat[15];
FILE *fea;
float maxr, maxlen;
int count[40];
int p=0, q=0, l=0, ll=0;
char recog [40];
float chang;
FILE *sout, *fout;   /* for surface output and feature output */
char soutput[15], foutput[15];
float u[15], u2[15], v[15], v2[15];

main()
{
    char output[15];
    FILE *out;
```
/****** MODULE I -- Data extraction and preprocess *******/

/***** 1. Extract entities from .DXF file *******/

printf("DXF file name: ");
scanf("%s", &filename[0]);
strcat(filename,".DXF");
printf("%s", filename);
if ((dxf=fopen(filename,"rt")) == NULL )
{
    printf(" Can't find file '%s'", filename);
    exit(1);
}

/**** Skip unless LAYER ****/
strcpy(s1,"0""");
strcpy(s2,"LAYER""");
strcpy(buff,"""");

layers:
flag1=0;
while(flag1==0)
{
    strcpy(m,"""");
    read_line(m,dxf);
    strncpy(buf,m,2);
    flag1=equal_strings(buf,s1);
}
strcpy(s,"""");
read_line(s,dxf);
flag1=equal_strings(buf,s1);
flag2=equal_strings(s,s2);
if (flag1==1 && flag2==1)
printf("s=%s",s);
else  
goto layers;

strcpy(s1,"CONTINUOUS");
strcpy(s2,"CENTER***");
lay:
strcpy(mcode,"********");
strcpy(s,"********");
read_line(m,dxf);
read_line(mcode,dxf);
for(i=1;i<=5;++i)
read_line(m,dxf);
read_line(s,dxf);
flag1=equal_strings(s,s1);
if(flag1==1) strcpy(cont,mcode);
flag2=equal_strings(s,s2);
if(flag2==1) strcpy(cont,mcode);
read_line(m,dxf);
strcpy(s,"********");
read_line(s,dxf);
flag1=equal_strings(s,"LAYER**");
if(flag1==1) goto lay;

/**** Skip unless ENTITIES section ****/

strcpy(s1,"2*******");
strcpy(s2,"ENTITIES");
strcpy(buff,"*******");

entities:
flag1=0;
while(flag1 == 0 )
{
  strcpy(m,"*******");
read_line(m, dxf);
strncpy(buff, m, 2);
flag1=equal_strings(buff, s1);
}
strcpy(s, "********");
read_line(s, dxf);
flag1=equal_strings(buff, s1);
flag2=equal_strings(s, s2);
if (flag1==1 && flag2==1)
printf("s=%s", s);
else
goto entities;

/**** Pointer for extracting lines, plines, circles or arcs. ****/

strcpy(s_line, "LINE***");
strcpy(s_pline, "POLYLINE");
strcpy(s_vertex, "VERTEX***");
strcpy(s_circle, "CIRCLE***");
strcpy(s_arcs, "ARC****");
strcpy(s_end, "ENDSEC***");
strcpy(m, "*********");
strcpy(s, "*********");

strcpy(line, "line");
lin=fopen(line, "w+");
strcpy(cline, "cline");
cl=fopen(cline, "w+");
strcpy(circle, "circle");
cir=fopen(circle, "w+");
strcpy(arcs, "arc");
arf=fopen(arcs, "w+");
strcpy(buffer, "buffer");
read_line(m, dxf);
read_line(s, dxf);
endflag=equal_strings(s, s_end);
while( endflag == 0)
{
    lineflag=equal_strings(s, s_line);
    plineflag=equal_strings(s, s_pline);
    circleflag=equal_strings(s, s_circle);
    arcflag=equal_strings(s, s_arcs);
    if(lineflag==1) extract_line();
    if(plineflag==1) extract_pline();
    if(circleflag==1) extract_circle();
    if(arcflag==1) extract_arc();
    endflag=equal_strings(s, s_end);
}
fclose(li);
fclose(cl);
fclose(cir);
fclose(arc);
2. Preprocess the data extracted

Preprocess lines extracted

```
lin=fopen(line,"r");
strcpy(x_co,"x_co");
strcpy(y_co,"y_co");
strcpy(z_co,"z_co");
x_co=fopen(x_co,"w+");
y_co=fopen(y_co,"w+");
z_co=fopen(z_co,"w+");

for(i=1;i<=2*n1;++i)
{
    fscanf(lin,"%f%f%f",&x[i],&y[i],&z[i]);
}
for(i=1;i<=2*n1;++i)
    buf[i]=x[i];
pickup_coor();
printf(x_co," %f",buf[1]);
nxco=nxco+1;
for(i=2;i<=2*n1;++i)
    if(buf[i] != buf[i-1]){
        printf(x_co," %f",buf[i]);
        nxco=nxco+1;}

for (i=1;i<=2*n1;++i)
    buf[i]=y[i];
pickup_coor();
printf(y_co," %f",buf[1]);
nyc=nyco+1;
for(i=2;i<=2*n1;++i)
    if(buf[i] != buf[i-1]){
        printf(y_co," %f",buf[i]);
        nyco=nyco+1;}
for(i=1;i<=2*n1;++i)
```
buf[1]=z[1];
pickup_coor();
fprintf(z_coor," %f",buf[1]);
nzco=nzco+1;
for(i=2;i<=2*n1;++i)
  if(buf[1] != buf[i-1]) {
    fprintf(z_coor," %f",buf[1]);
    nzco=nzco+1;
  }
fclose(1in);

strcpy(hline,"hline");
strcpy(vline,"vline");
strcpy(dline,"dline");
strcpy(oxyline,"oxyline");
strcpy(oyzline,"oyzline");
strcpy(oxzline,"oxzline");
hl=fopen(hline,"w");
v1=fopen(vline,"w");
d1=fopen(dline,"w");
oxyl=fopen(oxyline,"w");
oyz1=fopen(oyzline,"w");
oxz1=fopen(oxzline,"w");
lin =fopen(line,"r");
for(i=1;i<=n1;++i) {
  fscanf(1in,"%f%f%f%f%f%f",&x[1],&y[1],&z[1],&x2[1],&y2[1],&z2[1]);

  u[1]=x[1]-x2[1];
  u[2]=y[1]-y2[1];
  u[3]=z[1]-z2[1];
  v[1]=u[1]<0? (-1)*u[1]:u[1];

  if(v[1]>0.001 && v[2]<=-0.001 && v[3]<=0.001) {
    if(x[1]>x2[1]) {
      buf[1]=x2[1];
      x2[1]=x[1];
    }
x[i]=buf[1];
fprintf(hl,"%f%f%f%f%f%f",x[i],y[i],z[i],x2[i],y2[i],z2[i]);
hl=nhl+1; }

if(v[1]<=0.001 && v[2]>0.001 && v[3]<=0.001) {
  if(y[i]>y2[i]) {
    buf[1]=y2[i];
y2[i]=y[i];
y[i]=buf[1];
    fprintf(v1,"%f%f%f%f%f",x[i],y[i],z[i],x2[i],y2[i],z2[i]);
    nv1=nv1+1; }

if(v[1]<=0.001 && v[2]<=0.001 && v[3]>0.001) {
  if(z[i]>z2[i]) {
    buf[1]=z2[i];
z2[i]=z[i];
z[i]=buf[1];
    fprintf(dl,"%f%f%f%f%f",x[i],y[i],z[i],x2[i],y2[i],z2[i]);
    ndl=ndl+1; }

if(v[1]>0.001 && v[2]>0.001 && v[3]<=0.001) {
    fprintf(oxy1,"%f%f%f%f%f%f",x[i],y[i],z[i],x2[i],y2[i],z2[i]);
oxyl=noxyl+1; }

if(v[1]<=0.001 && v[2]>0.001 && v[3]>0.001) {
    fprintf(oyzl,"%f%f%f%f%f%f",x[i],y[i],z[i],x2[i],y2[i],z2[i]);
noyzl=noyzl+1; }

if(v[1]>0.001 && v[2]<=0.001 && v[3]>0.001) {
    fprintf(oxzl,"%f%f%f%f%f%f",x[i],y[i],z[i],x2[i],y2[i],z2[i]);
noxzl=noxzl+1; }
}

fclose(hl);
fclose(v1);
fclose(dl);
fclose(oxy1);
fclose(oyzl);
fclose(oxzl);

read_file(hl,hline,nhl); /* re_arrange hline*/
re_order(hl,hline,nhl,x_coor,x_co,nxco,y_coor,y_co,nyco,x,y,z);
read_file(vl,vline,nvl); /*re_arrangeder vline*/
re_order(vl,vline,nvl,y_coor,y_co,nyco,x_coor,x_co,nxco,y,x,z);
read_file(vl,vline,nvl);
vl=fopen(vline,"w");
for(i=1;i<nvl;++i)
fprintf(vl,"%f%f%f%f%f",y[i],x[i],z[i],x2[i],y2[i],z2[i]);
fclose(vl);

read_file(dl,dline,ndl); /* re_arrange dline */
re_order(dl,dline,ndl,z_coor,z_co,nzco,x_coor,x_co,nxco,z,x,y);
read_file(dl,dline,ndl);
dl=fopen(dline,"w");
for (i=1;i<ndl;++i)
fprintf(dl,"%f%f%f%f%f",y[i],z[i],x[i],x2[i],y2[i],z2[i]);
fclose(dl);

/**** Preprocess circles extracted ****/

strcpy(circ1,"circ1");
strcpy(circ2,"circ2");
strcpy(circ3,"circ3");
strcpy(circ4,"circ4");
strcpy(circ5,"circ5");
strcpy(circ6,"circ6");
strcpy(circ7,"circ7");
strcpy(circ8,"circ8");
strcpy(circ9,"circ9");
strcpy(circ10,"circ10");
strcpy(circle,"circle");

if (ncl!= 0) {

}
cl=fopen(cline,"r");
for(i=1;i<=nc1;++i)
    fscanf(cl,"%f%f%f%f%f%f",&x[i],&y[i],&z[i],&x2[i],&y2[i],&z2[i]);
fclose(cl); }
if (nc1==1)
    re_arrcir(cir,circle,nc); 
if (nc1==2) {
    cir=fopen(circle,"r");
    cir1=fopen(circ1,"w");
    cir2=fopen(circ2,"w");
    for(i=1;i<=nc;++i){
        fscanf(cir,"%f%f%f%f",&buf1[i],&buf2[i],&buf3[i],&r[i]);
        u1[i]=buf1[i]-x[1];
        u2[i]=buf2[i]-y[1];
        v1[i]=u1[i]<0? (-1)*u1[i]:u1[i];
        v2[i]=u2[i]<0? (-1)*u2[i]:u2[i];
        if (v1[i]<=0.001 && v2[i]<=0.001) {
            fprintf(cir1,"%f%f%f%f",buf1[i],buf2[i],buf3[i],r[i]);
            nci=nci+1; }
        u1[i]=buf1[i]-x[2];
        u2[i]=buf2[i]-y[2];
        v1[i]=u1[i]<0? (-1)*u1[i]:u1[i];
        v2[i]=u2[i]<0? (-1)*u2[i]:u2[i];
        if (v1[i]<=0.001 && v2[i]<=0.001) {
            fprintf(cir2,"%f%f%f%f",buf1[i],buf2[i],buf3[i],r[i]);
            nc2=nc2+1; }
    }
    fclose(cir1);
    fclose(cir2);
    fclose(cir);
    re_arrcir(cir1,circ1,nc1);
    re_arrcir(cir2,circ2,nc2); }

if (nc1==3) {
    cir=fopen(circle,"r");
    cir1=fopen(circ1,"w");
cir2=fopen(circ2, "w");
cir3=fopen(circ3, "w");
for (i=1; i<=nc; ++i) {
    fscanf(cir, "%f%f%f%f", &buf[1], &buf2[1], &buf3[1], &r[1]);
    u[1]=buf[1]-x[1];
    u[2]=buf2[1]-y[1];
    v[1]=u[1]<0? (-1)*u[1]: u[1];
    if (v[1]<=0.001 && v[2]<=0.001) {
        fprintf(cir1, "%f%f%f%f", buf[1], buf2[1], buf3[1], r[1]);
        nc1=nc1+1;
    }
    u[1]=buf[1]-x[2];
    u[2]=buf2[1]-y[2];
    v[1]=u[1]<0? (-1)*u[1]: u[1];
    if (v[1]<=0.001 && v[2]<=0.001) {
        fprintf(cir2, "%f%f%f%f", buf[1], buf2[1], buf3[1], r[1]);
        nc2=nc2+1;
    }
    u[1]=buf[1]-x[3];
    u[2]=buf2[1]-y[3];
    v[1]=u[1]<0? (-1)*u[1]: u[1];
    if (v[1]<=0.001 && v[2]<=0.001) {
        fprintf(cir3, "%f%f%f%f", buf[1], buf2[1], buf3[1], r[1]);
        nc3=nc3+1;
    }
}
fclose(cir1);
fclose(cir2);
fclose(cir3);
fclose(cir);
re_arrcir(cir1, circ1, nc1);
re_arrcir(cir2, circ2, nc2);
re_arrcir(cir3, circ3, nc3);
if ( ncl == 4 ) {
    cir=fopen(circle,"r");
    cir1=fopen(circ1,"w");
    cir2=fopen(circ2,"w");
    cir3=fopen(circ3,"w");
    if(( cir4=fopen(circ4,"w"))==NULL){
        printf("Can't open circ4");
    }

    for(i=1;i<=nc;++i)
        fscanf(cir,"%f%f%f%f",&buf[i],&buf2[i],&buf3[i],&r[i]);
    for(i=1;i<=nc;++i){
        u[1]=buf[i]-x[1];
        u[2]=buf2[i]-y[1];
        v[1]=u[1]<0? (-1)*u[1]:u[1];
        if (v[1]<=0.001 && v[2]<=0.001) {
            fprintf(cir1,"%f%f%f%f",buf[i],buf2[i],buf3[i],r[i]);
            ncl=ncl+1;
        }
        u[1]=buf[i]-x[2];
        u[2]=buf2[i]-y[2];
        v[1]=u[1]<0? (-1)*u[1]:u[1];
        if (v[1]<=0.001 && v[2]<=0.001) {
            fprintf(cir2,"%f%f%f%f",buf[i],buf2[i],buf3[i],r[i]);
            nc2=nc2+1;
        }
        u[1]=buf[i]-x[3];
        u[2]=buf2[i]-y[3];
        v[1]=u[1]<0? (-1)*u[1]:u[1];
        if (v[1]<=0.001 && v[2]<=0.001) {
            fprintf(cir3,"%f%f%f%f",buf[i],buf2[i],buf3[i],r[i]);
            nc3=nc3+1;
        }
        u[1]=buf[i]-x[4];
        u[2]=buf2[i]-y[4];
        v[1]=u[1]<0? (-1)*u[1]:u[1];
if (v[1]<=0.001 && v[2]<=0.001) {
    fprintf(cir4,"%f%f%f%f", buf[1], buf2[1], buf3[1], r[1]);
    nc4=nc4+1;
}
fclose(cir1);
fclose(cir2);
fclose(cir3);
fclose(cir4);
fclose(cir);
re_arrc(cir1,circ1,nc1);
re_arrc(cir2,circ2,nc2);
re_arrc(cir3,circ3,nc3);
re_arrc(cir4,circ4,nc4);
}

if (nc1==5) {
    cir1=fopen(circ1,"w");
    cir2=fopen(circ2,"w");
    cir3=fopen(circ3,"w");
    cir4=fopen(circ4,"w");
    cir5=fopen(circ5,"w");
    cir=fopen(circle,"r");
    for (i=1;i<nc;++i){
        fscanf(cir,"%f%f%f%f", &buf[1], &buf2[1], &buf3[1], &r[1]);
        u[1]=buf[1]-x[1];
        u[2]=buf2[1]-y[1];
        v[1]=u[1]<0? (-1)*u[1]:u[1];
        if (v[1]<=0.001 && v[2]<=0.001) {
            fprintf(cir1,"%f%f%f%f", buf[1], buf2[1], buf3[1], r[1]);
            nc1=nc1+1;
            u[1]=buf[1]-x[2];
            u[2]=buf2[1]-y[2];
            v[1]=u[1]<0? (-1)*u[1]:u[1];
            if (v[1]<=0.001 && v[2]<=0.001) {

73
fprintf(cir2,"%f%f%f%f", buf[1], buf2[1], buf3[1], r[1]);
nc2=nc2+1;
}
u[1]=buf[1]-x[3];
u[2]=buf2[1]-y[3];
v[1]=u[1]<0? (-1)*u[1]:u[1];
if (v[1]<=0.001 && v[2]<=0.001) {
    fprintf(cir3,"%f%f%f%f", buf[1], buf2[1], buf3[1], r[1]);
    nc3=nc3+1;
}

u[1]=buf[1]-x[4];
u[2]=buf2[1]-y[4];
v[1]=u[1]<0? (-1)*u[1]:u[1];
if (v[1]<=0.001 && v[2]<=0.001) {
    fprintf(cir4,"%f%f%f%f", buf[1], buf2[1], buf3[1], r[1]);
    nc4=nc4+1;
}

u[1]=buf[1]-x[5];
u[2]=buf2[1]-y[5];
v[1]=u[1]<0? (-1)*u[1]:u[1];
if (v[1]<=0.001 && v[2]<=0.001) {
    fprintf(cir5,"%f%f%f%f", buf[1], buf2[1], buf3[1], r[1]);
    nc5=nc5+1;
}
}
fclose(cir1);
fclose(cir2);
fclose(cir3);
fclose(cir4);
fclose(cir5);
fclose(cir);
re_arrcir(cir1, circ1, nc1);
re_arrcir(cir2, circ2, nc2);
re_arrcir(cir3, circ3, nc3);
re_arrcir(cir4, circ4, nc4);
re_arrcir(cir5, circ5, nc5);
}
if (ncl==6) {
    cir1=fopen(circ1,"w");
    cir2=fopen(circ2,"w");
    cir3=fopen(circ3,"w");
    cir4=fopen(circ4,"w");
    cir5=fopen(circ5,"w");
    cir6=fopen(circ6,"w");
    cir=fopen(circle,"r");
    for (i=1; i<=nc; ++i) {
        fscanf(cir,"%f%f%f%f", &buf[i], &buf2[i], &buf3[i], &r[i]);
        u[1]=buf[i]-x[1];
        u[2]=buf2[i]-y[1];
        v[1]=u[1]<0? (-1)*u[1]: u[1];
        if (v[1]<=0.001 && v[2]<=0.001) {
            fprintf(cir1,"%f%f%f%f", buf[i], buf2[i], buf3[i], r[i]);
            ncl=ncl+1;
        }
        u[1]=buf[i]-x[2];
        u[2]=buf2[i]-y[2];
        v[1]=u[1]<0? (-1)*u[1]: u[1];
        if (v[1]<=0.001 && v[2]<=0.001) {
            fprintf(cir2,"%f%f%f%f", buf[i], buf2[i], buf3[i], r[i]);
            nc2=nc2+1;
        }
        u[1]=buf[i]-x[3];
        u[2]=buf2[i]-y[3];
        v[1]=u[1]<0? (-1)*u[1]: u[1];
        if (v[1]<=0.001 && v[2]<=0.001) {
            fprintf(cir3,"%f%f%f%f", buf[i], buf2[i], buf3[i], r[i]);
            nc3=nc3+1;
        }
        u[1]=buf[i]-x[4];
        u[2]=buf2[i]-y[4];
        v[1]=u[1]<0? (-1)*u[1]: u[1];
        if (v[1]<=0.001 && v[2]<=0.001) {
            fprintf(cir",%f%f%f%f", buf[i], buf2[i], buf3[i], r[i]);
            nc4=nc4+1;
        }
        } else {  // do something else
        }
    }
}

75
fprintf(cir4, "%f%f%f%f", buf[1], buf2[1], buf3[1], r[1]);
nc4=nc4+1;
}
u[1]=buf[1]-x[5];
u[2]=buf2[1]-y[5];
v[1]=u[1]<0? (-1)*u[1]:u[1];
if (v[1]<0.001 && v[2]<0.001) {
  fprintf(cir5, "%f%f%f%f", buf[1], buf2[1], buf3[1], r[1]);
  nc5=nc5+1;
}
u[1]=buf[1]-x[6];
u[2]=buf2[1]-y[6];
v[1]=u[1]<0? (-1)*u[1]:u[1];
if (v[1]<0.001 && v[2]<0.001) {
  fprintf(cir6, "%f%f%f%f", buf[1], buf2[1], buf3[1], r[1]);
  nc6=nc6+1;
}
} fclose(cir1);
fclose(cir2);
fclose(cir3);
fclose(cir4);
fclose(cir5);
fclose(cir6);
fclose(cir);
re_arrcdir(cir1, circ1, nc1);
re_arrcdir(cir2, circ2, nc2);
re_arrcdir(cir3, circ3, nc3);
re_arrcdir(cir4, circ4, nc4);
re_arrcdir(cir5, circ5, nc5);
re_arrcdir(cir6, circ6, nc6);
}

if (nc1==7) {
cir1=fopen(circ1, "w");
cir2=fopen(circ2, "w");
cir3=fopen(circ3, "w");
}
cir4=fopen(circ4, "w");
cir5=fopen(circ5, "w");
cir6=fopen(circ6, "w");
cir7=fopen(circ7, "w");
cir=fopen(circle, "r");
for(i=1; i<nc;++i){
    fscanf(cir, "%f%f%f%f", &buf1[i], &buf2[i], &buf3[i], &r[i]);
    u[1]=buf1[i]-x[1];
    u[2]=buf2[i]-y[1];
    v[1]=u[1]<0? (-1)*u[1]:u[1];
    if (v[1]<=0.001 && v[2]<=0.001) {
        fprintf(cir1, "%f%f%f%f", buf1[i], buf2[i], buf3[i], r[i]);
        nc1=nc1+1;
    }
    u[1]=buf1[i]-x[2];
    u[2]=buf2[i]-y[2];
    v[1]=u[1]<0? (-1)*u[1]:u[1];
    if (v[1]<=0.001 && v[2]<=0.001) {
        fprintf(cir2, "%f%f%f%f", buf1[i], buf2[i], buf3[i], r[i]);
        nc2=nc2+1;
    }
    u[1]=buf1[i]-x[3];
    u[2]=buf2[i]-y[3];
    v[1]=u[1]<0? (-1)*u[1]:u[1];
    if (v[1]<=0.001 && v[2]<=0.001) {
        fprintf(cir3, "%f%f%f%f", buf1[i], buf2[i], buf3[i], r[i]);
        nc3=nc3+1;
    }
    u[1]=buf1[i]-x[4];
    u[2]=buf2[i]-y[4];
    v[1]=u[1]<0? (-1)*u[1]:u[1];
    if (v[1]<=0.001 && v[2]<=0.001) {
        fprintf(cir4, "%f%f%f%f", buf1[i], buf2[i], buf3[i], r[i]);
        nc4=nc4+1;
    }
    u[1]=buf1[i]-x[5];
    u[2]=buf2[i]-y[5];
v[1] = u[1] < 0? (-1)*u[1]:u[1];
if (v[1] <= 0.001 && v[2] <= 0.001) {
    fprintf(cir5,"%f%f%f%f", buf[i], buf2[i], buf3[i], r[i]);
    nc5 = nc5 + 1;
} 

u[1] = buf[i] - x[6];
u[2] = buf2[i] - y[6];
v[1] = u[1] < 0? (-1)*u[1]:u[1];
if (v[1] <= 0.001 && v[2] <= 0.001) {
    fprintf(cir6,"%f%f%f%f", buf[i], buf2[i], buf3[i], r[i]);
    nc6 = nc6 + 1;
} 

u[1] = buf[i] - x[7];
u[2] = buf2[i] - y[7];
v[1] = u[1] < 0? (-1)*u[1]:u[1];
if (v[1] <= 0.001 && v[2] <= 0.001) {
    fprintf(cir7,"%f%f%f%f", buf[i], buf2[i], buf3[i], r[i]);
    nc7 = nc7 + 1;
}

close(cir1);
close(cir2);
close(cir3);
close(cir4);
close(cir5);
close(cir6);
close(cir7);
close(cir);
re_arccir(cir1, circ1, nc1);
re_arccir(cir2, circ2, nc2);
re_arccir(cir3, circ3, nc3);
re_arccir(cir4, circ4, nc4);
re_arccir(cir5, circ5, nc5);
re_arccir(cir6, circ6, nc6);
re_arccir(cir7, circ7, nc7);
if (nc1==8) {
  cir1=fopen(circ1,"w");
  cir2=fopen(circ2,"w");
  cir3=fopen(circ3,"w");
  cir4=fopen(circ4,"w");
  cir5=fopen(circ5,"w");
  cir6=fopen(circ6,"w");
  cir7=fopen(circ7,"w");
  cir8=fopen(circ8,"w");
  cir=fopen(circle,"r");
  for(i=1;i<=nc;++i){
    fscanf(cir,"%f%f%f%f", &buf1[i], &buf2[i], &buf3[i], &r[i]);
    u[1]=buf1[i]-x[1];
    u[2]=buf2[i]-y[1];
    v[1]=u[1]<0? (-1)*u[1]:u[1];
    if (v[1]<0.001 && v[2]<0.001) {
      fprintf(cir1,"%f%f%f%f", buf1[i], buf2[i], buf3[i], r[i]);
      nc1=nc1+1;
    }
    u[1]=buf1[i]-x[2];
    u[2]=buf2[i]-y[2];
    v[1]=u[1]<0? (-1)*u[1]:u[1];
    if (v[1]<0.001 && v[2]<0.001) {
      fprintf(cir2,"%f%f%f%f", buf1[i], buf2[i], buf3[i], r[i]);
      nc2=nc2+1;
    }
    u[1]=buf1[i]-x[3];
    u[2]=buf2[i]-y[3];
    v[1]=u[1]<0? (-1)*u[1]:u[1];
    if (v[1]<0.001 && v[2]<0.001) {
      fprintf(cir3,"%f%f%f%f", buf1[i], buf2[i], buf3[i], r[i]);
      nc3=nc3+1;
    }
    u[1]=buf1[i]-x[4];
    u[2]=buf2[i]-y[4];
    v[1]=u[1]<0? (-1)*u[1]:u[1];
  }
}
if (v[1]<=0.001 && v[2]<=0.001) {
    fprintf(cir4,"%f%f%f%f",buf[1],buf2[1],buf3[1],r[1]);
    nc4=nc4+1;
    u[1]=buf[1]-x[5];
    u[2]=buf2[1]-y[5];
    v[1]=u[1]<0? (-1)*u[1]:u[1];
    if (v[1]<=0.001 && v[2]<=0.001) {
        fprintf(cir5,"%f%f%f%f",buf[1],buf2[1],buf3[1],r[1]);
        nc5=nc5+1;
        u[1]=buf[1]-x[6];
        u[2]=buf2[1]-y[6];
        v[1]=u[1]<0? (-1)*u[1]:u[1];
        if (v[1]<=0.001 && v[2]<=0.001) {
            fprintf(cir6,"%f%f%f%f",buf[1],buf2[1],buf3[1],r[1]);
            nc6=nc6+1;
            u[1]=buf[1]-x[7];
            u[2]=buf2[1]-y[7];
            v[1]=u[1]<0? (-1)*u[1]:u[1];
            if (v[1]<=0.001 && v[2]<=0.001) {
                fprintf(cir7,"%f%f%f%f",buf[1],buf2[1],buf3[1],r[1]);
                nc7=nc7+1;
                u[1]=buf[1]-x[8];
                u[2]=buf2[1]-y[8];
                v[1]=u[1]<0? (-1)*u[1]:u[1];
                if (v[1]<=0.001 && v[2]<=0.001) {
                    fprintf(cir8,"%f%f%f%f",buf[1],buf2[1],buf3[1],r[1]);
                    nc8=nc8+1;
                }
            }
        }
    }
}
close(cir1);
close(cir2);
close(cir3);
close(cir4);
fclose(cir5);
fclose(cir6);
fclose(cir7);
fclose(cir8);
fclose(cir);
re_arrcir(cir1,circ1,nc1);
re_arrcir(cir2,circ2,nc2);
re_arrcir(cir3,circ3,nc3);
re_arrcir(cir4,circ4,nc4);
re_arrcir(cir5,circ5,nc5);
re_arrcir(cir6,circ6,nc6);
re_arrcir(cir7,circ7,nc7);
re_arrcir(cir8,circ8,nc8);
}

if (nc1==9) {
  cir1=fopen(circ1,"w");
  cir2=fopen(circ2,"w");
  cir3=fopen(circ3,"w");
  cir4=fopen(circ4,"w");
  cir5=fopen(circ5,"w");
  cir6=fopen(circ6,"w");
  cir7=fopen(circ7,"w");
  cir8=fopen(circ8,"w");
  cir9=fopen(circ9,"w");
  cir=fopen(circle1,"r");
  for(i=1;i<nc;++i){
    fscanf(cir,"%f%f%f%f",&buf[i],&buf2[i],&buf3[i],&r[i]);
    u[1]=buf[1]-x[1];
    u[2]=buf2[1]-y[1];
    v[1]=u[1]<0?(-1)*u[1]:u[1];
    if (v[1]<=0.001 && v[2]<=0.001) {
      fprintf(cir1,"%f%f%f%f",buf[1],buf2[1],buf3[1],r[1]);
      nc1=nc1+1;
    }
    u[1]=buf[1]-x[2];
    u[2]=buf2[1]-y[2];
  }
\[ v(1) = u(1) < 0? (-1)*u(1):u(1); \]
\[ v(2) = u(2) < 0? (-1)*u(2):u(2); \]
if \( v(1) <= 0.001 && v(2) <= 0.001 \) {
    fprintf(cmp2, "%.6f%6.6f", buf(1), buf(1), buf(1), r(1));
    nc2 = nc2 + 1;
} \]
\[ u(1) = buf(1) - x[3]; \]
\[ u(2) = buf2[1] - y[3]; \]
\[ v(1) = u(1) < 0? (-1)*u(1):u(1); \]
\[ v(2) = u(2) < 0? (-1)*u(2):u(2); \]
if \( v(1) <= 0.001 && v(2) <= 0.001 \) {
    fprintf(cmp3, "%.6f%6.6f", buf(1), buf(2), buf(1), r(1));
    nc3 = nc3 + 1;
} \]
\[ u(1) = buf(1) - x[4]; \]
\[ u(2) = buf2[1] - y[4]; \]
\[ v(1) = u(1) < 0? (-1)*u(1):u(1); \]
\[ v(2) = u(2) < 0? (-1)*u(2):u(2); \]
if \( v(1) <= 0.001 && v(2) <= 0.001 \) {
    fprintf(cmp4, "%.6f%6.6f", buf(1), buf(2), buf(2), r(1));
    nc4 = nc4 + 1;
} \]
\[ u(1) = buf(1) - x[5]; \]
\[ u(2) = buf2[1] - y[5]; \]
\[ v(1) = u(1) < 0? (-1)*u(1):u(1); \]
\[ v(2) = u(2) < 0? (-1)*u(2):u(2); \]
if \( v(1) <= 0.001 && v(2) <= 0.001 \) {
    fprintf(cmp5, "%.6f%6.6f", buf(1), buf(2), buf(3), r(1));
    nc5 = nc5 + 1;
} \]
\[ u(1) = buf(1) - x[6]; \]
\[ u(2) = buf2[1] - y[6]; \]
\[ v(1) = u(1) < 0? (-1)*u(1):u(1); \]
\[ v(2) = u(2) < 0? (-1)*u(2):u(2); \]
if \( v(1) <= 0.001 && v(2) <= 0.001 \) {
    fprintf(cmp6, "%.6f%6.6f", buf(1), buf(2), buf(3), r(1));
    nc6 = nc6 + 1;
} \]
\[ u(1) = buf(1) - x[7]; \]
\[ u(2) = buf2[1] - y[7]; \]
\[ v(1) = u(1) < 0? (-1)*u(1):u(1); \]
\[ v(2) = u(2) < 0? (-1)*u(2):u(2); \]
if (v[1]<=0.001 && v[2]<=0.001) {
    fprintf(cir7, "%f%f%f%f", buf[1], buf2[i], buf3[i], r[i]);
    nc7=nc7+1;
}u[1]=buf[1]-x[8];
u[2]=buf2[i]-y[8];
v[1]=u[1]<0? (-1)*u[1]:u[1];
if (v[1]<=0.001 && v[2]<=0.001) {
    fprintf(cir8, "%f%f%f%f", buf[1], buf2[i], buf3[i], r[i]);
    nc8=nc8+1;
}u[1]=buf[1]-x[9];
u[2]=buf2[i]-y[9];
v[1]=u[1]<0? (-1)*u[1]:u[1];
if (v[1]<=0.001 && v[2]<=0.001) {
    fprintf(cir9, "%f%f%f%f", buf[1], buf2[i], buf3[i], r[i]);
    nc9=nc9+1;
}
fclose(cir1);
fclose(cir2);
fclose(cir3);
fclose(cir4);
fclose(cir5);
fclose(cir6);
fclose(cir7);
fclose(cir8);
fclose(cir9);
fclose(cir);
re_arrcir(cir1, cir1, nc1);
re_arrcir(cir2, cir2, nc2);
re_arrcir(cir3, cir3, nc3);
re_arrcir(cir4, cir4, nc4);
re_arrcir(cir5, cir5, nc5);
re_arrcir(cir6, cir6, nc6);
re_arrcir(cir7, cir7, nc7);
re_arrcir(cir8, cir8, nc8);
re_arrcir(cir9, cir9, nc9);
if (nc1==10) {
    cir1=fopen(circ1,"w");
    cir2=fopen(circ2,"w");
    cir3=fopen(circ3,"w");
    cir4=fopen(circ4,"w");
    cir5=fopen(circ5,"w");
    cir6=fopen(circ6,"w");
    cir7=fopen(circ7,"w");
    cir8=fopen(circ8,"w");
    cir9=fopen(circ9,"w");
    cir10=fopen(circ10,"w");
    cir=fopen(circle,"r");
    for(i=1;i<=nc;++i){
        fscanf(cir,"%f%f%f%f",&buf[1],&buf2[1],&buf3[1],&r[1]);
        u[1]=buf[1]-x[1];
        u[2]=buf2[1]-y[1];
        v[1]=u[1]<0? (-1)*u[1]:u[1];
        if (v[1]<=0.001 && v[2]<=0.001) {
            fprintf(cir1,"%f%f%f%f",buf[1],buf2[1],buf3[1],r[1]);
            nc1=nc1+1;
        } u[1]=buf[1]-x[2];
        u[2]=buf2[1]-y[2];
        v[1]=u[1]<0? (-1)*u[1]:u[1];
        if (v[1]<=0.001 && v[2]<=0.001) {
            fprintf(cir2,"%f%f%f%f",buf[1],buf2[1],buf3[1],r[1]);
            nc2=nc2+1;
        } u[1]=buf[1]-x[3];
        u[2]=buf2[1]-y[3];
        v[1]=u[1]<0? (-1)*u[1]:u[1];
        if (v[1]<=0.001 && v[2]<=0.001) {
            fprintf(cir3,"%f%f%f%f",buf[1],buf2[1],buf3[1],r[1]);
nc3=nc3+1;

u[1]=buf[1]-x[4];
u[2]=buf2[1]-y[4];
v[1]=u[1]<0? (-1)*u[1]:u[1];
if (v[1]<=0.001 & & v[2]<=0.001) {
    fprintf(cir4, "%f%f%f%f", buf1, buf2[1], buf3[1], r[1]);
    nc4=nc4+1;
}

u[1]=buf[1]-x[5];
u[2]=buf2[1]-y[5];
v[1]=u[1]<0? (-1)*u[1]:u[1];
if (v[1]<=0.001 & & v[2]<=0.001) {
    fprintf(cir5, "%f%f%f%f", buf1, buf2[1], buf3[1], r[1]);
    nc5=nc5+1;
}

u[1]=buf[1]-x[6];
u[2]=buf2[1]-y[6];
v[1]=u[1]<0? (-1)*u[1]:u[1];
if (v[1]<=0.001 & & v[2]<=0.001) {
    fprintf(cir6, "%f%f%f%f", buf1, buf2[1], buf3[1], r[1]);
    nc6=nc6+1;
}

u[1]=buf[1]-x[7];
u[2]=buf2[1]-y[7];
v[1]=u[1]<0? (-1)*u[1]:u[1];
if (v[1]<=0.001 & & v[2]<=0.001) {
    fprintf(cir7, "%f%f%f%f", buf1, buf2[1], buf3[1], r[1]);
    nc7=nc7+1;
}

u[1]=buf[1]-x[8];
u[2]=buf2[1]-y[8];
v[1]=u[1]<0? (-1)*u[1]:u[1];
if (v[1]<=0.001 & & v[2]<=0.001) {
    fprintf(cir8, "%f%f%f%f", buf1, buf2[1], buf3[1], r[1]);
    nc8=nc8+1;
}

u[1]=buf[1]-x[9];
u[2]=buf2[i]-y[9];
v[1]<0? (-1)*u[1]:u[1];
v[2]<0? (-1)*u[2]:u[2];
if (v[1]<=0.001 & v[2]<=0.001) {
    fprintf(cir9,"%f%f%f%f",buf[i],buf2[i],buf3[i],r[i]);
    nc9=nc9+1;}
u[1]=buf[i]-x[10];
u[2]=buf2[i]-y[10];
v[1]=u[1]<0? (-1)*u[1]:u[1];
if (v[1]<=0.001 & v[2]<=0.001) {
    fprintf(cir10,"%f%f%f%f",buf[i],buf2[i],buf3[i],r[i]);
    nc10=nc10+1;}
}
fclose(cir1);
fclose(cir2);
fclose(cir3);
fclose(cir4);
fclose(cir5);
fclose(cir6);
fclose(cir7);
fclose(cir8);
fclose(cir9);
fclose(cir10);
fclose(cir);
re_arrcir(cir1,circ1,nc1);
re_arrcir(cir2,circ2,nc2);
re_arrcir(cir3,circ3,nc3);
re_arrcir(cir4,circ4,nc4);
re_arrcir(cir5,circ5,nc5);
re_arrcir(cir6,circ6,nc6);
re_arrcir(cir7,circ7,nc7);
re_arrcir(cir8,circ8,nc8);
re_arrcir(cir9,circ9,nc9);
re_arrcir(cir10,circ10,nc10);
/******* MODULE II -- Surfaces identification.*******

if (nc! = 0) {
    strcpy(cysurf,"cysurf");
cysur=fopen(cysurf,"w");}

if (nc1! = 1) id_cy_surf(cir,circle,nc);
if (nc1! = 0) id_cy_surf(cir1,circ1,nc1);
if (nc2! = 0) id_cy_surf(cir2,circ2,nc2);
if (nc3! = 0) id_cy_surf(cir3,circ3,nc3);
if (nc4! = 0) id_cy_surf(cir4,circ4,nc4);
if (nc5! = 0) id_cy_surf(cir5,circ5,nc5);
if (nc6! = 0) id_cy_surf(cir6,circ6,nc6);
if (nc7! = 0) id_cy_surf(cir7,circ7,nc7);
if (nc8! = 0) id_cy_surf(cir8,circ8,nc8);
if (nc9! = 0) id_cy_surf(cir9,circ9,nc9);
if (nc10! = 0) id_cy_surf(cir10,circ10,nc10);
fclose(cysur);

/**** Identify surfaces functions.****/

id_cy_surf(fil,filename,nn)
char filename[];
FILE *fil;
int nn;
{
    float xx,yy,zz,rr;

    float ratio;
    int kk=0,k0=0,c=0;
    int bbb=0;
    /*strcpy(number,"No.");*/
    fil=fopen(filename,"r");
    for(j=1;j<=nn;++j)
        fscanf(fil,"%f%f%f%f", &x[j],&y[j],&z[j],&r[j]);
for (i=1; i<=nn; ++i)
{
    printf ("i=%d", i);
    u[3]=z[i]-z[i+1];
    if(v[3]>0.01)
    {
        u[6]=z[i+1]-z[i+2];
        if(v[6]>0.01)
        {
            u[7]=r[i]-r[i+1];
            if(v[7]<0.01)
            {
                strcpy(string,"straight");
                id_code[i]=100;
                /* id_code=100: straight cylindrical surface. */
                ratio=((z[i+1]-z[i])/(2*r[i]));
                cysf=cysf+1;
                fprintf(cysur,"%s%d%s",number,cysf,string);
                fprintf(cysur,"%d",cysf,id_code[i]);
                fprintf(cysur,"%f%f%f%f",x[i],y[i],z[i],r[i]);
                fprintf(cysur,"%f%f%f",x[i+1],y[i+1],z[i+1]);
                fprintf(cysur,"%f%f",r[i+1],ratio);
            }
            if (v[7]>0.01){
                strcpy(string,"tapered");
                id_code[i]=200;
                /* id_code=200: tapered cylindrical surface. */
                ratio=((z[i+1]-z[i])/((r[i+1]-r[i])/2));
                cysf=cysf+1;
                fprintf(cysur,"%s%d%s",number,cysf,string);
                fprintf(cysur,"%d",cysf,id_code[i]);
                fprintf(cysur,"%f%f%f%f",x[i],y[i],z[i],r[i]);
                fprintf(cysur,"%f%f%f",x[i+1],y[i+1],z[i+1]);
                i=i-1;  /* for picking up circle[i+1] next time. */
            }
        }
    }
}
if(v[6]<=0.01)
{
    strcpy(string,"straight");
    id_code[i]=100;
    if (v[7]<=0.01)
    {
        ratio=((z[i+1]-z[i])/(2*r[i+1]));
        cysf=cysf+1;
        fprintf(cysur,"%s%d%s", number, cysf, string);
        fprintf(cysur,"%d%d", cysf, id_code[i]);
        fprintf(cysur,"%f%f%f", x[i], y[i], z[i], r[i]);
        fprintf(cysur,"%f%f%f", x[i+1], y[i+1], z[i+1]);
        fprintf(cysur,"%f%f", r[i+1], ratio);
    }
}

u[8]=r[1]-r[1+2];

v[8]=u[8]<0? (-1)*u[8]: u[8];

if(v[8]<=0.01){
    ratio=((z[i+1]-z[i])/(2*r[i+2]));
    cysf=cysf+1;
    fprintf(cysur,"%s%d%s", number, cysf, string);
    fprintf(cysur,"%d%d", cysf, id_code[i]);
    fprintf(cysur,"%f%f%f", x[i], y[i], z[i], r[i]);
    fprintf(cysur,"%f%f%f", x[i+2], y[i+2], z[i+2]);
    fprintf(cysur,"%f%f", r[i+2], ratio);
    xx=x[i+1];
    yy=y[i+1];
    zz=z[i+1];
    rr=r[i+1];
    x[i+1]=x[i+2];
    y[i+1]=y[i+2];
    z[i+1]=z[i+2];
    r[i+1]=r[i+2];
    x[i+2]=xx;
    y[i+2]=yy;
    z[i+2]=zz;
    r[i+2]=rr;
}

}

i=i+1;
goto hui; /* for go to next i. */

if(v[3]<0.01)
{
    id_code[i]=100;
k0=2;
kk=k0;
printf ("kk=%d",kk);
for(k=i+1;k<nn;++k){
    if(flag=(z[i]==z[k+1])) kk=kk+1;
    printf ("z[1]=%f",z[i]);
    printf ("z[k+1]=%f",z[k+1]);
}

for(j=i;j<=i+kk-1;++j) {
    xx=x[j];
yy=y[j];
    zz=z[j];
rk=r[j];
    c=kk;
    flag=(c==nn);
    while(flag==0)
    {
        k=c;
u[9]=x-r[i+k];
if(v[9]<0.01){
    ratio=((z[i+k]-zz)/(2*rr));
cysf=cysf+1;
    fprintf(cysur,"%d",cysf, id_code[i]);
    fprintf(cysur,"%f%f%f%f",xx, yy, zz, rr);
    fprintf(cysur,"%f%f%f%f",x[i+k], y[i+k], z[i+k]);
    fprintf(cysur,"%f%f",r[i+k],ratio);
}
    c=c+1;
    flag=(c==nn);
}\}
i=i+kk-1; /* tiao guo kk ge circles*/
}\n\hui:\n\bbb=\bbb+1;
}\n}
/* ******** MODULE III -- Part features recognition. ********/

/***** 1. Determine the relationships of the surfaces. *****/

strcpy(rela,"rela");
if(!(relation=fopen(rela,"w"))==NULL)
  printf("Can't open relfile!");
  printf("cysf=%d",cysf);
if(cysf!=0){
  if(!(cysurf=fopen(cysurf,"r"))=NULL)
    printf("Can't open cysurf file!");
  for (i=1;i<=cysf;++i){
    fscanf(cysur,"%d%d",&number1[i],&id_code[i]);
    fscanf(cysur,"%f%f%f%f",&x[i],&y[i],&z[i],&r[i]);
    fscanf(cysur,"%f%f%f%f",&x2[i],&y2[i],&z2[i],&r2[i],&ratio[i]);
  }
  fclose(cysur);
}
for(i=1;i<=cysf;++i){
  bufm=x2[i];
  bufy=y2[i];
  bufz=z2[i];
  burr=r2[i];
  for (j=i+1;j<=cysf;++j)
    /* These lines below are to determine the ON relation of two cysurf*/
      u[1]=x[j]-bufm;
    u[2]=y[j]-bufy;
    u[3]=z[j]-bufz;
    v[1]=u[1]<0? (-1)*u[1]: u[1];
    if(v[1]<=0.01 && v[2]<=0.01 && v[3]<=0.01){
      len1=(z2[i]-z[1]);
      len2=(z2[j]-z[j]);
      printf("cysurf%d on cysurf%d", number1[i],number1[j]);
if(id_code[i]==100 && id_code[j]==100 && r[i]<r[j])
re_code=110;
if(id_code[i]==100 && id_code[j]==100 && r[i]>r[j])
re_code=120;
fprintf(relation,"%d%rd%rd",number1[i],re_code,number1[j]);
fprintf(relation,"%f%f%f%f",x[i],y[i],z[i],r[i],len1);
fprintf(relation,"%f%f%f%f",x[j],y[j],z[j],r[j],len2);
nrela=nrela+1;
}
}

for(j=i+1;j<=cysf;++j)
/*These lines below are to determine the IN relation*/
{
u[1]=x[j]-bufm;
u[2]=y[j]-bufy;
u[3]=z[j]-bufz;
v[1]=u[1]<0? (-1)*u[1]: u[1];
if(v[1]<=0.01 && v[2]<=0.01 && v[3]<=0.01){
    len1=(z2[i]-z[i]);
    len2=(z2[j]-z[j]);
    if(bufr<r[j]){
        printf ("cysurf%d in cysurf%d",number1[i],number1[j]);
u[10]=len1-len2;
v[10]=u[10]<0? (-1)*u[10]: u[10];
if(v[10]<=0.01) re_code=130;
if(len1<len2) re_code=150;
fprintf(relation,"%d%rd%rd",number1[i],re_code,number1[j]);
fprintf(relation,"%f%f%f%f",x[i],y[i],z[i],r[i],len1);
fprintf(relation,"%f%f%f%f",x[j],y[j],z[j],r[j],len2);
nrela=nrela+1;
}
if(r[j]<bufr){
if(len1==len2) re_code=130;
if(len1 >len2) re_code=150;
printf ("cysurf%d in cysurf%d",number1[j],number1[i]);
fprintf(relation,"%d%d%d", number1[j], re_code, number1[i]);
fprintf(relation,"%f%f%f%f", x[j], y[j], z[j], r[j], len2);
fprintf(relation,"%f%f%f%f", x[i], y[i], z[i], r[i], len1);
nrela=nrela+1;
}
}

bufm=x[i];
bufy=y[i];
bufz=z[i];
bufr=r[i];

for(j=i+1;j<=cysf;++j)
/*These lines below are to determine the IN relation*/
{
  if(x[j]==bufm && y[j]==bufy && z[j]==bufz){
    len1=(z2[i]-z[i]);
    len2=(z2[j]-z[j]);
    if(bufr<r[j]){ printf ("cysurf%dis in cysurf%d", number1[i], number1[j]);
      u[10]=len1-len2;
      v[10]=u[10]<0? (-1)*u[10]:u[10];
      if(v[10]<=0.01) re_code=130;
      if(len1<len2) re_code=140;
      fprintf(relation,"%d%d%d", number1[i], re_code, number1[j]);
      fprintf(relation,"%f%f%f%f", x[i], y[i], z[i], r[i], len1);
      fprintf(relation,"%f%f%f%f", x[j], y[j], z[j], r[j], len2);
      nrela=nrela+1;
    }
    if(r[j]<bufr){
      u[10]=len1-len2;
      v[10]=u[10]<0? (-1)*u[10]:u[10];
      if(v[10]<=0.01) re_code=130;
      if(len1 >len2) re_code=140;
      printf ("cysurf%dis in cysurf%d", number1[j], number1[i]);
      fprintf(relation,"%d%d%d", number1[j], re_code, number1[i]);
      fprintf(relation,"%f%f%f%f", x[j], y[j], z[j], r[j], len2);
      fprintf(relation,"%f%f%f%f", x[i], y[i], z[i], r[i], len1);
      nrela=nrela+1;
    }
  }
}
nrela=nrela+1;
}
}
}

/** the following function will repeat for nc1,nc2,...**/
if(cysf<(2*nrela)){
cysur=fopen(cysurf,"r");
for(i=1;i<=cysf;++i){
 fscanf(cysur,"%d%d",&number1[i],&id_code[i]);
 fscanf(cysur,"%f%f%f%f",&x[i],&y[i],&z[i],&r[i]);
 fscanf(cysur,"%f%f%f%f",&x2[i],&y2[i],&z2[i],&r2[i],&ratio[i]);
}
 fclose(cysur);

if(nc1==0 || nc2 == 0 || nc3==0::nc4==0::nc5==0){
 if(nc1==1) {
  cir=fopen(circle,"r");
  for(i=1;i<=nc;++i)
    fscanf(cir,"%f%f%f%f",&buf[i],&buf2[i],&buf3[i],&br[i]);
  fclose(cir);
  for(j=1;j<=cysf;++j){
    u[1]=x[j]-buf[1];
    u[2]=y[j]-buf2[1];
    v[1]=u[1]<0? (-1)*u[1]: u[1];
    if(v[1]<=0.01 && v2<=0.01){
      len1=(buf3[2]-buf3[1]);
      len2=0.000000;
      number2[j]=0;
      buf[2]=0.000000;
      buf2[2]=0.000000;
      buf3[2]=0.000000;
      br[2]=0.000000;
      printf(relation,"%d%d",number1[j],id_code[j],number2[j]);
      printf(relation,"%f%f%f%f",x[j],y[j],buf3[1],br[1],len1);
      printf(relation,"%f%f",buf2[2],buf3[2]);
      printf(relation,"%f",br[2],len2);
      }
```c
}
}
}
}
fclose(relation);

******/ 2. Part feature recognition. ******/

relation=fopen(rela,"r");
for(i=1;i<=nrela;++i){
    fscanf(relation,"%d%d%d",&number1[i],&id_code[i],&number2[i]);
    fscanf(relation,"%f%f%f%f",&x[i],&y[i],&z[i],&r[i],&length1[i]);
    fscanf(relation,"%f%f%f%f",&x2[i],&y2[i],&z2[i],&r2[i],&length2[i]);
    buf3[i]=z[i];
}
fclose(relation);

/*@ Following rearrange buf3[i], (z coordinates) from small to large*/
for(j=1;j<=nrela;++j)
{
    minz=buf3[j];
    for(i=j+1;i<=nrela;++i)
    {
        if(buf3[i]<minz){
            bufz=minz;
            minz=buf3[i];
        }
    }
    buf[j]=minz;
}

/*@ Following to unique buf[i]*/
ceng=1;
for (j=1;j<=nrela;++j)
{
    buf2[ceng]=buf[j];
    if(buf[i]!=buf[i+1]) ceng=ceng+1;
}
```
/*Part Feature Recognition*/

strcpy(feat,"feat");
fea=fopen(feat,"w");
strcpy(foutput,"foutput");
fout=fopen(foutput,"w");
printf(fout,"Feature No. Feature type Consisting surface ID");
printf(fout,""");
for(k=1;k<=ceng;++k)
{
    geshu=0;
    for(j=1;j<=nrela;++j)
    {
        u[3]=z[j]-buf[k];
        if(v[3]<0.01) { geshu=geshu+1; count[geshu]=j; }
    }
    re_fea();
}
fclose(fea);

/******** Recognize part features function *******/

re_fea()
{
    printf("geshu=%d", geshu);
    l=count[1];
    ll=1;
    if(id_code[1]==130 || id_code[1]==140)
    {
        if(r[1]>r2[1]) { maxr=r[1]; maxlen=length1[1];}
        if(r2[1]>r[1]) { maxr=r2[1]; maxlen=length2[1];}
    }
    if(id_code[1]==120) {
        maxr=r[1];
        maxlen=length1[1];
    }
    printf("maxr=%d maxlen=%d", maxr, maxlen);
}
if(id_code[1]==110) {
    maxr=r2[1];
    maxlen=length2[1];
}

for(p=2; p<=geshu; ++p) {
    l=count[p];
    if(r1[1]>r2[1]) {bufr=r[1]; bufz=length1[1];}
    if(r2[1]>r[1]) {bufr=r2[1]; bufz=length2[1];}
    if(bufr>maxr) {maxr=bufr; maxlen=bufz; ll=1;}
}

j=1; /* ji zhu the No. of features below*/
for(q=1; q<=geshu; ++q) {
    l=count[q];
    if(l==ll)
    {
        strcpy(recog,"external straight cylindrical surface");
        if(id_code[1]==100) {
            fprintf(fea,"cysurf %d is %s", number1[1], recog);
            fprintf(fout,"%-16dexternal(diameter) cysurf%d", j, number1[1]);
            j=j+1;
        }
    }

        if(maxr=r[1])
        {
            fprintf(fea,"cysurf %d is %s", number1[1], recog);
            fprintf(fout,"%-16dexternal(diameter) cysurf%d", j, number1[1]);
            j=j+1;
            if(length2[1]<length1[1]){ 
                fprintf(fea,"cysurf %d is blind hole", number2[1]);
                fprintf(fout,"%-16dblind hole cysurf%d", j, number2[1]);
                j=j+1;
            }
        }
    }
}
u[1]=length2[1]-length1[1];
v[1]=u[1]<0?(-1)*u[1]:u[1];

if(v[1]<=0.01){
    fprintf(fea,"cysurf %d is through hole",number2[1]);
    fprintf(fout,"%-16dthrough_hole cysurf%d",j,number2[1]);
    j=j+1;
}

if(maxr==r2[1])
{
    fprintf(fea,"cysurf %d is %s",number2[1],recog);
    fprintf(fout,"%-16dexternal(diameter) cysurf%d",j,number2[1]);
    j=j+1;
    if(length1[1]<length2[1]){
        fprintf(fea,"cysurf %d is blind hole",number1[1]);
        fprintf(fout,"%-16dbind_hole cysurf%d",j,number1[1]);
        j=j+1;}
    u[1]=length2[1]-length1[1];
v[1]=u[1]<0?(-1)*u[1]:u[1];

    if(v[1]<=0.01){
        fprintf(fea,"cysurf %d is through hole",number1[1]);
        fprintf(fout,"%-16dthrough_hole cysurf%d",j,number1[1]);
        j=j+1;}
}

if(id_code[1]==120) {
    fprintf(fea,"cysurf %d and cysurf %d form a step(right)",
            number1[1],number2[1]);
    fprintf(fout,"%-16d step(right) cysurf%d and cysurf%d",j,
            number1[1],number2[1]);
    j=j+1;}

if(id_code[1]==110) {
    fprintf(fea,"cysurf %d and cysurf %d form a step(left)",
            number1[1],number2[1]);
    fprintf(fout,"%-16d step(left) cysurf%d and cysurf%d",j,
number1[l], number2[l]);
    
    j=j+1;
    }
    }

if(l!=ll)
{
    chang=length1[l]+length2[l];
    if(chang<maxlen) {
        fprintf(fea,"cysurf %d & cysurf %d form a blind counter_sink bore", number1[l], number2[l]);
        fprintf(fout,"%16dblind countersink_bore cysurf%d and cysurf%d", j, number1[l], number2[l]);
        j=j+1;
    }

    if(chang==maxlen ){
        if(id_code[l]==110 :: id_code[l]==120){
            fprintf(fea,"cysurf %d and cysurf %d form a counter_sink hole", number1[l], number2[l]);
            fprintf(fout,"%16dcountersink_bore cysurf%d and cysurf%d", j, number1[l], number2[l]);
            j=j+1;
        }
    }
    }

if(id_code[l]==200) {
    fprintf(fea,"cysurf %d is %s", number1[l], recog);
    fprintf(fout,"%16tapered surface cysurf%d", j, number1[l]);
    j=j+1;
    }
}
/***** write to a output file******/
strcpy(soutput,"soutput");
sout=fopen(soutput,"w");
fprintf(sout,"No. surface starting face(edge) ending face(edge) L/D\n");
fprintf(sout,"label x1 y1 z1 r1 x2 y2 z2 r2\n");
fprintf(sout,"\n");
cysurf=fopen(cysurf,"r");
for(i=1;i<=cysf;++i){
    fscanf(cysurf,"%d",&number1[i],&id_code[i]);
    fscanf(cysurf,"%f\%f\%f\%f",&x[i],&y[i],&z[i],&r[i]);
    fscanf(cysurf,"%f%f%f%f",&x2[i],&y2[i],&z2[i],&r2[i],&ratio[i]);
    fprintf(sout,"%-7d%-10d%5.2f%5.2f%5.2f%5.2f%5.2f%5.2f%5.2f%5.2f\n");
    number1[i],id_code[i],x[i],y[i],
    z[i],r[i],x2[i],y2[i],z2[i],r2[i],ratio[i]);
}
fclose(sout);
fclose(fout);
/***** Function to extract lines ****/

```c
extract_line()
{
    while(lineflag==1)
    {
        /*nl=nl+1;*/
        read_line(m,dxf);
        strcpy(m,"********");
        read_line(m,dxf);
        contflag=equal_strings(m,cont);
        centflag=equal_strings(m,cent);
        for(n=1;n<7;++n)
        {
            strcpy(data," ");
            read_line(m,dxf);
            read_line(data,dxf);
            if(contflag==1)
                fprintf(lin," %s",data);
            if(centflag==1)
                fprintf(cl," %s",data);
        }
        if(contflag==1)
            nl=nl+1;
        if(centflag==1)
            ncl=ncl+1;
        strcpy(s,"********");
        read_line(m,dxf);
        read_line(s,dxf);
        lineflag=equal_strings(s,s_line);
    }
}
```
/**** Function to extract circles ****/

extract_circle()
{
    while(circleflag==1)
    {
        nc=nc+1;
        read_line(m, dxf);
        read_line(m, dxf);
        for(n=1; n<5; ++n)
        {
            strcpy(data, " ");
            read_line(m, dxf);
            read_line(data, dxf);
            fprintf(cir, " %s", data);
        }
        strcpy(s, "********");
        read_line(m, dxf);
        read_line(s, dxf);
        circleflag=equal_strings(s, s_circle);
    }
}
/**** Function to extract pline ****/

extract_pline()
{
    int i=0, j=0;
    for(n=i; n<12; ++n) {
        strcpy(m,"********");
        read_line(m,dxf);
    }
    pflag=equal_strings(m,"0*******");
    if(pflag==1)
    {
        strcpy(s,"********");
        read_line(s,dxf);
        buf_file=fopen(buffer,"w+");
        vertexflag=equal_strings(s,s_vertex);
        while(vertexflag==1)
        {
            i=i+1;
            read_line(m,dxf);
            read_line(m,dxf);
            for (n=1; n<4; ++n)
            {
                strcpy(data,"    ");
                read_line(m,dxf);
                read_line(data,dxf);
                fprintf(buf_file," %s",data);
            }
            strcpy(s,"********");
            read_line(m,dxf);
            read_line(m,dxf);
            vertexflag=equal_strings(s,s_vertex);
        }
        fclose(buf_file);
        buf_file=fopen(buffer,"r");
        for(j=1; j<i; ++j)
{  
    fscanf(buf_fil,"%f%f%f",&x[j],&y[j],&z[j]);
}
fclose(buf_fil);

fprintf(lin,"%f%f%f",x[1],y[1],z[1]);
for (j=2;j<i;++j)
{
    fprintf(lin,"%f%f%f",x[j],y[j],z[j]);
    fprintf(lin,"%f%f%f",x[j],y[j],z[j]);
}
fprintf(lin,"%f%f%f",x[i],y[i],z[i]);
nl=nl+(i-1);
goto yinyeng;
}

read_line(s,dxf);
pflag=equal_strings(s,"SEQEND**");
while(pflag==0)
{
    read_line(s,dxf);
    pflag=equal_strings(s,"SEQEND**");
    strcpy(s,"*******");
}

yinyeng:
strcpy(s,"*******");
for(n=1;n<k;++n)
  read_line(m,dxf);
read_line(s,dxf);
}
/**** Function to extract arc ****/

extract_arc()
{
    while(arcflag==1)
    {
        na=na+1;
        read_line(m,dxf);
        read_line(m,dxf);
        for (n=1;n<5;++n)
        {
            strcpy(data," ");
            read_line(m,dxf);
            read_line(data,dxf);
            fprintf(arc," %s",data);
        }
        strcpy(s,"*******");
        for(n=1;n<5;++n)
            read_line(m,dxf);
        read_line(s,dxf);
        read_line(s,dxf);
        arcflag=equal_strings(s,s_arcs);
    }
}


/**** Function to read_line ****/

read_line(string, fil)
FILE *fil;
char string[];
{
    int i=0;
    char c;
    c='c';
    while(c != '')
    {
        c=fgetc(fil);
        if((c != ' ') && (c != ''))
        {
            string[i]=c;
            i++;
        }
    }
}
/** Function to determine if two strings are equal. **/ 

int equal_strings(str1, str2) 
char str1[ ], str2[ ]; 
{
    int i=0, answer;

    while (str1[i]==str2[i] && str1[i]!='$' && str2[i]!='$')
        ++i;
    if (str1[i] == '$' && str2[i] == '$')
        answer = 1; /* strings equal */
    else
        answer = 0; /* not equal */

    return (answer);
}
/**** Function to pick up different coordinates along an axis ****/

pickup_coor()
{
    for (j=1; j<=2*n1; ++j)
    {
        minbuf=buf[j];
        for (i=j; i<=2*n1; ++i)
        {
            if (buf[i] < minbuf)
            {
                bufm=minbuf;
                minbuf=buf[i];
                buf[i]=bufm;
            }
        }
        buf[j]=minbuf;
    }
}
/**** Function to read file for input. ****/

read_file(file,fileline,nn)
int nn;
char fileline[];
FILE *file;
{
file=fopen(fileline,"r");
for(i=1;i<=nn;++i)
scanf(file,"%f%f%f%f%f",&x[1],&y[1],&z[1],&x2[1],&y2[1],&z2[1]);
fclose(file);
}
/**** Function to reorder coordinates. ****/

int nn, nco1, nco2;
char filename[], co1[], co2[];
float p[], q[], w[];
FILE *file, *coor1, *coor2;
{
  coor1=fopen(co1, "r");
  coor2=fopen(co2, "r");
  file=fopen(filename, "w");
  for (j=1; j<=nco2; ++j)
  {
    fscanf(coor2, "%f", &buf[j]);
    for (k=1; k<=nco1; ++k)
    {
      fscanf(coor1, "%f", &buf2[k]);
      for (i=1; i<=nn; ++i)
      {
        u[1]=p[i]-buf2[k];
        u[2]=q[i]-buf[j];
        v[1]=u[1]<0? (-1)*u[1]: u[1];
        if (v[1]<=0.01 && v[2]<=0.01) {
          fprintf(file, "%f%f%f%f", u[i], v[i], w[i]);
          fprintf(file, "%f%f%f%f", x2[i], y2[i], z2[i]);
        }
      }
    }
  }
  fclose(coor1);
  fclose(coor2);
  fclose(file);
/***** Function to re_arrange circles in a group *****/

re_arrcirs (ffil, fname, nnn)
FILE *ffil;
char fname[];
int nnn;
{
    float xx, yy, zz, rr;
    ffil=fopen(fname, "r");
    for (i=1; i<=nnn; ++i)
        fscanf(ffil,"%f%f%f%f", &buf[i], &buf2[i], &buf3[i], &r[i]);
    fclose(ffil);
    for (j=1; j<=nnn; ++j)
    {
        xx=buf[j];
        yy=buf2[j];
        zz=buf3[j];
        rr=r[j];
        for (k=j+1; k<=nnn; ++k)
        {
            if (buf3[k]<=zz)
            {
                bufm=xx;
                bufy=yy;
                bufz=zz;
                bufr=rr;
                xx=buf[k];
                yy=buf2[k];
                zz=buf3[k];
                rr=r[k];
                buf[k]=bufm;
                buf2[k]=bufy;
                buf3[k]=bufz;
                r[k]=bufr;
            }
        }
    }
}
x[j]=xx;
y[j]=yy;
z[j]=zz;
r[j]=rr;
}
ff11=fopen(flname,"w");
for (j=1;j<=nnn;++j)
 fprintf(ff11,"%f%f%f%f",x[j],y[j],z[j],r[j]);
fclose(ff11);
VITA AUCTORIS

1960    Born in Beijing, P.R. China, on May 11\textsuperscript{th}.

1983    Graduated from Tsinghua University, Beijing, P.R.C.
        with a Bachelor of Engineering in Thermal Engineering.

1983-1989    Worked in Beijing Special Engineering Design and Research
        Institute, Beijing, P.R.C., as a mechanical engineer.

1991    Currently a candidate for the degree of Master of Applied
        Science in Industrial Engineering at the University of
        Windsor, Windsor, Ontario, Canada