DECISION-MAKING ALGORITHMS IN GEOMETRIC MODELLING

P. Cooley, B.Sc.(Eng.) (London)

A thesis submitted for the degree of Ph.D.

The University of Aston in Birmingham

October 1984

Decision-making algorithms in geometric modelling

Peter Cooley

Submitted for the degree of Doctor of Philosophy of the University of Aston in Birmingham 1984

SUMMARY

Many of the applications of geometric modelling are concerned with the computation of well-defined properties of the model. The applications which have received less attention are those which address questions to which there is no unique answer. This thesis describes such an application: the automatic production of a dimensioned engineering drawing. One distinctive feature of this operation is the requirement for sophisticated decision-making algorithms at each stage in the processing of the geometric model. Hence, the thesis is focussed upon the design, development and implementation of such algorithms.

Various techniques for geometric modelling are briefly examined and then details are given of the modelling package that was developed for this project. The principles of orthographic projection and dimensioning are treated and some published work on the theory of dimensioning is examined. A new theoretical approach to dimensioning is presented and discussed.

The existing body of knowledge on decision-making is sampled and the author then shows how methods which were originally developed for management decisions may be adapted to serve the purposes of this project.

The remainder of the thesis is devoted to reports on the development of decision-making algorithms for orthographic view selection, sectioning and cross-hatching, the preparation of orthographic views with essential hidden detail, and two approaches to the actual insertion of dimension lines and text.

The thesis concludes that the theories of decision-making can be applied to work of this kind. It may be possible to generate computer solutions that are closer to the optimum than some man-made dimensioning schemes. Further work on important details is required before a commercially acceptable package could be produced.

Geometric modelling
Decision Theory
Engineering Drawing
Computer-aided design
Computer Graphics

ACKNOWLEDGEMENTS

I wish to thank my supervisor, Mr. T.H. Richards, for his unflagging guidance and support throughout this project.

I would like to record my gratitude to Dr. R. Jeffers and the Faculty of the School of Engineering, University of Connecticut for their hospitality and co-operation during the year which I spent there as Visiting Associate Professor.

Thanks are due to Servelec Computer Systems Ltd. for advice on the operation of the microcomputer used during the latter stages of this project and for the provision of editing and linking software.

I wish to thank all those draughtsmen, engineers and managers who responded to the questions which are detailed in Chapter 9. Their assessments, comments and suggestions were of great assistance in the development of the dimensioning algorithms.

I am most grateful to Professor K. Foster for reading the draft thesis and his comments thereon.

Lastly, I must extend my gratitude to all members of the Academic staff of the former Department of Mechanical Engineering, University of Aston in Birmingham who inevitably carry an increased workload when a colleague is engaged upon a project of this kind.

List of Contents

LIST OF ILLUSTRATIONS

NOMENCLATURE

Symbol	Description	First Reference on Page
a b c	Homogeneous coordinates of line in 2-space	48
d	Displacement	67
f	Parameter for position on line in 2-space	49
i	Number of "clear" surfaces	133
m	Number of combinations	135
p	General parameter	-
r	Relative (dimensionless) distance	125
t	Parameter for perpendicular to line	48
u	Variation in length of strut	67
u_r	Maximum value of rth parameter	76
v_r	Minimum value of rth parameter	76
x y z	Cartesian coordinates of point in 3-space	-
A	Angle of rotation of point around axis	47
B	Angle of rotation of plane around axis	-
C	Number of combinations	133
$C^{(i)}$	Continuity of ith order at Node	194
E	Number of Edges	26
F	Number of surfaces	26

Symbol	Description	First Reference on Page
$F(x)$	Objective function of independent variable(s) x	104
H	Maximum value	–
H	Number of hidden Nodes	157
I	Additional information	100
K	Initial knowledge	100
L	Length of perpendicular	48
L	Minimum value	–
L_{ij}	Length of strut between ith and jth Node	67
M	Class of Model	88
N	Number of Nodes	26
N	State of Nature	85
P	Probability	–
P_j	Probability of jth Class of Model	93
$P(M_j \mid K)$	Probability of jth Class of Model, given initial knowledge	99
Q	Decrement of probability	102
Q	Number of members of Class of Model	134
S	Strategy	85
T	Tolerance	68
U	Utility	89
$U(S_n)$	Utility of the nth Strategy	93
$\bar{U}$	Partial average of Utilities	106
W	Weighting for class of surface	156
W	Worse-case indicator	68

Symbol	Description	First Reference on Page
X_0	Least desirable consequence	89
X_1	Most desirable consequence	89
X_{ij}	Consequence of ith Strategy with jth Class of Model	86
Y	Summation of Utilities	103
$\mathbf{d}$	Displacement vector	67
$\mathbf{u}$	Variation vector	67
$\mathbf{C}$	Consequence matrix	115
$\mathbf{F}$	Inverse of Rigidity matrix	68
$\mathbf{R}$	Rigidity matrix	67
$\mathbf{S}$	View Selection matrix	114
$\mathbf{V}$	Visibility matrix	115
$\cup^*$	Boolean Union operation	18
$\cap^*$	Boolean Intersection operation	18
$-^*$	Boolean difference operation	18
ϕ	Abbreviation for "diameter"	60
γ	Homogeneous form of Lines in 2-space	B2

CHAPTER 1

INTRODUCTION AND SCOPE OF THE PROJECT

The author has sought to extend the range of graphics that may be produced from a geometric model by developing decision-making algorithms for the automatic production of dimensioned engineering drawings. A geometric model is, by definition, an informationally complete representation of a solid object. Such models are processed for various purposes: pictorial display, finite-element mesh generation, geometric properties, etc. One application which has received less attention from research workers is the automatic output of working drawings, even though their preparation and usage are still at the core of much engineering design and manufacture.

1.1 Aims and Tools for the Project

Even if the represented object is to be manufactured using numerically controlled machines, drawings will continue to find a place, for assembly, for inspection or simply as an easy way to check that the dimensions are correct. This last use is particularly important for the offline user who cannot interrogate the data structure by normal means. If the object is to be manufactured in whole or in part, by conventional manufacturing techniques then a dimensioned drawing will be required. Drawings may be required for commercial, technical or instructional purposes. Such drawings could, in theory, be produced completely automatically. The steps from geometric model to dimensioned working drawing require decisions on matters of increasing detail. The design and development of algorithms for making these decisions constitute the greater part of the work reported here.

The geometric form of the object is completely specified by the geometric model. However, this specification, although necessary for the production of a working drawing, is not sufficient. Additional data are required in order to produce an engineering drawing. These data specify the object's material, method of manufacture, standard of finish and, by reference to its function, the amounts by which certain dimensions may deviate from their nominal value.

From these inputs a draughtsman is able to produce a working drawing. The way in which he does this involves a complex set of decision-making processes because the solution he arrives at is only partially constrained by the problem definition. In most other applications of geometric modelling the solution is completely constrained by the problem definition. There is, for example, only one correct way to display a solid object, viewed from a certain position with its hidden edges eliminated. This is not to imply that there is only one method for solving a completely constrained problem nor that there may not be a range of acceptable solutions.

There are, for example, many ways to discover the roots of a polynomial equation and some roots found by using a numerical technique may be close, but not exact. The essential difference between the completely constrained problem and the type of problem discussed below is that it has a large number of valid solutions.

One approach to any problem with many valid solutions is to employ a numerical optimization technique. Although such methods have a contribution to make, they are not applicable to the problem as a whole. In order to optimize it is essential to define an objective function that is to be minimized. If one accepted that the only valid solutions to the problem are those which define the object completely and unambiguously (a point that is taken up in Chapter 4) it would then be necessary to find some figure of merit of a particular solution and to relate this to a large number of constrained independent variables. Clearly, such an approach is impracticable. Numerical optimization is not the right tool for the overall task.

Two other mathematical tools were examined for this project: Game Theory and Decision Theory. The former provides a guide for a "player" who seeks some desirable end, has a finite number of strategies (or rules of the game) from which to choose, and must consider the possible actions of another player who seeks his own ends. The latter provides guidance for those who seek to answer questions such as: "which strategy should I adopt in order to achieve a certain aim that will be affected by conditions outside my contol?" Clearly, the two subjects are closely related and, in much of the literature, there is little attempt to distinguish between them.

Undoubtedly, the development of a working drawing has both an aim and a set of rules that must be obeyed. It is then necessary to identify another "player" in order to make use of Game Theory. The decision-making process may be regarded as a game against nature but it was found that this was not a particularly useful concept for any of the algorithms that are described.

At one time, it seemed to the author that Decision Theory was not an appropriate tool for this project because all conditions are known. There is not the slightest possibility of changes to the geometric model that is being processed. Nor will there be changes to dimensioning standards or conventions during the process of producing the working drawing. Later, it became clear that this view was mistaken. It is argued in Chapter 5 that the model itself is the uncertainty.

There is a finite number of courses of action from which to select at each stage in the development of the working drawing but an infinity of different objects that could be modelled. Therefore, it is necessary to devise a finite number of classes of model and to seek to identify the class to which the model belongs. If the class of model cannot be identified with certainty, it is necessary to estimate the probability of it being a member of each class.

1.2 Stages in the Development of a Working Drawing

The following stages are essential to the process of producing a working drawing from a geometric model:

1.2.1 The choice of exterior orthographic views.

At least two views are required. If the object has any flat surfaces, however few and however limited in size, it is convenient to draw a view as seen in a direction normal to such a surface. There are two normal vectors and hence two viewing directions and one of these may reveal more detail than the other. It is necessary to compare these two views and to discover if one is in some way more useful for dimensioning purposes.

Having discovered useful views for dimensioning, it is necessary to determine a suitable combination of views which properly defines the solid object, can be drawn out in First or Third Angle projection without excessive annotation, and will enable the dimensions to be arranged around the views in a readable fashion.

1.2.2 The analysis of possible sectional views.

The analysis of external views may reveal that all the candidate views contain significant hidden detail. It is then necessary to investigate the internal surfaces of the object in order to determine if there is a useful sectional view. Single plane sections are considered first and, if no such section appears to offer satisfactory dimensioning possibilities, it may be necessary to investigate a multiple-plane section.

1.2.3 Preparation of Orthographic Views

An individual view is a projection of the solid object onto a plane. For dimensioning purposes it is sometimes necessary to display the object with hidden edges represented by dashed lines. Visible lines take precedence over hidden lines and it may be advisable to suppress some of the non-essential hidden detail.

The preparation of sectional views is a more complex operation than that for exterior views. Parts of the model are cut by an imaginary section plane or planes and, wherever the plane passes through the solid object, the sectional shape is cross-hatched. Some hidden detail may be placed on a sectional view, provided the net result is clear and unambiguous.

An additional complication is the widely-adopted convention of showing features such as ribs in outside view, when the cutting plane passes longitudinally through them. This is done in order to avoid giving a misleading impression of bulk, even though the resulting view is not strictly correct.

1.2.4 Construction and annotation of dimension lines

This operation is the final and most difficult phase of the entire process. It is necessary to decide how a particular feature is to be dimensioned, on which view the dimensions are to be placed, and how each new dimension should be arranged so as not to obscure those that have already been inserted. There are some general conventions to act as a guide but the entire subject is remarkably complex.

One of the complicating factors is the well-established practice of inference from missing dimensions. Thus, the dimensions that are not placed on a drawing may be quite as important as those that are.

1.3 Thesis Plan

The objective of the thesis is to describe a complex process in its logical sequence: from geometric model to working drawing, without distorting the picture by an over-emphasis of the author's contribution. This comprises a theory of dimensioning based upon surface classification (Section 4.5), and the ten decision-making algorithms for the preparation of engineering drawings, which are described in Chapters 6-10.

The ultimate objective of the work is the generation of a fully dimensioned engineering drawing from a stored shape description. The initial input to the procedures that are described is this shape description: the geometric model. Hence, the subject of Chapter 2 is Geometric Modelling, where special reference is made to the technique known as "Boundary representation" which was employed throughout this project.

Next, the basic tool that was developed for the definition of geometric models is examined. Chapter 3 first describes the hardware on which much of the work was done. Next, details are given of SUPERMODEL, the software developed for geometric modelling. Its data structure and some of its more original features are described. More details of SUPERMODEL, in the form of a user's guide, are given in Appendix C.

The decision-making algorithms described in later Chapters are designed to output a fully dimensioned working drawing of an object that is defined by using SUPERMODEL. In Chapter 4 the principles of dimensioning and tolerancing are described, some published theoretical work on dimensioning is examined, and a new approach is proposed and analysed.

In Chapter 5 the terminology and techniques of decision-making are examined. Particular emphasis is placed upon the theory of decision-making under conditions of "incomplete knowledge", as it forms the basis for several of the algorithms discussed in later Chapters. This tool was developed to aid decision-making in business. The ways in which it may be adapted to serve the requirements of this project are examined by slightly anticipating the material of later Chapters.

Chapter 6 initiates an examination of the process of selecting views of an object by dealing with a class of objects that may be adequately described by means of exterior views. In Chapter 7 there is an examination of whether to indicate hidden detail by means of dashed lines, a sectional view, or some combination of the two techniques. The literature of Hidden-surface algorithms is surveyed at the start of Chapter 8, before treating the specific problem of the preparation of an orthographic view for dimensioning purposes.

The author has considered two approaches to the insertion of dimensions. In Chapter 9, the strategy is to process one orthographic view at a time, requiring interaction with the software user. It is the user who must consider the interrelationship between the orthographic views. Feedback from the users of this algorithm led to a decision to reject this approach and indicated a new direction of attack.

Chapter 10 gives details of an algorithm that processes the entire geometric model, selecting the view on which to indicate the relevant dimension. In Chapter 11, the project is discussed and some conclusions are stated, the areas where work remains to be done are identified, and some potential benefits are listed .

Clearly, the concept of a geometric model is central to this project. It is the primary input to the decision-making algorithms that are discussed in later Chapters. There are many techniques of geometric modelling in the literature; some of the more widely-used schemes are described in Chapter 2 before turning attention to the particular one that was adopted for this work.

CHAPTER 2

GEOMETRIC MODELLING

2.1 Computers, Geometric Models and the Design process

An artifact produced within a primitive society is likely to take the form that is does because a pattern was firmly implanted in the memory of the person who manufactured it. The man who made the article learned how to do so from some mentor who, by word and practical demonstration, defined the materials, method of manufacture and final form. Such means of communication are entirely adequate for the societies where they are employed. In industrialised societies the need to innovate and to produce in quantity were the main reasons for the adoption of drawings as the primary medium of geometric specification.

It is arguable whether attempts to make computers draw pictures led to the development of computer-draughting systems or if it was the high labour cost of engineering drawings that promoted interest in exploiting computers for this activity. Whatever the reasons there was a noticeable trend towards the computerization of draughting in the 1970's. There are now over 20 international companies offering systems which replace the drawing board, pencil and microfilm with graphics display, digitizer, magnetic tape and plotter. Sales of these systems have grown rapidly because they improve productivity by significant factors and they allow design specifications to be managed with greater efficiency.

However, there is much that computer-aided draughting systems cannot do. The main deficiencies are in the capture of the information contained in hand-produced drawings and in the procedures that may be carried out on new drawings. A straightforward draughting system cannot, for example, compute the volume of the object whose views have been drawn. The essential difference between a computerized engineering drawing and a geometric model of an object lies at the core of such problems. The former data have to be interpeted by a human brain or computer-system. The result is simply the interpreter's "understanding" of the data. A geometric model is an informationally complete and unambiguous representation of a solid object.

It follows that such a representation permits any well-defined geometrical property to be calculated automatically. This does not imply that the computation is trivial nor that the methods used will be obvious and easily discovered.

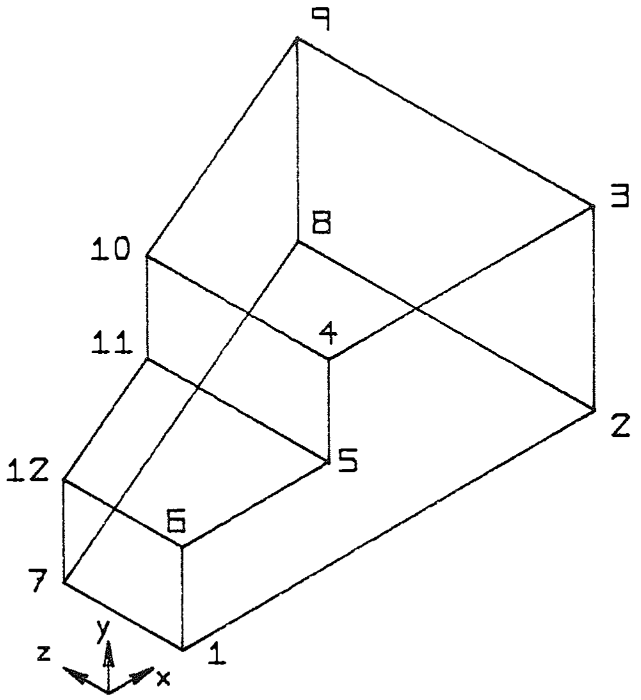
The general purpose of geometric modelling is to define a set of points in 3-space. Requicha [1] asserts that six families of schemes for the unambiguous representation of solids are known; he excludes the "wireframe" representation on the grounds that it leads to ambiguities. However, the technique is of historical importance and serves as an introduction to the subject.

2.2 Wireframe representation

A wireframe model consists of a set [N], of Nodes in 3-space and a set [E], of Edges which make connections between the Nodes. The position of each Node is specified by (x,y,z) coordinates from some convenient origin. An Edge may be defined in terms of its two terminal Nodes. This leads to a simple data structure consisting of a 3-column array of real numbers for the nodal positions and a 2-column array of integers for the Edges.

This modelling technique is illustrated in Fig. 2.1 and the two arrays are listed in full. The major advantages of the system are simplicity and efficiency of storage. Two major disadvantages are illustrated in Fig. 2.2 and 2.3 - the model at the top of Fig.2.2 may be interpreted as any of the three pictorial views shown below it. Fig.2.3 illustrates the problem that arises with Edges that are not straight lines. A particular view of a cylindrical object should show a tangent between the end circles but the position of this tangent depends upon the viewing direction and, if it is stored as an Edge, there is a loss of generality.

The other well-established disadvantages of wireframe modelling are its verbosity and credulity. A user has to input much low-level data in order to define an object as simple as that shown in Fig.2.1.



Geometry Matrix

Node	x	y	z
1	5	0	0
2	19	0	0
3	19	6	0
4	10	6	0
5	10	3	0
6	5	3	0
7	5	0	4
8	19	0	10
9	19	6	10
10	10	6	6.143
11	10	3	6.143
12	5	3	4

Topology Matrix

Edge	Start	Finish
1	1	2
2	2	3
3	3	4
4	4	5
5	5	6
6	6	1
7	7	8
8	8	9
9	9	10
10	10	11
11	11	12
12	12	7
13	1	7
14	2	8
15	3	9
16	4	10
17	5	11
18	6	12

Fig. 2.1 Wireframe model with data structure

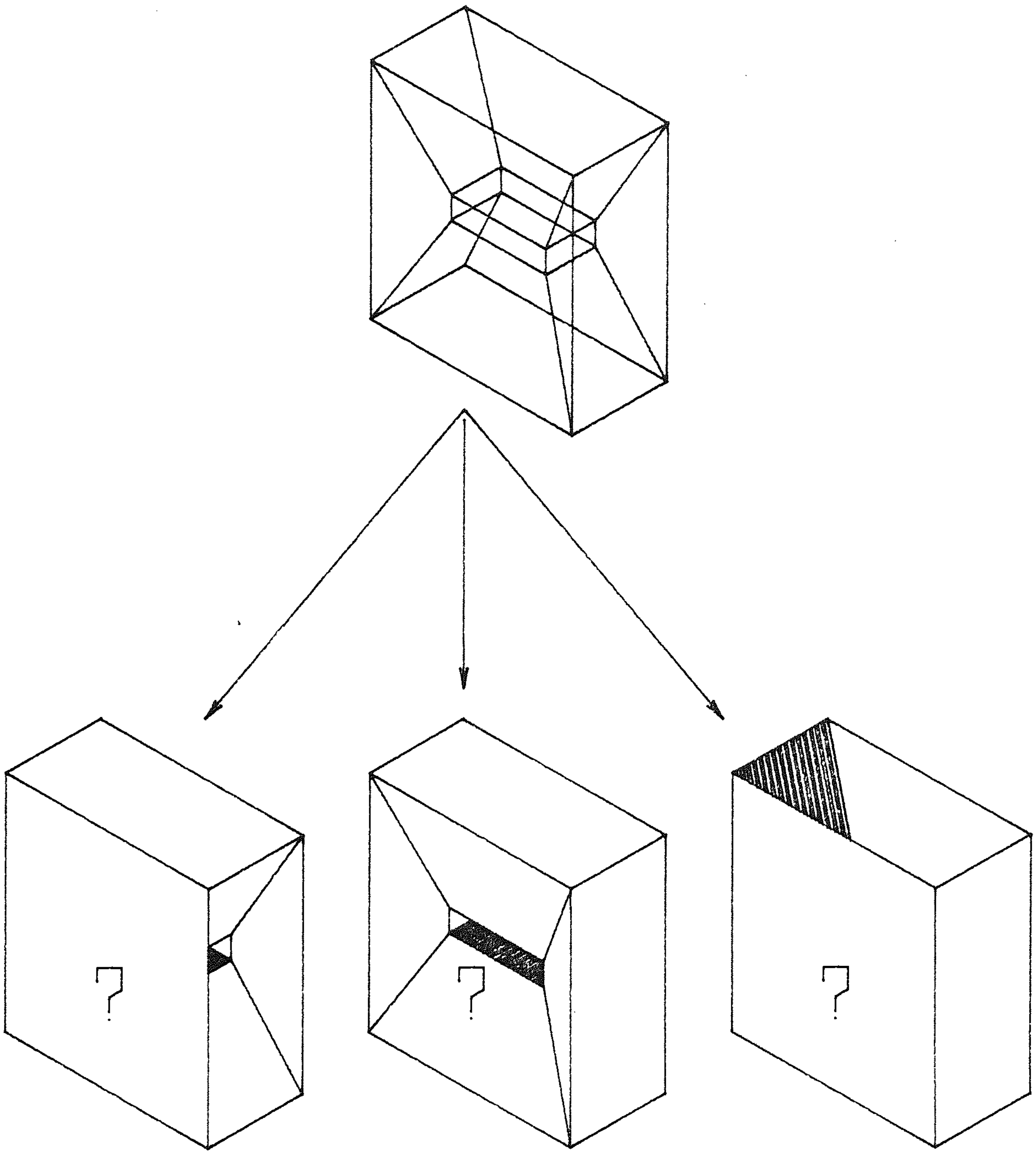


Fig. 2.2 Ambiguity of wireframe modelling

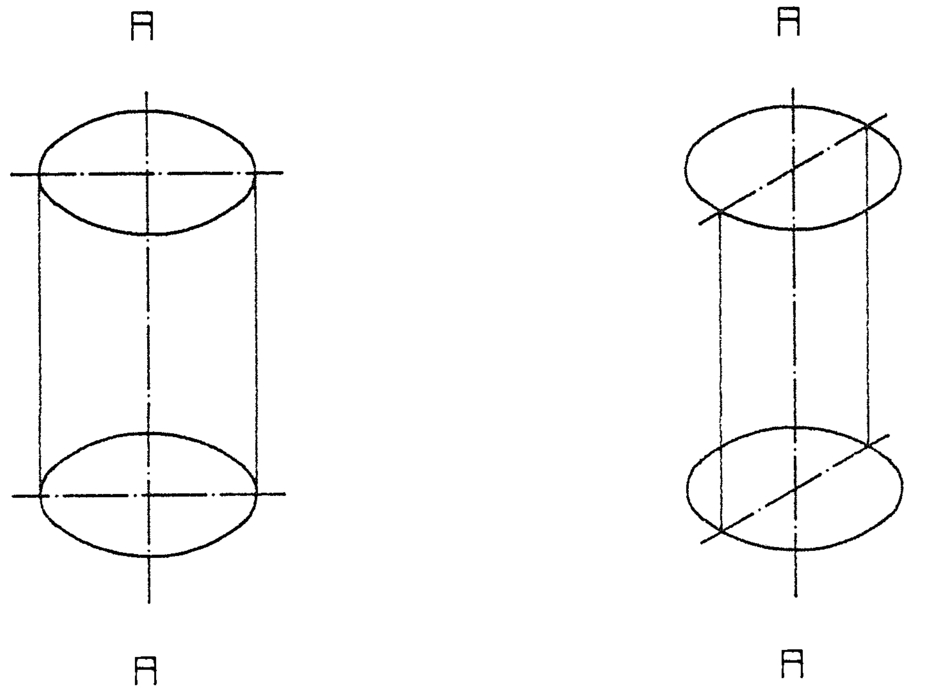


Fig. 2.3 Loss of generality in wireframe model

Without a set of very sophisticated checking algorithms a wireframe modelling system will accept all kinds of impossible objects, such as occur if one Edge is missing or left dangling in space. However, in spite of the above criticisms, wireframe modelling has been successfully exploited for commercial applications. Much of this success is due to the quality of the enhancements that have been made to the basic technique. Other workers, notably Braid [2] and Baumgart [3], investigated and developed techniques for "true" solid modelling, three examples of which are now briefly described.

2.3 Boundary Representations

Every solid object is finite. The interface between the object and its surrounding medium is a boundary which delimits the object. This boundary may be specified as a set of surfaces. These surfaces may be planar or have single or double curvature. The proper assemblage of such surfaces is a Boundary Representation geometric model. Such a model is shown in Fig.2.4

Clearly, it would be difficult and tedious to construct such a model by direct input to a computer. Those systems which employ a model of this kind usually generate the model by conversion from a user-friendly input protocol. A quasi-3-dimensional technique is a typical example; draughting is done on a working view and its effects on other views of the object are shown.

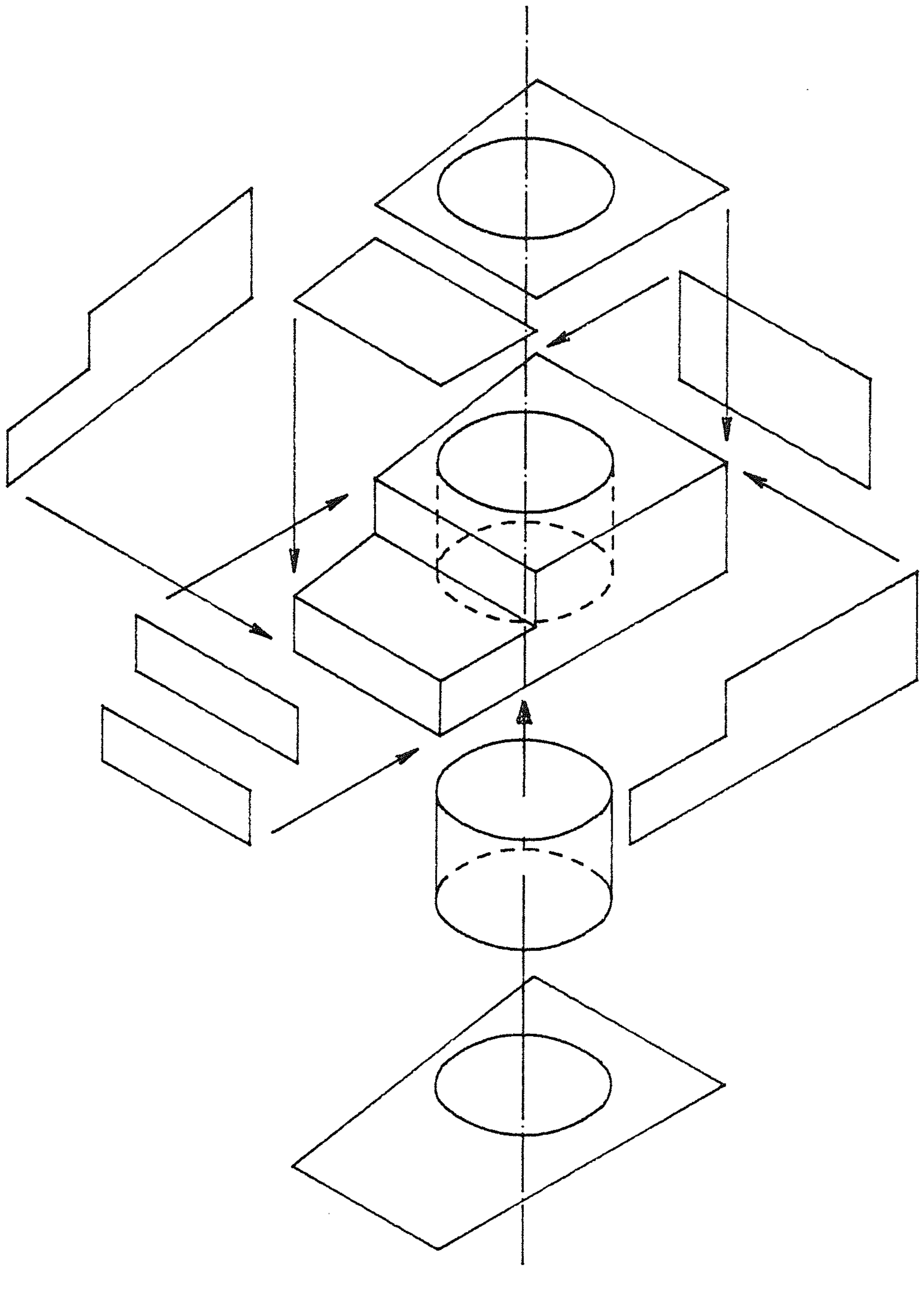


Fig. 2.4 Boundary representation technique of Geometric Modelling

2.4 Constructive Solid Geometry (CSG)

The fundamental concept of CSG is that a complicated solid can be regarded as being formed by carrying out various operations on much simpler, "primitive" shapes. Fig. 2.5 illustrates the Boolean set operations of Union ($\cup^*$) and Difference ($-^*$) used to model the same object as Fig. 2.4, where it was shown as a Boundary Representation.

The Boolean operator symbols shown in the binary tree are starred to indicate the significant difference between this type of operation and those with the same names that are used in classical Set Theory. Algorithms used in CSG have to ensure that the result of a Boolean operation on representations of two solid objects is also a representation of a complete, unambiguous solid object.

Some CSG schemes work with the lowest possible level of primitive input - the "Half-space". Fig. 2.5 indicates that the cuboid at the bottom-left may be represented as the Intersection ($\cap^*$) of six Half-spaces. A planar Half-space is the infinite plane defined by:

$$ax + by + cz + d = 0$$

plus all points on one side of the plane.

Other CSG implementations use a wide range of primitive shapes. The cuboid and cylinder are common to most. The cone, wedge and sphere are useful primitives for mechanical engineering components.

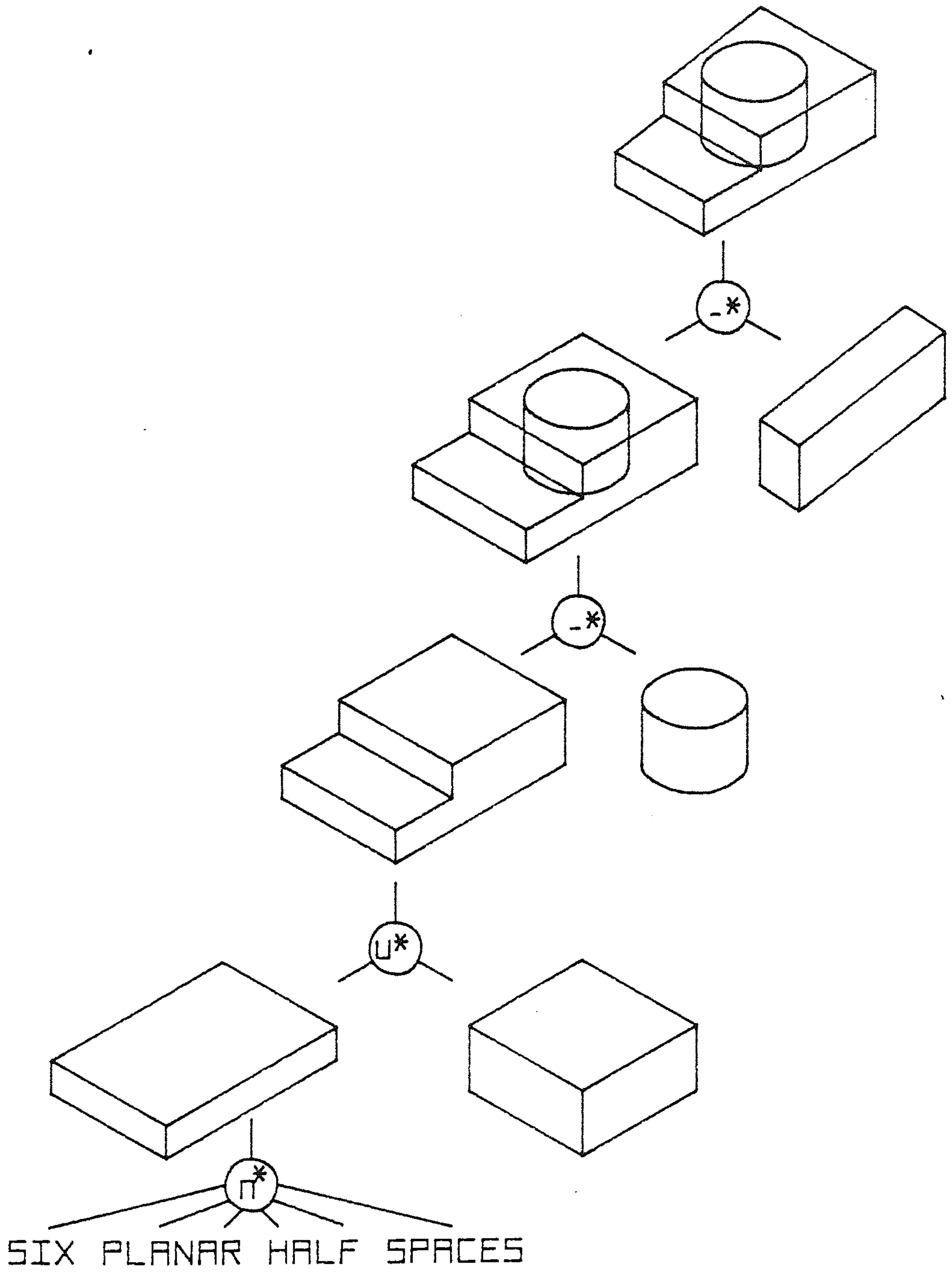


Fig. 2.5 Boolean operations in Constructive Solid Geometry

2.5 Sweep Representations

As the objective of geometric modelling is to define a set of points in 3-space, it is highly convenient to begin by defining a set of points in 2-space: an area, and then sweep it along a prescribed path. The Cartesian product of an "Area-set" and a "Trajectory-set" represent a "Solid-set". Fig.2.6 illustrates this concept at the lower-left: an Area Set A is swept vertically upwards to produce a Solid-set X. The process is analogous to the manufacture of an extruded object, i.e. one produced by translational sweeping. Axi-symmetric objects may be represented by rotational sweeping and the process is analogous to profile turning.

The solid shown in Fig 2.4 and 2.5 also appears at the top of Fig. 2.6 but it cannot be produced by simple sweeping and so some modelling systems provide hybrid operations which combine sweeping and a down-graded Boolean Union operator which has been dubbed "gluing". In Fig. 2.6 two Solid-sets X and Y are produced by translational sweeping. These sets are then "glued" (G) together to create the object illustrated.

The algorithms required for converting a user-friendly input protocol into a valid solid model are less complex than those required for operations in Constructive Solid Geometry. Intuitively, the technique appears to restrict the range of models that can be built, but there is no firm evidence that this occurs in practice.

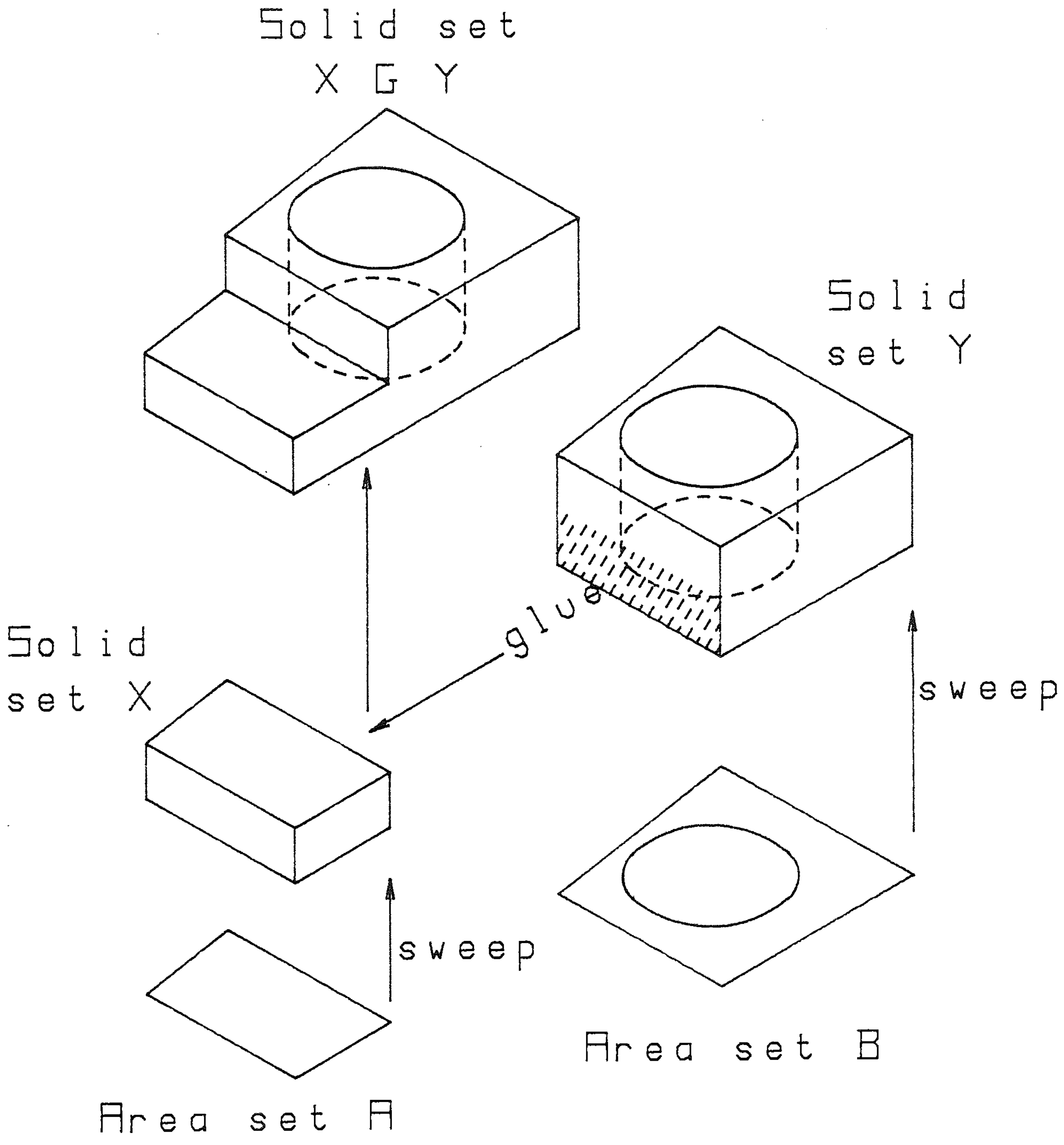


Fig. 2.6 Solid model produced by translational sweeping and gluing

A whole new range of rather interesting solids may be modelled by defining the "area-set" as a function of one independent variable and the "trajectory-set" as a function of a second independent variable. Barr [4] and his co-workers have specialized in the use of the super-conic section as an area-set which is then modulated by the sweeping function to produce superquadric solids. Fig.2.7 shows a selection of superquadrics.

2.6 Applications

Geometric Modelling techniques have developed rapidly in the past decade and the commercial exploitation of such techniques is now gathering momentum. Requicha and Voelcker [5] list some of the Geometrical Modelling packages that are now commercially available. It is clear from this list that the preferred modelling techniques are Boundary Representations (B-Reps) and Constructive Solid Geometry (CSG). Solid modelling has established a foothold in commercial CAD/CAM in a remarkably short time but, at present, only three major applications are understood well enough to be handled automatically in most systems:

a) Graphics - Orthographic and Perspective pictorial views, Engineering drawings, etc.

b) Mass property calculations

c) Static interference checking

There are several well-publicised applications which, at

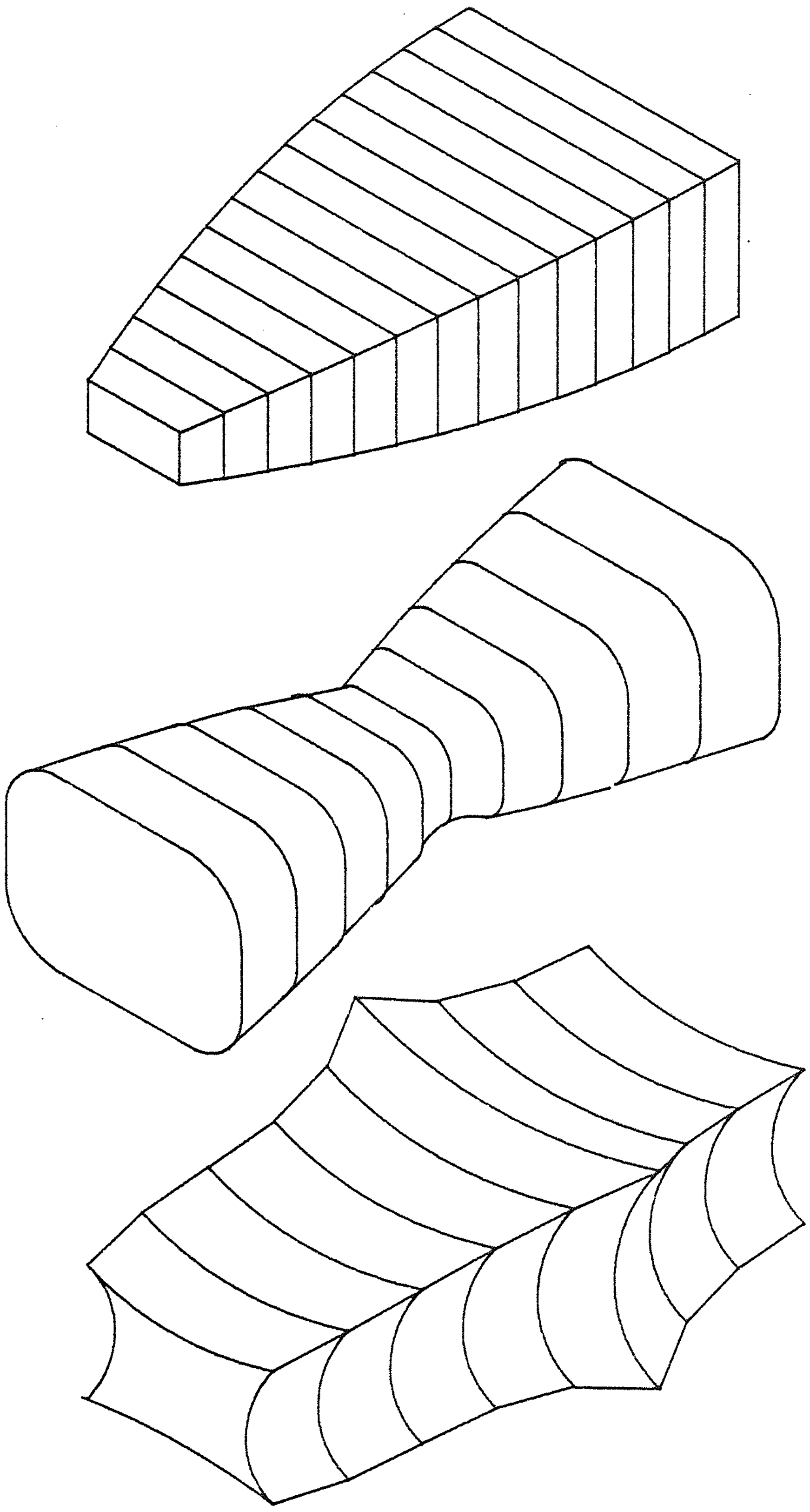


Fig. 2.7 Examples of Superquadric surfaces

present, must be regarded as important research projects which have yet to reach the commercial market:

d) Dynamic interference checking

e) Interpetation of Engineering Drawings as solid objects

f) Automated stress and temperature distribution analysis.

This thesis describes a project which falls squarely into category (a) above. The automatic production of dimensioned drawings was seen by Braid[op cit] to be a highly desirable application as long ago as 1971. Hillyard and Braid [6] state: "This work is intended to lead to the production of dimensioned views from stored shape descriptions." Although progress has been made towards this end, the range of models that can be handled is limited and the problem of view selection does not appear to have been addressed.

2.7 Comparison of Geometric Modelling techniques

The survey by Baer, Eastman and Henrion [7] distinguishes four categories of model: the "Definition language", the "Data representation", the "Conceptual model", and "Application program". Initially the designer/draughtsman must enter the shape description in a form that can be interpreted by the computer system. This "Definition language" may employ graphical or keyboard entry or both. Additionally, shape modification capabilites are provided in all practical systems. The language is usually tailored to the needs of the designer/draughtsman.

Statements in the "Definition language" may be evaluated upon entry into an expanded internal form that contains more explicit shape information needed for applications. It is quite possible to use the "Definition language" for this purpose and many systems do so. Any expansion or rationalization of the input code, even if this is not written to mass-storage, may be regarded as a "Data representation" model.

Baer, Eastman and Henrion [op cit] note that the most general formal representation of a solid body is a set of contiguous points in 3-space and that the abstract problem of representing a shape is to define this point set. In earlier work, Eastman [8] used the point set concept literally and represented 3-space by a three-dimensional array with a solid object as simply a set of adjacent cells. The same approach was tried but abandoned by Braid [op cit] : the first scheme he describes (pp.27-29) relied on building bricks of unit size. The representation is highly redundant because any cell is likely to have the same state as the cells adjacent to it. Only at the boundaries of a solid object is this not so.

The Constructive Solid Geometry (CSG) approach has the merit that only true solids can be designed. However, as

Braid, Hillyard and Stroud[9] point out, certain kinds of shape are difficult to design using this system. The original BUILD program implemented by Braid [op cit] incorporated a variable primitive which the designer could specify by using methods closer to traditional draughting practice. The price of this useful addition was the loss of the assurance that the input defined a solid object.

During the period that BUILD was written, a more general method of creating shapes directly was developed by Baumgart[op cit]. The mathematician Euler suggested a rule relating the number of elements of a convex polyhedron:

$$F + N - E = 2$$

where F, N and E are the number of faces, Nodes and Edges. As the objects represented always obeyed Euler's rule Baumgart referred to them as Euler objects. The operations for processing such constructs he called Euler operations and these terms are now widely used in commercial as well as research applications.

Thus, Baumgart's fundamental concept is that a solid may be represented by storing a description of its boundary. The Boundary representation is divided into surfaces of finite area. Each surface may be defined by a single equation. Surfaces meet in pairs at Edges, often but not always at a discontinuity of slope. An Edge runs between two Nodes and each Node is specified by (x,y,z) coordinates from some convenient origin. Surfaces may be multiply-connected and are bounded by one or more Circuits of Edges.

One major drawback of the Boundary-representation technique is the ease with which the user may input nonsense objects. Baer, Eastman and Henrion [op cit] provide a digest of conditions for well-formed surfaces. The conditions of importance include requirements that the surface be closed, orientable, nonself-intersecting, bounding and connected. The application of Euler's rule is shown to satisfy the bounded and orientable conditions.

The Boundary-representation technique was adopted for this project. The primary reason for this choice was the fact that the ultimate objective of the project is the output of engineering drawings which consist of orthographic views. The viewing directions are generally normal to the bounding surfaces of the object. Thus, this form of representation implies a data-structure which is particularly suitable for the output of orthographic views.

A secondary reason for the adoption of the Boundary-representation technique is that the historical means by which the surfaces are defined by the draughtsman are found to be of significance for dimensioning. The CSG and Sweep-representation techniques allow the user to specify and manipulate primitive shapes, but it is doubtful if the way in which he does this has any relationship to the function or manufacture of the object represented. It is argued in Chapter 9 that the actual procedures of draughting boundary surfaces provide valuable additional data about the object, which other modelling techniques cannot.

For the purposes of this project the Euler rule and its extensions were used to ensure that the input data could be interpreted as a solid object. The main reason for this check concerns the drawing of hidden detail lines on views which are to be dimensioned. The absence of just one relatively small surface has a disproportionate affect on the accuracy of the orthographic view.

Some techniques of Geometric Modelling have now been examined. The geometric models which are processed in order to output a dimensioned drawing could be formulated and input directly in "Data representation" format. However, it was soon clear that realistic engineering components are so complex that direct input was impracticable.

Existing modelling software was not considered suitable because of cost, limited access to the computers on which it could be mounted, lack of access to the source code, lack of information about the data structure, and deficiencies in the routines provided. For these reasons it was necessary to develop a wide-ranging suite of draughting and modelling routines. The Boundary-representation technique was adopted as being more appropriate for this work.

The input to the decision-making algorithms is therefore a Boundary-representation model. This "input model" is prepared by using a suite of routines developed by the author and named SUPERMODEL. These routines constitute the apparatus which is used to generate the "input model". Those features of the apparatus which are primarily designed for modelling are described in Chapter 3.

CHAPTER 3

THE INTERACTIVE GEOMETRIC MODELLING PACKAGE

3.1 Introduction

It was clear from the start of the project that a comprehensive modelling package was required. The type of mechanical engineering components that were selected for analysis were found to be of such complexity that the direct formulation and input of geometry and topology matrices was impracticable. It was essential that the model be created by using interactive graphics and that the data file produced be readily accessible by other programs.

No suitable software was accessible for this work and so the package described below was developed in parallel with the decision-making algorithms which are the subject of later Chapters. With hindsight, this requirement has been advantageous in at least one important respect: the decision-making algorithms were not constrained by the demands of the geometric modeller. Whenever such a situation threatened, the modeller was enhanced.

The development of these routines was carried out on various machines at the University of Aston in Birmingham and on a DEC PDP11/55 at the University of Connecticut. For simplicity, the final package, as

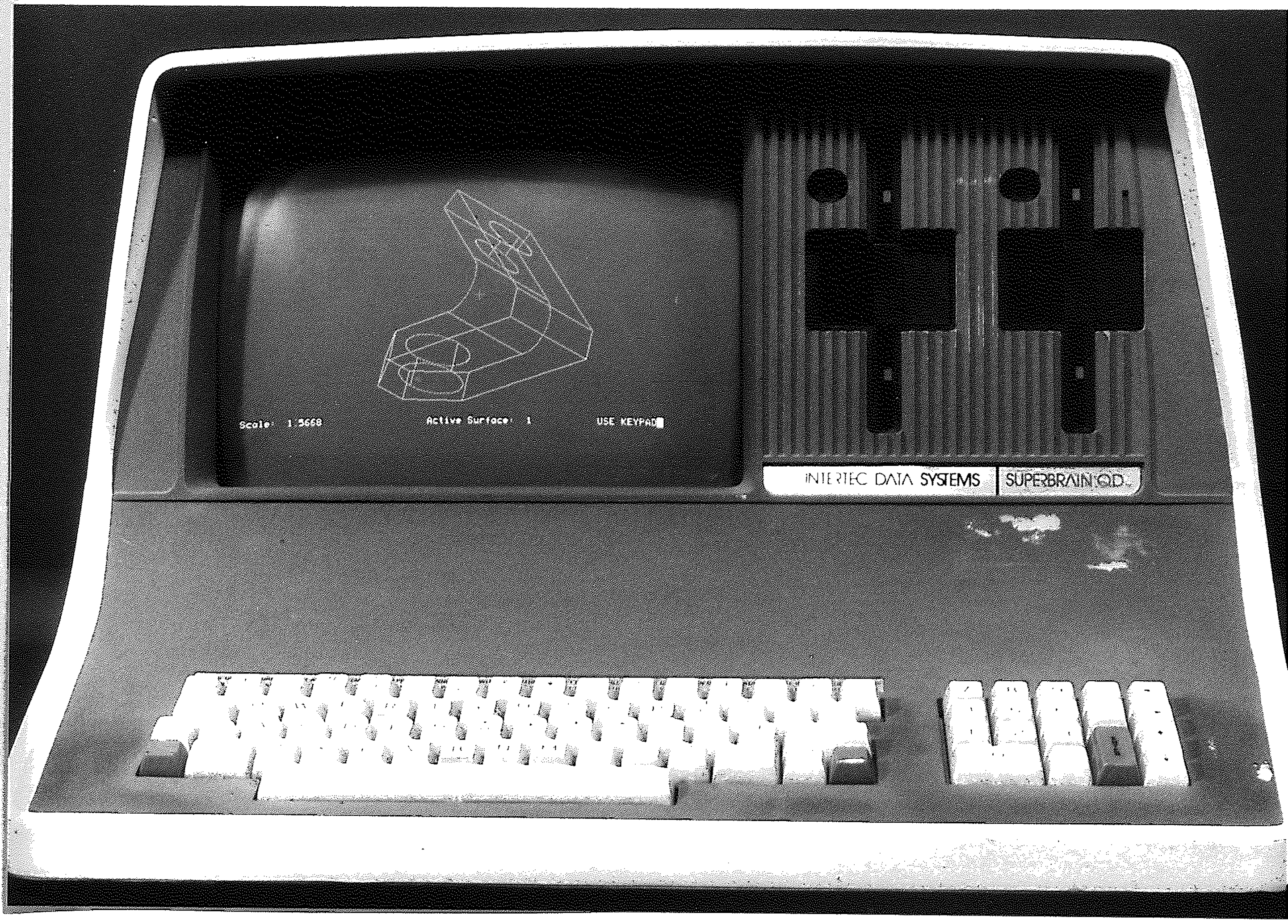


Plate 3.1 Superbrain Microcomputer with model displayed

implemented on an Intertec Superbrain QD, is described below and no other implementations are detailed.

Plate 3.1 illustrates the type of machine on which the modelling algorithms were implemented. Displayed on the screen is a representative model, viewed in isometric projection, before hidden surface elimination has been carried out.

The Superbrain computer has an adequate graphics capability, being able to address 512 x 255 pixels. In the hope of achieving some degree of transportability the routines were written in a version of FORTRAN 66 - Microsoft F80. Initially, linking and loading were executed using Microsoft L80 but, as the program grew, the drawbacks of this Linker became too great and later work was linked using the LYNX [10] stand-alone, multiple segment linkage editor. Its features are comparable to linkage processors found on minicomputers and mainframes.

For convenience, the suite of programs has been dubbed SUPERMODEL. A front end view of the system is given in Section 3.2. The data structure of the software is described in Section 3.3 and then the modelling algorithms are detailed in Section 3.4. This Chapter concludes with a description of the model display algorithms. A user-orientated guide to the package is given in Appendix C.

3.2 Front End

The Superbrain computer has two diskette drives. Executable code, overlays, a messages file and a character generation file are stored on the diskette in Drive A (the left-hand slot visible on Plate 3.1). The data generated in the course of making a drawing are stored on the diskette in Drive B. When the drawing is complete and a hard-copy is required the data file is processed by other programs.

The software is menu-driven. The user interacts with the display by means of the arrow keys which are visible at the extreme right of the keypad on Plate 3.1. Each touch of an arrow key moves a cursor one preset step in the selected direction. Functions are selected from a menu of approximately sixty items. The user selects a menu-item by typing the one or two-digit code required. The general purpose of each item is shown in Appendix C, Table C.1

Overlaid routines called from the menu are automatically loaded from diskette; the user is merely aware of a slight delay while loading takes place. The user has up to 64 surfaces available in which to draw or write. He may place any combination of straight lines and circular arcs into the currently active graphics surface, or write text of any size or orientation into the currently active text surface. The user has a wide range of geometrical definition routines available. He may initiate a search for a particular element, delete or modify it.

Each surface may either be displayed or not. The default scale may be reset and the user may zoom in or out of the current mix of displayed surfaces. Any surface may be stored on diskette with a file-name specified by the user. This file is distinct from the overall drawing file and may therefore be used as one item in a library of parts built up by a user.

3.3 Data Structure

The primary subdivision of the total set of data stored by SUPERMODEL is a surface. There are two types:

3.3.1 A graphics surface which consists of up to 42 Edges. An Edge is a straight line or a circular arc. The line-style for each Edge may be either continuous, dash, chain or continuous with arrow head.

3.3.2 A text surface for the writing of dimension figures, titles, labels, notes, etc. Each text surface may contain up to 32 strings of text. Each string may have up to 80 characters and both the height of the characters and the angle at which they are written are infinitely variable.

The definition of both types of surface is based upon a geometry matrix which stores 64 sets of (x,y,z) coordinates with 4-byte precision. This matrix requires

4x3x64 = 768 eight-bit bytes of diskette-storage space and conveniently fits into 6x128 byte records.
The topology of the surface is specified with a 42 row x 3 column array which stores integers with 1-byte precision. The first column specifies the addresses of the start-point coordinates. The second column specifies the addresses of the finish-point coordinates. If the value stored in the third column is zero then the Edge is a straight line between start and finish points. Otherwise, the third column specifies the address of the Centre of a circular arc.

The addresses of the start and finish points are signed according to the line-style. For example, a continuous line has a negative sign for both start and finish point addresses. The sense in which an arc is to be drawn is specified by the sign of the integer in the third column. A positive integer denotes an arc that is drawn anti-clockwise from the start to the finish point.

The 126 byte topology array, plus two bytes for the maximum number of Edges drawn and the maximum number of coordinates defined, fits conveniently into one record. Thus, each surface is stored as seven records on diskette which can be rapidly brought into memory using an unformatted READ statement.

The structure for a text surface is slightly different. The start and finish point addresses are stored in the

same way as in a graphics surface. After looking up the coordinates it is possible to compute the angle and the spatial length of the string. The contents of the third column is the address of the text string in a separate text file. This string is read and the number of characters determined; it is then possible to compute the width of each character cell.
There is also an index array which is stored in one record of the text file. This array is used to look up the starting addresses of a particular surface. It handles up to 64 surfaces: 38 graphics surfaces and 26 text surfaces.

3.4 Modelling Algorithms

The set of objects that may be defined, using SUPERMODEL, are those bounded by plane and/or conical surfaces. A typical model, unfolded into its component flat surfaces and with curved surfaces appended, is shown in Fig. 3.1. The set of plane surfaces that may be specified are those bounded by straight lines and/or circular arcs. The set of conical surfaces which the user may input are those which lie between two circular arcs (complete circles not excepted) situated in parallel planes. The cones may be right or oblique.

It may be seen by reference to the SUPERMODEL menu (Table C.1), that the user has a comprehensive range of geometric definition routines available from which to

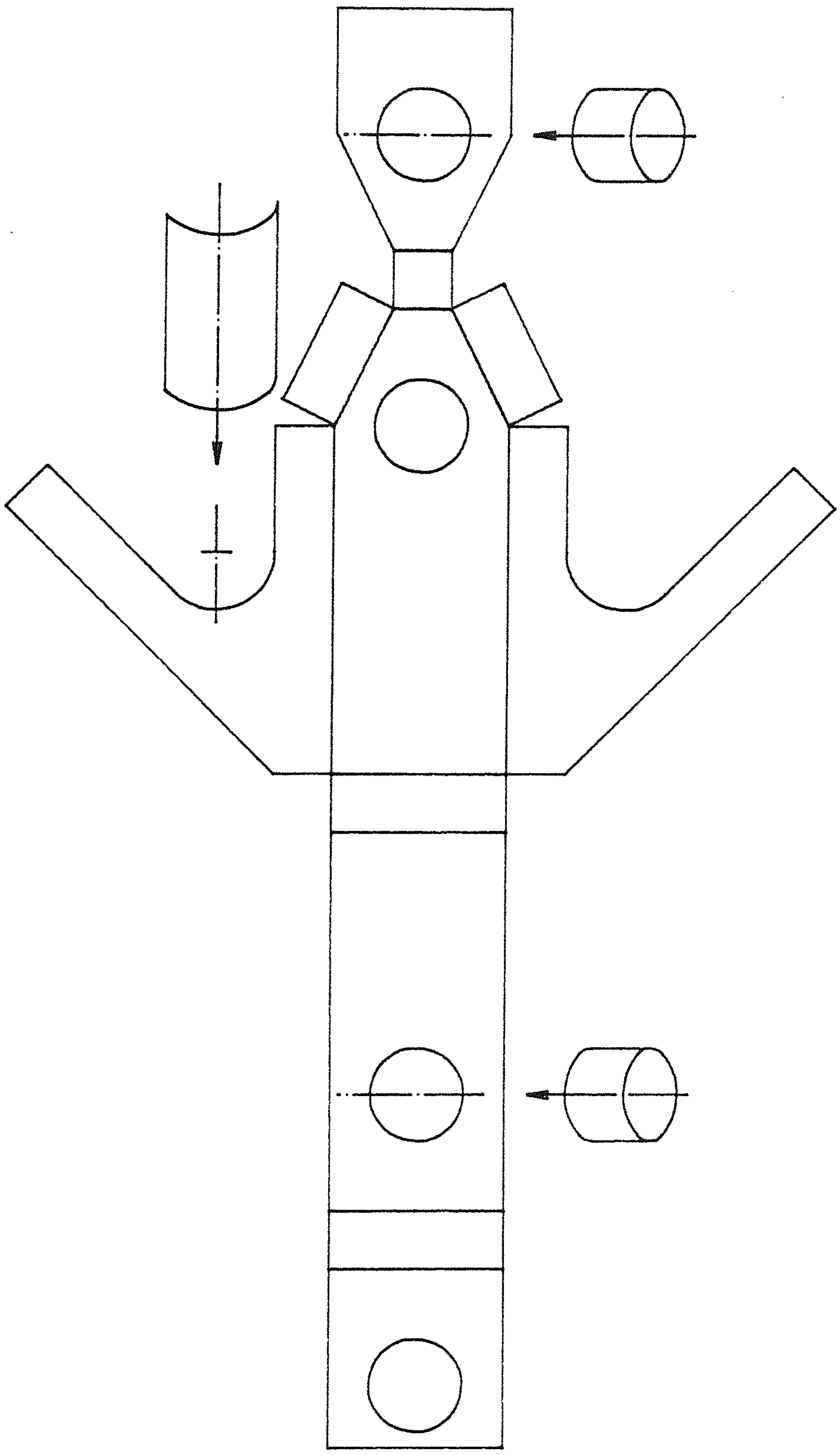


Fig. 3.1 Component surfaces of a geometric model

construct his plane and conical surfaces. The range of geometric definition tools required by engineers was deduced from documentation for various applications as well as by reacting to comments from users of early versions of the software.

From reference manuals such as NELAPT Part Programming [11] and the GINO-F Graphics Package [12] it is clear which are the most useful routines to have available. It is on the margin of this catholic selection that provision of a particular routine may or may not be justified. Fortunately, the overlaying capability provided by LYNX [op cit] means that each additional routine provided merely increases the size of the overlay file and does not necessarily increase the demand for random access memory.

Space does not permit the inclusion of details of the SUPERMODEL geometric definition algorithms in this thesis. Suffice it to say that considerable use was made of the techniques of homogenous coordinates, some of which are described in Appendix B. The interested reader will find much useful information in Newman and Sproull [13]. Algorithms for geometric definition, implemented in FORTRAN, are to be found in the book by Bowyer and Woodwark [14]. Angell [15] also has some helpful chapters on this subject. Algorithms for the output of straight lines and circular arcs to a pixel-mapped screen may be found in Walker, Gurd and Drawneek [16] and Bresenham [17]. Faux and Pratt [18] treat some of the relevant topics in computational geometry.

3.4.1 Placement of flat surfaces into the model

Conceptually, the user initially constructs a flat surface which lies in the plane of the computer display screen. In other words, he specifies the coordinates of Nodes and Centres directly or indirectly and, for the moment, all z coordinates are zero.

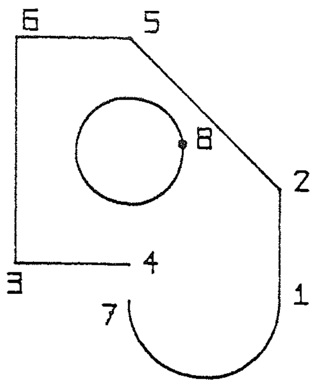
When he is ready to place this surface into the model he first defines the axis about which it is to be rotated by specifying a start point and a finish point which lie on this axis. If the surface lies in a plane of constant z-value, then there is no need to specify this start-finish axis. When the user opts to position a flat surface he is first prompted to input the z coordinate of the surface he has defined. The start-finish axis and the surface are translated into that plane. The next action required of the user is to input the angle (in degrees, anticlockwise) through which the surface is to be rotated around the start-finish axis.

The decision-making algorithms which process the model that is prepared by using SUPERMODEL require that the data presented to them truly represent a solid object. In order to guarantee this, SUPERMODEL first checks that the line/arc Graph which the user has input bounds an area and is non self-intersecting. Fig.3.2 illustrates three line/arc Graphs and their SUPERMODEL topology arrays. The Graph at upper left does not bound an area and should be rejected. The Graph at upper right does bound an area but there are two intersecting Edges; it should be rejected. The Graph at the bottom of Fig.3.2 passes both tests.

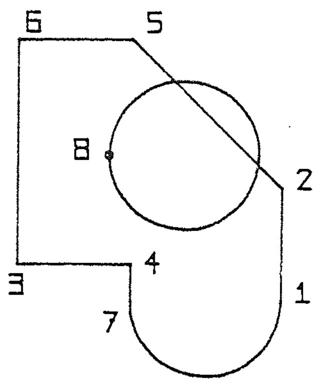
The method of checking for bounding is to count the number of times that each "active" Node appears in the first or second column of the topology array. Nodes which are terminal to either dashed, chain or arrow lines are considered inactive and are ignored. If every "active" Node appears an even number of times then the line/arc Graph bounds an area. If this check fails then the user is informed that the surface cannot be placed into the model.

The method of checking for non self-intersection is time-consuming but data are generated during the process which can be put to good use at a later stage. First, the straight line Edges are expressed as column vectors (see Appendix B) and the radius, start angle and finish angle of each arc Edge is computed. Using these data, each Edge is tested against every other Edge for a valid intersection, i.e. one which lies between the terminal Nodes of the Edge. If a valid intersection is found then the Graph is rejected and the user is immediately informed.

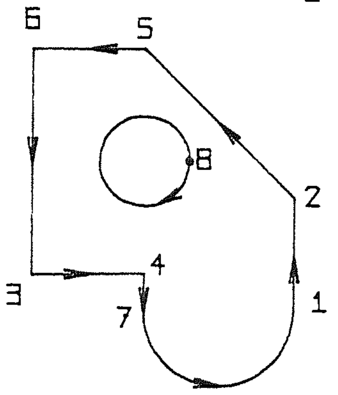
The "initial data" shown in Fig.3.2 includes some rows in the topology array with a zero in the first column. This is a flag to indicate that the Edge which was



Edge	Start	Finish
1	1	2
2	3	4
3	5	6
4	7	1
5	8	8
6	6	3
7	2	5



Edge	Start	Finish
1	1	2
2	3	4
3	5	6
4	7	1
5	8	8
6	6	3
7	2	5
8	4	7



INITIAL DATA		CIRCUIT SORT		DIGRAPH SORT	
Start	Finish	Start	Finish	Start	Finish
1	2	8	8	1	2
3	4	3	6	2	5
0	1	6	5	5	6
5	6	5	2	6	3
7	1	2	1	3	4
8	8	1	7	4	7
6	3	7	4	7	1
2	5	4	3	8	8
0	4				
4	7				

Fig. 3.2 Check and conversion of line/arc Graphs before placement of flat surface

stored in that row has been deleted. The SUPERMODEL topology array is updated by searching first for such flagged rows whenever a new Edge is to be stored. If no such rows are found then the new Edge data are added to the end of the topology array. Because of this method of updating and the possible presence of dashed and chain Edges the topology array is now edited and rearranged into Circuits.

This rearrangement is shown in Fig.3.2 where it is labelled "Circuit Sort". Each Circuit consists of a closed Edge sequence or one complete circle, which is a special case. The mechanism for rearranging the rows is basically to interchange rows so that the initial Node of one Edge is the same as the terminal Node of the Edge in the row above it. The order of Nodes in Edges may need to be reversed and this requires the sign for the Centre of an arc to be changed.

One further rearrangement is necessary in order to generate a valid solid model: the Circuits must be directed so as to indicate on which side of an Edge the solid material lies. This is to satisfy what Eastman and Preiss [19] term the "orientability condition" and Klein [20] dubs "Mobius' Law". An orientable surface allows recognition of its solid and non-solid sides.

If this condition is not met there is ample evidence in the literature of problems transmitted to application software. For example, Barton and Buchanan [21] define several concepts designed to ensure generality and correctness in the specification of polygons built using their package. A propos the point discussed here, they define a "sheet" as a set of points enclosed by an ordered ring of alternating Nodes and Edges. The polygons in the package which they describe consist of a number of such "sheets".

The author prefers to employ the terminology of Graph Theory (see Appendix A): a Graph which consists of one or more Circuits is converted to a Digraph in which the solid object is always on the same side of a di-Edge. The convention adopted for SUPERMODEL is that, when viewed from outside the object (as opposed to through it), a flat surface has material on the left side of a directed Circuit. The user is therefore asked to indicate whether the surface he has drawn is as viewed from outside or through the solid object. The arrows on Fig.3.2 indicate the di-Edges and the rearranged topology array is labelled "Digraph sort".

The data which were initially computed for the non-self intersection check are reused in STEP 3 of the following procedure. If there are NCIR Circuits and NEDGE Edges to be processed then the algorithm for producing the Digraph is as follows:

STEP 1. Set Circuit counter, ICIR, to 1.

STEP 2. Compute the coordinates (X,Y) of a point, P, half way along the first Edge of the ICIRth Circuit. If the surface is being viewed from outside the object then displace P by a small amount to the left of the Edge direction, otherwise displace P to the right.

STEP 3. Compute the number of intersections, N, of a line to the right through (X,Y) and all NEDGE Edges. If N is odd then go to STEP 5.

STEP 4. Reverse the order of the Edges in the ICIRth Circuit.

STEP 5. Increment ICIR by 1. If ICIR is does not exceed NCIR then go to STEP 2, otherwise exit.

This is the final data rearrangement that is done for the purposes of validating the model. Later validation procedures are carried out on all the surfaces specified by the user. Euler's rules are the basis for these procedures. The z coordinate of the Graph before rotation, the coordinates of the start-finish axis and the angle or rotation are stored in the geometry array and both geometry and topology arrays are written to diskette.

3.4.2 Placement of conical surfaces into the model

The user first draws two circles, or circular arcs which form the two flat, parallel surfaces at the ends of the conical element. As with flat surfaces (Section 3.4.1), if the arcs are not to be in planes given by: z = a constant then the user must specify a start and finish point in order to define the axis around which the arcs are rotated.

When the user opts to position a conical surface, he is

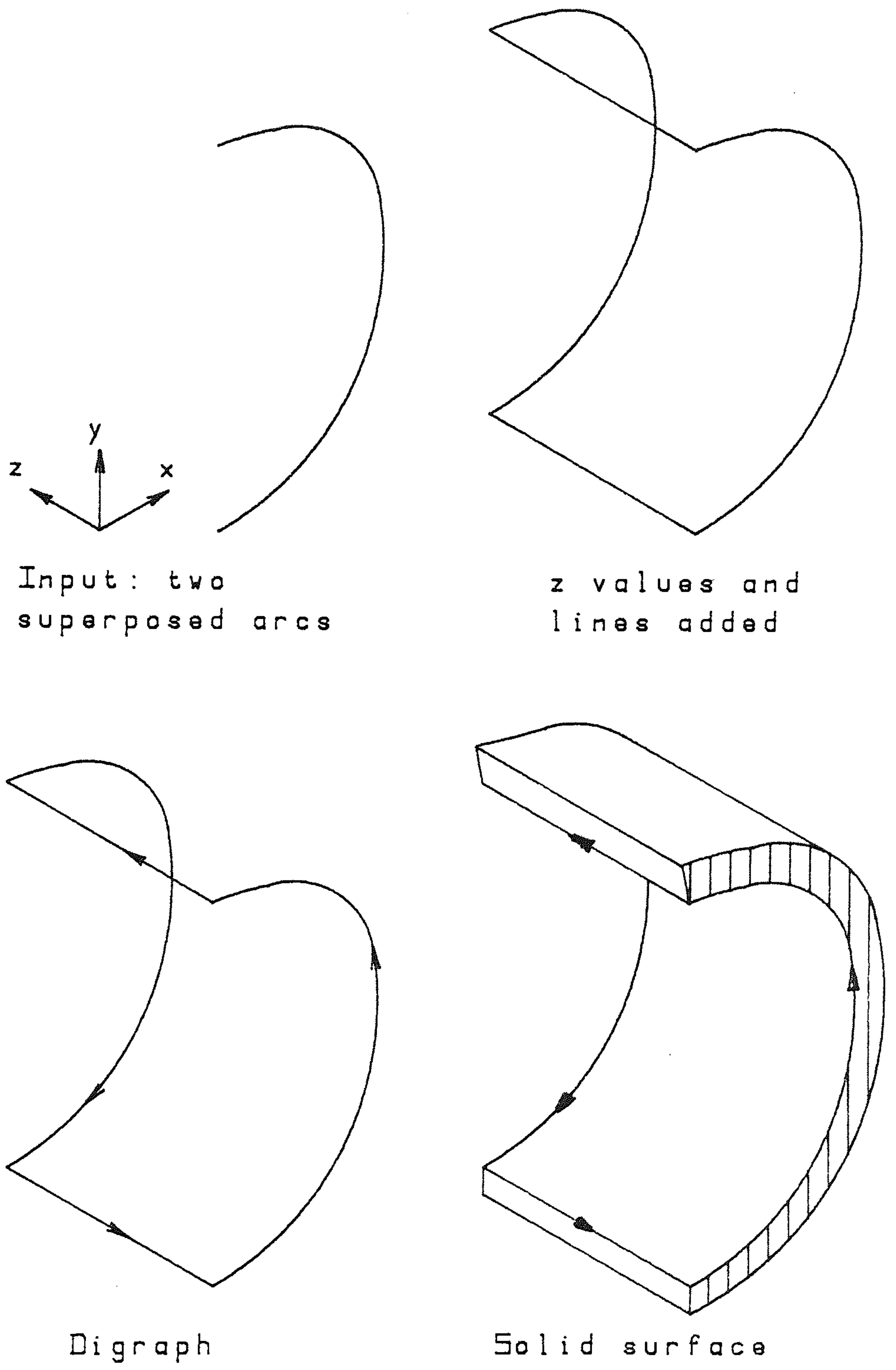


Fig. 3.3 Construction and checking of curved surfac

prompted to input the z-coordinates of the first and second arcs. SUPERMODEL first checks that there are exactly two arcs within the active surface and that they are not of dash, chain or arrow-head type. Both arcs must subtend the same angle at their Centre. If either check fails then the user is informed of the fact. Otherwise, the arcs are accepted for rotation about the start-finish axis which is taken as co-planar with the first arc. The user is prompted to input the angle of rotation which follows the same convention as for the positioning of flat surfaces.

Fig.3.3 illustrates the original user input and the additions that are made by the software. Two straight line Edges are added to the data structure to connect the ends of the arcs and to define a bounded conical surface. In the case of two circles the added Edges will be coincident and terminate at the arbitary start/finish point on each circle. Such superposition is necessary in order to maintain a consistent data structure but causes problems when the surface is to be displayed; there is no discontinuity of slope and so no Edge should be visible. This point is taken up again in Chapter 8.
The user is then asked to indicate whether there is solid material on the Centre side of the first arc. The response is used in order to arrange the four Edges into a Digraph. The correct Digraph is indicated by the arrows in Fig.3.3. The arcs are then drawn in their new

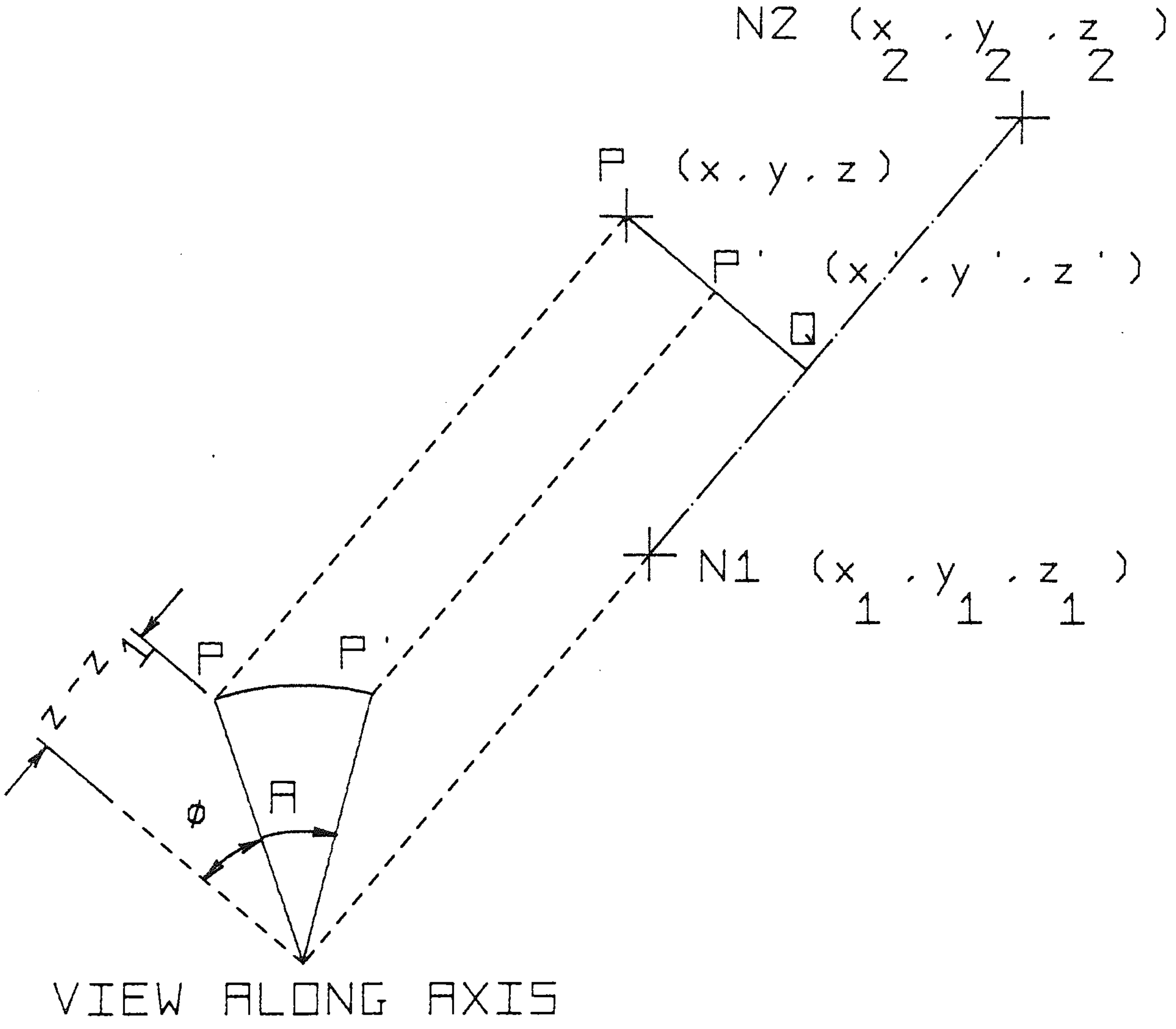


Fig. 3.4 Position of point within total model

positions and the data-structure of the surface is updated to indicate that a curved surface has been created. The data are written to diskette.

3.5 Display Algorithms

Any SUPERMODEL surface may be rotated about an inclined axis which lies in the same plane as a flat surface, or in the same plane as the first-drawn arc of a conical surface. Hence, the most general case is as illustrated in Fig. 3.4. A point P (x,y,z) (which may be a Node or some arbitrary point along a circular arc) is to be rotated about a line which passes through points N1 (x_1,y_1,z_1) and N2 (x_2,y_2,z_2). The SUPERMODEL display algorithm is limited to the case where z_1 equals z_2, but does not necessarily equal z.

The effect of a rotation around N1-N2 is to move P to P' along the perpendicular PQ. The line N1-N2 may be expressed in homogeneous coordinates as:

$$au + bv + c = 0 \text{ , where}$$

$$a = y_1 - y_2$$

$$b = x_2 - x_1$$

$$c = x_1 y_2 - x_2 y_1$$

Point Q has coordinates $(x_1+bt,\ y_1-at,\ z)$, where

$$t = \frac{b(x-x_1) - a(y-y_1)}{\sqrt{a^2 + b^2}}$$

Hence, the length, L of perpendicular PQ is given by:

$$L = \frac{ax + by + c}{\sqrt{a^2 + b^2}}$$

After rotation through angle A, the new position of the point may be determined by defining a parameter, f, where:

$$f = \frac{x' - x_1 - bt}{x - x_1 - bt} = \frac{R\cos(\phi+A)}{R\cos\phi} = \frac{L\cos A - (z-z_1)\sin A}{L}$$

or $\quad f = 1$, for the special case when $L=0$

Therefore the new position, P', is given by:

$$x' = fx + (1-f)(x_1+bt)$$

$$y' = fy + (1-f)(y_1-at)$$

$$z' = z_2 + L\sin A + (z-z_1)\cos A$$

The same equations may be used for the rotation of the entire model around some specified inclined axis. A point on the boundary of a SUPERMODEL surface is first rotated around the local axis, appropriate to that surface and then rotated around the global axis. The display algorithms that were implemented on the Superbrain were found to be have acceptable drawing times even when working around large arcs by incrementing the angle subtended at the centre. Thus, it was unnecessary to adopt some more sophisticated pixel-plotting algorithm.

The tool that is used to define geometic models has been described. It was developed by the author, initially to facilitate the input of "boundary representation" models, but was later extended in order to implement some of the algorithms that are described below.
The geometric models that may be built are to be processed by decision-making algorithms in order to produce a suitable combination of exterior views and sectional views in orthographic projection. These views will then be dimensioned and toleranced in order to output an usable engineering drawing. It is appropriate, therefore, in Chapter 4, to focus attention on the principles of engineering drawing, and in particular, dimensioning and tolerancing, so as to gain a clear view of exactly what the decision-making algorithms are designed to output.

CHAPTER 4
PRINCIPLES OF DIMENSIONING AND TOLERANCING

4.1 Historical Background

Contemporary methods of dimensioning and tolerancing on Engineeering Drawings are due in great measure to the underlying requirement for interchangeability of components. Booker [22] suggests that this is the result of an historical accident rather than being a rational development. The need for systematic dimensioning and tolerancing arises in any environment where the design and manufacturing functions are largely separated. Thus, a cooper produces the complex geometry and water-tight fit of a wooden barrel without the benefit of a dimensioned working drawing because he is both designer and manufacturer and his components are not interchangeable.

The development of the techniques of mass-production was fostered by two distinct motives - ease of service/overhaul and cost of manufacture. In 19th century American society a hunting rifle was normally kept in good repair by its owner. The manufacturers of such weapons were able to supply him with spare components to replace broken parts and these components would fit any rifle of that design. The manufacturers of clocks (for the same society) made the parts interchangeable, not to facilitate repair, but so that they could be assembled by cheap, unskilled labour. At this time, interchangeability was not achieved through the medium of drawings but through the introduction of gauges and templates. More sophisticated manufacturing techniques employed special tools and jigs.

In 1815 the American Federal government adopted a plan to produce muskets in armouries located in many different places by copying perfect weapons which had been made from master gauges and jigs. Each armoury was required to make its own tools from the patterns supplied and was instructed to permit no deviations from the pattern. The method may have been somewhat extravagent but it was successful in its declared aim of producing complete interchangeability. Some forty years later the British Small Arms Commission, as reported by Roe [23], recommended the Government to adopt "The American System" in order to achieve the same objective.

Ultimately it was not this method but a totally different technique which led to the significant separation of the manufacturing and design functions. Instead of supplying a perfect sample of the hardware to copy it was gradually realised that a precision drawing of the object could be made to serve the same purpose, provided it was carefully annotated in such a way that the geometry was completely and unambiguously defined. The reasons why this medium came to be preferred are easily understood. The drawing is readily sent from department to department, company to company or country to country. Chemical processes enabled it to be copied cheaply and developments in metrology ensured that standard sizes could be maintained and the accuracy of the final component verified.

Nevertheless, the success of the engineering drawing as a means of communication rests heavily on its ability to define the geometry of the hardware it represents to the required degree of precision. Potentially, the dimensioned drawing is so versatile in this respect that geometries may be described which are difficult, if not impossible, to attain in practice. This type of problem is less frequent than another which arises in new design work and has caused incalculable loss of time and money in the 20th century.

A report - "Dimensional Analysis of Engineering Designs" (24) published in 1948 pinpointed certain difficulties in the production of new designs. The authors concluded that such problems were generally attributable to one or both of the following causes:

a) Incorrect analysis of dimensional requirements of the design and

b) Incorrect, ambiguous or incomplete statement of the requirements on the engineering drawing.

It is all too easy for an engineering drawing to become a representation of the ideal. A component which has holes at either end may be produced by ,say, boring one hole and then rotating the component through 180^{o} and boring the other hole. The measured sizes of the holes may agree very precisely with the sizes stated on the drawing and yet they might not be quite on the same centre-line and thus cause problems in assembly and operation. Another frequent source of trouble is the machined surface which is not quite square to another surface.

The problem addressed in "Dimensional Analysis of Engineering Designs" was that sometimes these inaccuracies of manufacture mattered and at other times they did not and yet they were represented on the drawings by an abstraction which gave no indication of what variation from the ideal could be tolerated.

Practising engineers have long been aware that, in order to manufacture a component, it must be "fully" dimensioned. Missing dimensions inevitably cause problems and expense. If the manufacturer refers back to the drawing office then there are delays but if he scales from the drawing or makes what he considers to be an intelligent guess then there are bound to be occasions when he makes the wrong guess and produces unsatisfactory components. Because the former course is generally the less expensive, many pre-printed company drawing sheets carry the bold legend: "IF IN DOUBT, ASK".

There are however, several well-established conventions which experienced engineers accept largely unquestioned. An angle which looks like 90^{o} is assumed, in the absence of any other information, to be a right-angle. Surfaces which look parallel are assumed to be so unless there are dimensions which specify or imply their convergence. Components that are perfectly or largely symmetrical about some axis are usually only partially dimensioned with reference to this axis: the symmetry is implied. Similarly, a set of holes on a pitch circle are often assumed to be equally spaced even if this is not explicitly stated.

The wording of the plea: "IF IN DOUBT, ASK" has been well chosen because the manufacturer would rarely be "in doubt" about some of the conventional "missing" dimensions whereas a lay person might be. Yet communication is only possible between two people to whom the same tokens (be they written or spoken words, symbols or diagrams) have essentially the same meanings. Thus, in engineering drawings, as in other media of communication, what is left unsaid is quite as important as what is stated.

The foregoing illustrates the need for a system of geometric tolerancing such as was developed in both North America and Europe during the immediate post-war years. Foster [25] lists several reasons for adopting such a system, the more important of which are:

a) it reduces costs,

b) it ensures that requirements which relate to the actual function of the object are specifically stated, and thus carried out,

c) it ensures interchangeability of mating parts at assembly and

d) it provides uniformity in draughting and interpretation, thereby reducing controversy and guesswork.

Various countries have adopted methods which enable the designer to specify tolerance of form and position. By these means he can take into account the actual function and relationship of the features of the parts he is designing. Details of the British Standard Institution's recommendations for this process are given at the end of the next section.

4.2 Contemporary British Practice

Although there is considerable variation in conventions, technique and emphasis the recommendations of British Standard 308 [26] have had an undoubted influence on many of the drawings in current use in this country. The original 1927 edition of B.S.308 contained three pages of text and one sheet of diagrams. The current edition is considerably larger but the recommendations on dimensioning principles are fairly easily summarised.

4.2.1 General Principles

Fig.4.1 illustrates the fundamental terms and format of dimensioning. The dimension is stated next to a thin

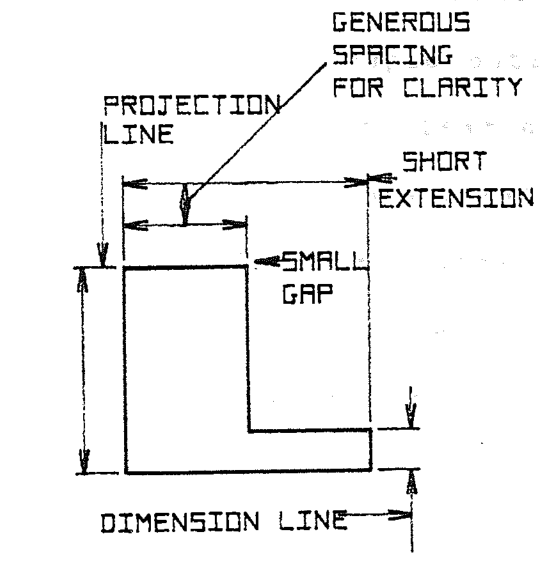


Fig. 4.1 Dimensioning definitions

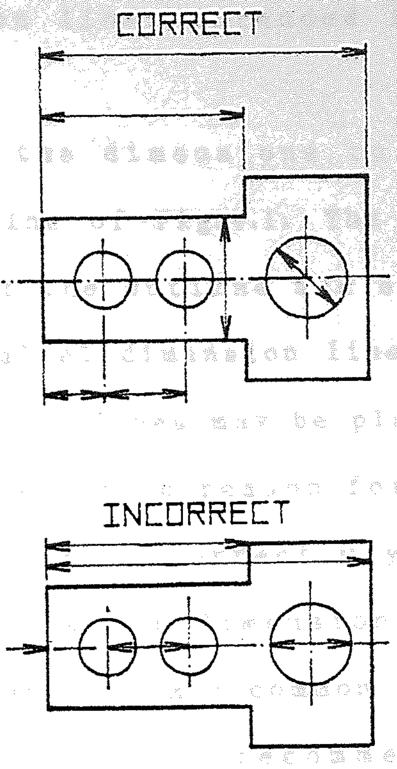


Fig. 4.2 Correct and Incorrect methods of dimensioning

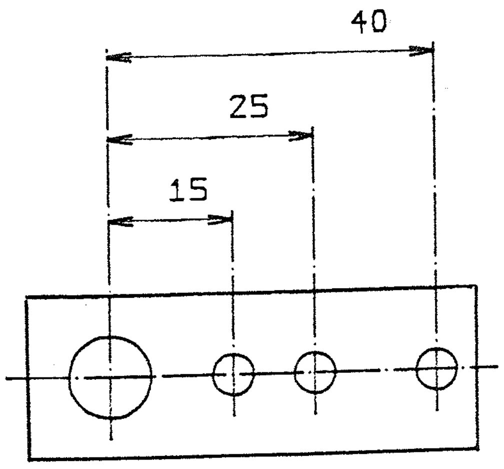


Fig. 4.3 Dimensions from common datum

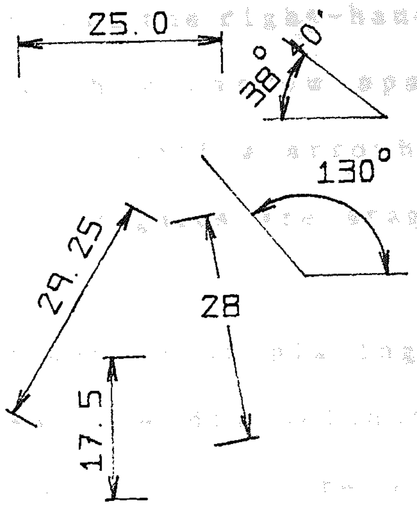


Fig. 4.4 Orientation of dimension figures

dimension line which has arrow-heads at either end. The arrowheads touch either the appropriate features of the object or thin projection lines extended from these features.

Projection lines enable the dimensions to be placed outside the L-shaped outline of Fig.4.1. The projection lines start just clear of the outline and extend just beyond the arrowhead. Parallel dimension lines are well spaced for clarity. Dimension lines may be placed within the outline if there is some good reason for doing so. Fig.4.2 shows the correct and incorrect ways of using projection and centre lines for dimensioning. Where several dimensions are stated from a common datum line, the method shown in Fig.4.3 is recommended. The dimension figures are placed near the appropriate arrowhead.

4.2.2 Arrangement of dimensions

In Fig.4.4 the numbers are orientated so that they may be read from either the bottom or the right-hand edge of the drawing. Fig.4.5 shows how narrow spaces are dimensioned by using inward pointing arrowheads. In Fig.4.6 the parallel dimension figures are staggered in order to improve clarity.

Fig.4.7 illustrates the advantage of placing overall dimensions outside intermediate dimensions: if the intermediate ones were outside then there would be a confusing crossing of lines.

Dimensions of diameters should be placed on the most

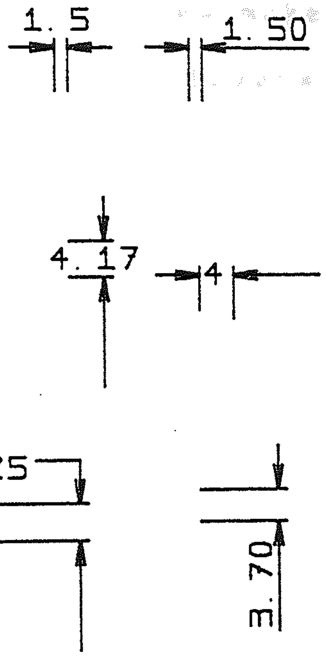


Fig. 4.5 Treatment of small dimensions

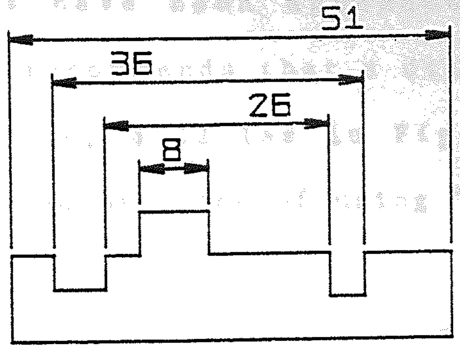


Fig. 4.6 Staggered dimension figures

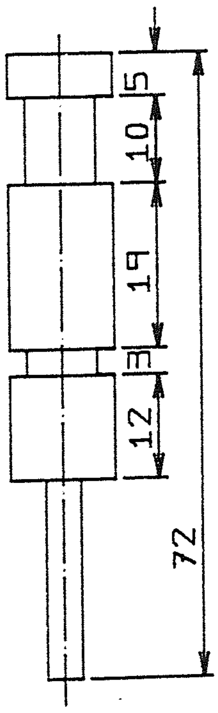


Fig. 4.7 Overall and intermediate dimensions

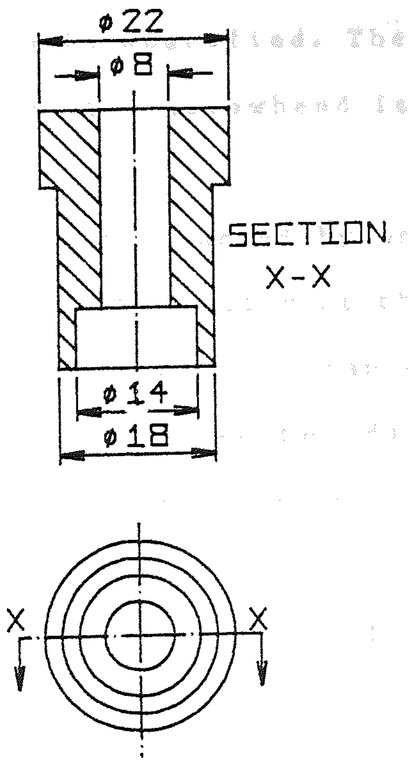


Fig. 4.8 Dimensioning of concentric circles

appropriate view. The set of concentric circles in the lower view of Fig.4.8 is difficult to interpret but the sectional view makes clear the reason for each circle and so the dimensions have been appended to the sectional view. B.S.308 recommends that a diameter is indicated by, for example, ϕ 22 (as in Fig.4.8) but there is still a widespread practice of using "22 DIA".

4.2.3 Circles, Radii and Holes

Circles are dimensioned by one of the methods shown in Fig.4.9. The method chosen is to some extent governed by the size of the circle and if there are other concentric circles to be dimensioned. Fig.4.10 shows how radii of arcs are dimensioned by a line lying along a radius. If the centre of the arc need not be located then it need not be indicated. If the centre must be located but lies in an inconvenient place then a fictitious centre may be drawn but its true position is specified. The part of the dimension line that bears the arrowhead is along a true radius.

An individual dimension may be toleranced by one of the methods shown in Fig. 4.11. The dimension at the top of the figure shows a size with limits of tolerance equally disposed above and below it. The next two dimensions show a size with a limit of tolerance in one direction only. The dimension at the bottom has both limits of size specified directly and is the preferred technique for the dimensioning algorithms described in Chapters 9 and 10.

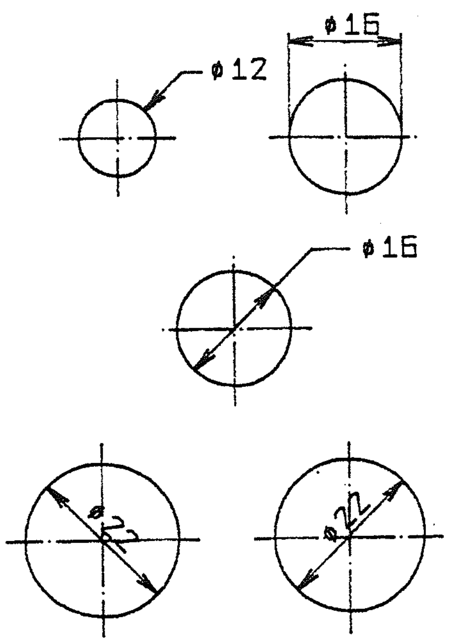


Fig. 4.9 Techniques for dimensioning circles

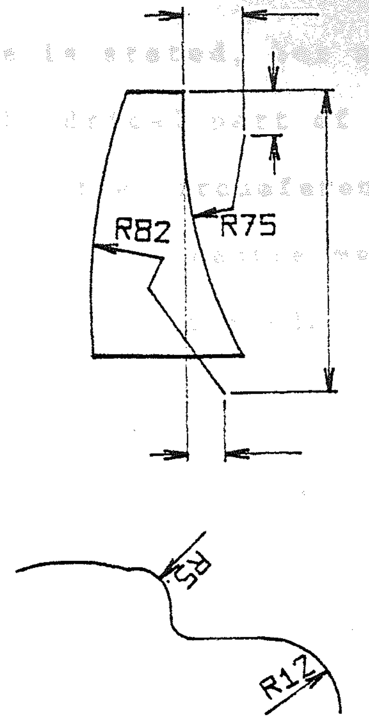


Fig. 4.10 Techniques for dimensioning arcs

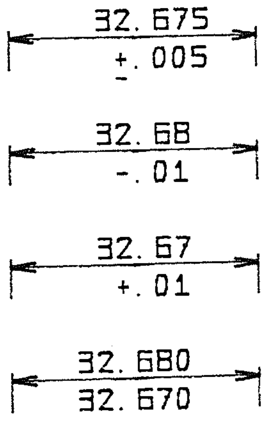


Fig. 4.11 Toleroncing of dimensions

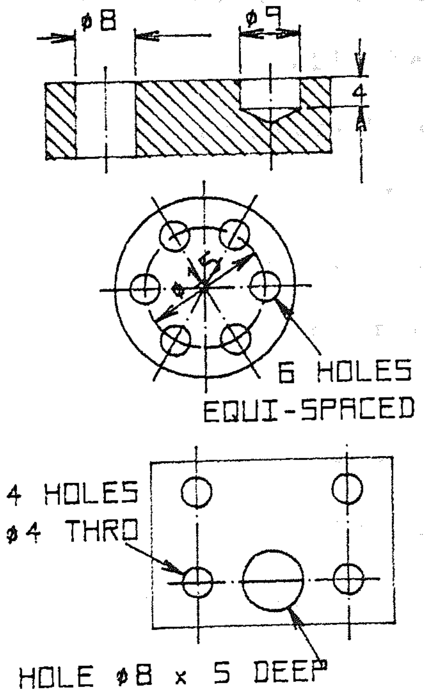


Fig. 4.12 Techniques for dimensioning holes

Holes are dimensioned by one of the methods shown in Fig.4.12. Identical holes may be specified by a leader line that points to only one of the holes. When the depth of a drilled hole is stated, but not drawn, the depth refers to the cylindrical part of the hole only. Holes whose centres lie on the circumference of a circle and subtend equal angles at the centre may be specified for position by the method illustrated.

4.2.4 Geometrical Tolerances

British Standard 308: Part 3 contains recommendations for the specification of straightness, flatness, roundness, cylindricity, parallelism, squareness, angularity, concentricity, symmetry and position. It states (p.6): "Geometrical tolerances should be specified for all requirements critical to functioning and interchangeability except when it is certain that the machinery and techniques which will be used can be relied upon to achieve the required standard of accuracy." The draughting details need not be discussed here but the relevant ones are examined in connection with the dimensioning of the total geometric model which is treated in Section 10.1.

Within the recommendations of B.S.308 there is still considerable scope for individual draughting preferences and the exercise of judgment. This, of course, means that the design of algorithms for the automatic output of an engineering drawing is, itself, a matter of judgment. This point is referred to again in Section 11.5, where the requirements for commercial software are discussed.

4.3 Dimensioning and Graph Theory

It is helpful to employ the terminology of Graph Theory in order to describe and to analyse some of the processes of dimensioning. Some introduction to the concepts of Graph Theory is given in Appendix A. A single view of an engineering drawing may be regarded as a Graph because its components are points in 2-space (Nodes) and various types of connections between Nodes. For engineering drawings derived from SUPERMODEL these connections are straight lines and circular and elliptical arcs, all of which are referred to below as "Edges".

In general a single view of an engineering drawing will contain Multiple Edges and Loops. A complete circle or ellipse are the only ways that a Loop can occur for the limited set of Graphs that are considered. Tracing around a circle there is an Edge which starts at a Node and the same Edge finishes at this Node. Nevertheless a Node through which a complete circle passes (provided no other Edges terminate there) is said to be a 2-Node.

The centre of an arc is not a Node unless, by coincidence, an Edge terminates at that point. It will be referred to as a Centre. One type of Node which is rarely dimensioned is the arbitrary position where a complete circle begins and ends.

The modelling algorithms provided by SUPERMODEL convert the user's initial input of a surface into one or more ordered Circuits during the process of checking that the straight lines and circular arcs bound an area. There is also a check that the surface is non self-intersecting and then the direction of the Circuits is arranged so that material is always to the left of a walk around the Circuit. The representation is then a directed Graph, or Digraph, in which the direction of an Edge is of significance.

The non self-intersecting check guarantees that every Node within the Digraph is a 2-Node. The type of continuity which may exist at a Node is considered in Section 9.2.

4.4 Review of theoretical work on dimensioning

Remarkably, there does not appear to have been any published attempt to develop a theory of dimensioning until computers began to be used as aids to the process of geometric design. There is a minute literature on this subject and only one analysis which the author considers to be relevant to this project. This is due to Hillyard [27] and his work was used by Light [28] in the development of symbolic dimensioning systems at Massachusetts Institute of Technology.

Hillyard chose a wire-frame model to describe the shape

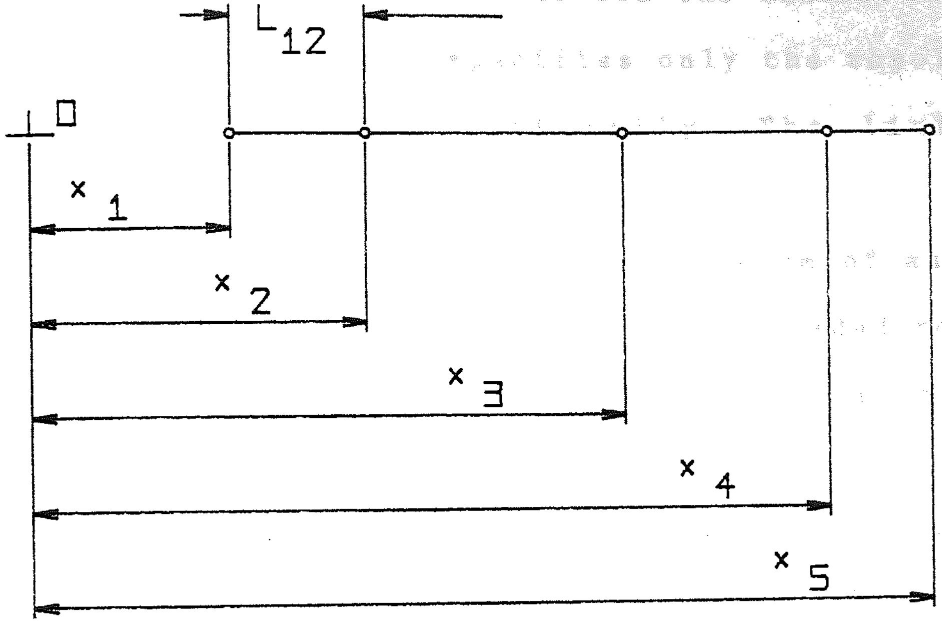


Fig. 4.13 A one-dimensional chain of Nodes

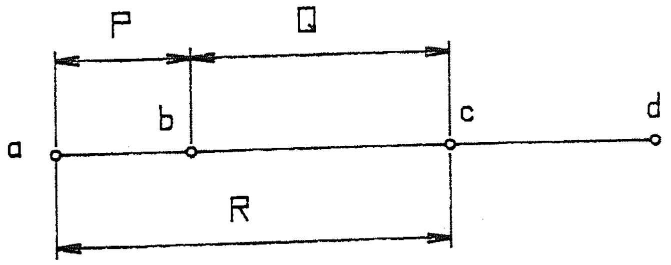


Fig. 4.14 Stiffening a chain

of an object. The wire-frame may be thought of as a linkage mechanism where the Edges correspond to variable length links and the Nodes are the centres of spherical joints. This model specifies only the topology of the object, at least initially. The links are of indeterminate length.

Hillyard defines the degree of freedom of an object as: "the number of possible variables needed to reduce it from a linkage into a structure." A constraint is any condition which diminishes the degree of freedom of the system. The imposition of a dimensional constraint on the model is analagous to the addition of a stiffener to a frame.

Hillyard proposes three types of "simple" stiffener. The first is dubbed a "strut", which fixes a distance. The second is a "web" which fixes an angle. The third is a "plate" which renders three or more Nodes coplanar. A composite stiffener is a combination of simple stiffeners, one of which is termed the primary stiffener, the others secondary stiffeners.

Hillyard's theory is well illustrated by the simple case of a one-dimensional chain of n Nodes. This is shown in Fig. 4.13. The first and last Nodes are joined once to Edges and the other Nodes each meet two Edges. Initially, the Nodes of the chain are free to move along the line. Each Node requires one constraint to fix its position and so a total of n constraints are required. The position of some point in the chain, such as its mean point must be fixed to prevent the entire chain moving as a rigid-body. The remaining n-1 constraints are imposed by adding an equal number of "struts" but, unless the "struts" are correctly positioned, the chain will not be stiff. In Fig. 4.14 section abc is overconstrained while cd is underconstrained even though the number of struts is equal to the number of Edges. The positions of the vertices are given by:

$$x_1 , x_2 , \ldots, x_n$$

A strut joining Node i to Node j has length L_{ij}, where

$$x_i - x_j = L_{ij}$$

If the permitted displacement of the positions of Nodes i and j is d_i and d_j respectively and the variation in the length of the strut joining them is u_{ij} then:

$$d_i - d_j = u_{ij} \qquad 4.3.1$$

The absolute value of u_{ij} must not exceed some Tolerance, T

The n-1 struts give rise to the same number of equations of the form 4.3.1 with unknowns $d_1, d_2, \ldots, d_n$. A further equation is obtained by fixing the mean point of the chain: the mean value of the variation in position of the nodes is zero

$$\frac{1}{n} \sum_{i=1}^{n} d_i = 0 \qquad 4.3.2$$

Equations 4.3.1 and 4.3.2 may be written in matrix form

$$\mathbf{R}\,\mathbf{d} = \mathbf{u} \qquad 4.3.3$$

where **R** is the rigidity matrix, **d** the displacement vector and **u** the variation vector.

Hillyard proposed to assess different dimensioning schemes by finding a single figure of merit which gives an overall indication of the sensitivity of the defined shape to small changes in dimensions. The problem is equivalent to finding a measure of how well-conditioned is the system of equations given by the rigidity matrix. The inverse **F** of the rigidity matrix, if one can be formed, may be used to determine the effect of variations in any dimension. For a one-dimensional chain the rigidity matrix may be written:

$$\mathbf{R} = \begin{bmatrix} 1/4 & 1/4 & 1/4 & 1/4 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & -1 & 0 \end{bmatrix}$$

By inverting this matrix it is possible to compute a "worst-case" indicator W, which is the maximum possible displacement at any Node when all the variations conspire together in their effect. W may readily be computed for two common, contrasting styles of dimensioning: datum dimensioning (Fig. 4.15) and size dimensioning (Fig. 4.16). For a chain of n Nodes and a

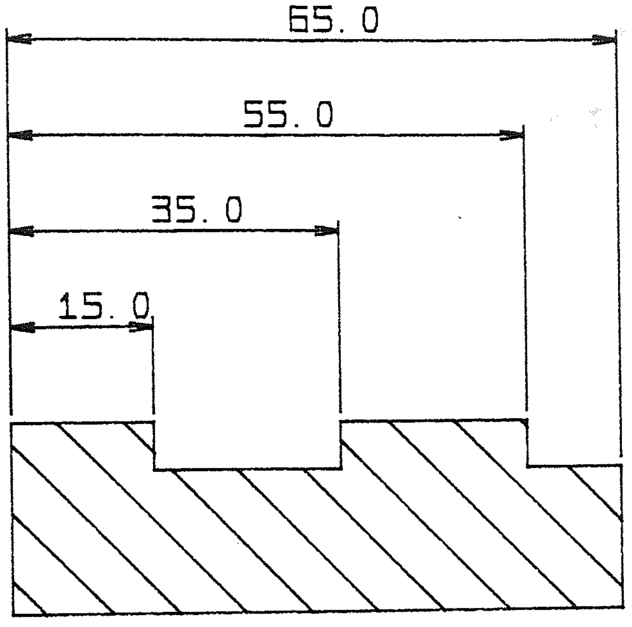


Fig. 4.15 Datum dimensioning

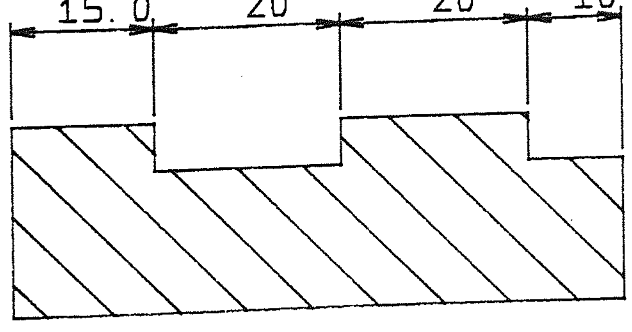


Fig. 4.16 Size dimensioning

tolerance of T on each dimension, the results are:

$$W_{datum} = T \quad \text{and}$$

$$W_{size} = 0.5(n + 1 - 2/n)T$$

If there are more than two Nodes, then datum dimensions give a more rigidly defined shape. Hillyard's theory explains how dimensions and tolerances act in engineering design and points the way towards a practical system of automatic dimensioning. Before this objective may be reached, however, it is essential to take account of some additional constraints on the process of dimensioning, such as material, method of manufacture, and function.

The author has chosen to address the question of the influence of function on a dimensioning scheme for two main reasons. Firstly, it is apparent from an examination of many manually-prepared detail drawings that the draughtsman is concerned with the function of the objects, and their geometrical accuracy is simply a means to this end. This is evidenced by some of the lengthy notes that appear on drawings to indicate what should happen when parts are assembled. Secondly, as shown in the following section, it is possible to generate a taxonomy of functions of surfaces which may be employed as the geometric model is built.

4.5 New Theory of Dimensioning

Hillyard [op cit] was primarily concerned with the interaction between a scheme of dimensioning and the effect that small variations in individual dimensions has on the total shape. One simple result, described in the last Section, suggests that datum dimensioning gives a more rigidly defined shape than size dimensioning. Although undeniable, the result is of limited practical use for there are many occasions when, because of the function of the specified surfaces, a draughtsman is constrained to indicate a size dimension.

The model for what follows is a boundary representation of the type used in SUPERMODEL, although the results may be extended to include more complex curved surfaces. The underlying hypothesis is that there is a taxonomy of surfaces from which the single function of a particular surface may be selected. These functions are:

Hard Interface

Soft Interface

Guide

Dam

Boundary

Signal

Each of these classes of surface is illustrated in Fig. 4.17 and is defined below.

THIRD ANGLE PROJECTION

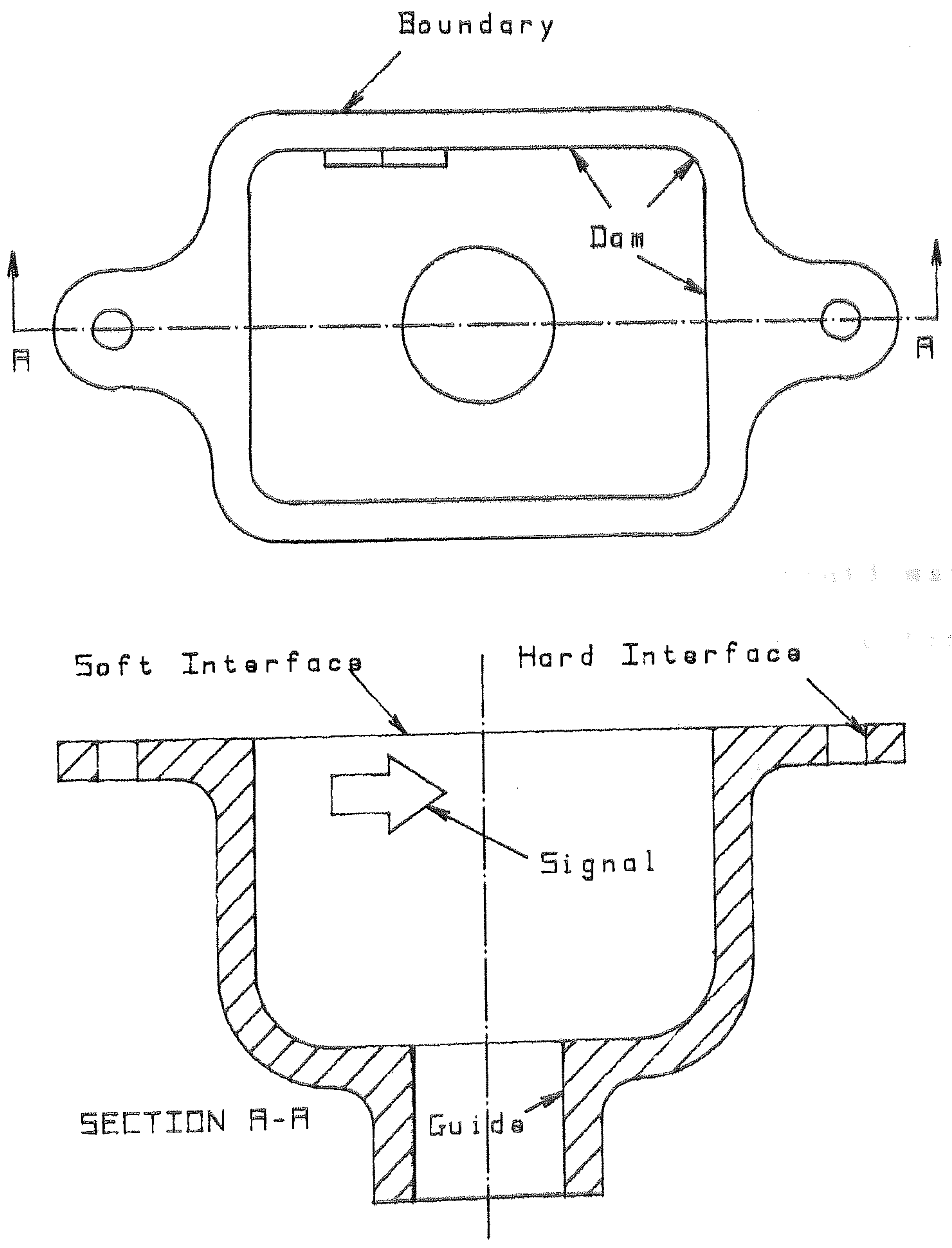


Fig. 4.17 Classification of surface by function

4.5.1 A surface is described as a Hard Interface if its function is to mate accurately with a geometrically complementary surface on another object.

4.5.2 A Soft Interface is a surface whose function is to contact a surface which may be selected from a range of geometries on other objects. Alternatively, the surface which contacts the Soft Interface may be compliant. These surfaces may belong to objects that are animate as well as inanimate. Thus, the surface of a handle is a Soft Interface, designed to suit a range of human hands.

4.5.3 A surface is descibed as a Guide if its function is to guide the flow of some fluid and its geometry has some effect on the fluid flow. One or more such surfaces may bound the channel along which a fluid may pass or the surface may simply react any dynamic forces from within the fluid.

4.5.4 A Dam is a surface which bounds a volume that is designed for the storage of a fluid. It may react static forces from within the fluid.

4.5.5 A surface is described as a Boundary if it fulfills no other function than that of delimiting the solid object. The position of such a surface may influence the strength and/or stiffness of a component.

4.5.6 A Signal is a surface whose function is visual communication with a person who is able to observe the object of which the surface is a part. Embossed or raised lettering has several Signal surfaces and an engraved logo falls into the same class. However, the raised dots of a language such as Braille, although constituting a signal, are properly classified as a Soft Interface, designed to contact a range of human hands.

Like all taxonomies, the above is open to criticism on the question of ease of classification. One might argue that a finned surface of an air-cooled engine is a Guide because air flows over it. The contrary argument is that it does not guide the air, in the way that an aerofoil does, and so it has no other function than to bound the deliberately large external area of the cylinder block. A flow-visualization technique might provide evidence for another argument: there is a reservoir of air trapped between fins, hence the surfaces in question are Dams.

Similar debates may arise over borderline cases when a distinction has to be made between Hard and Soft Interfaces. It is shown below that these decisions have a less sigificant effect on dimensioning schemes than an intuitive view might suggest.

Surfaces are classified in this manner in order to determine whether their geometry must be specific or may be implicit. As shown in Fig. 4.18, every Edge is common to two surfaces. As dimensions should not be duplicated, once an Edge, being regarded as part of the boundary of surface A, has been fixed in space, the same Edge, being regarded as part of surface B, requires no further

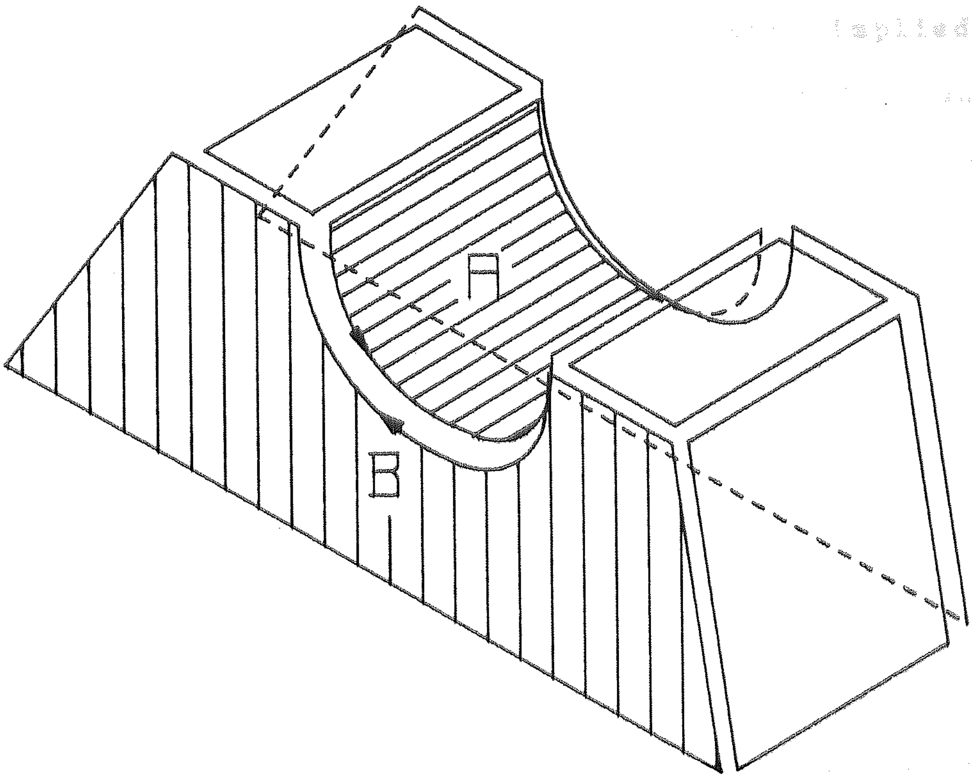


Fig. 4.18 Geometry at common edge

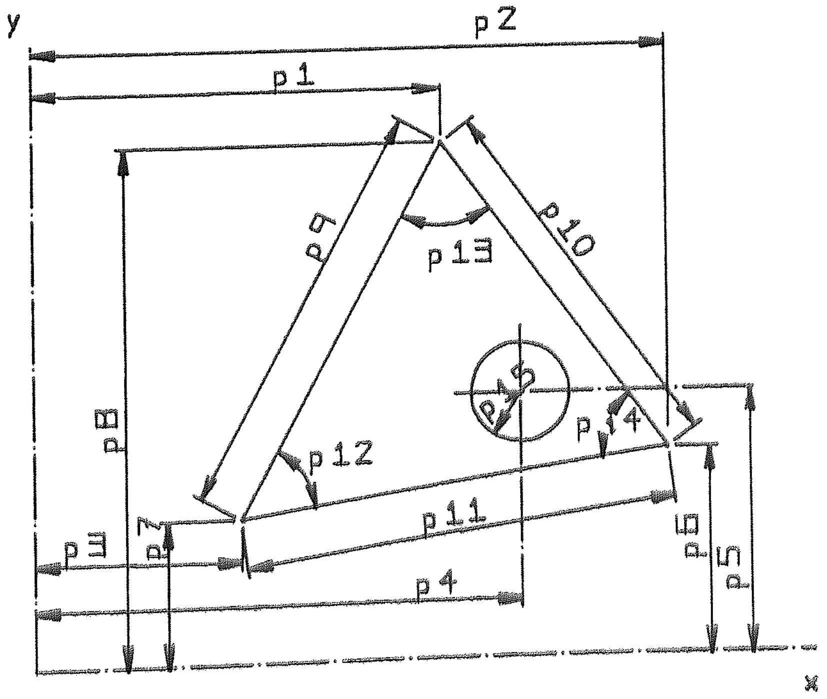


Fig. 4.19 Constraining Parameters

attention.

The purpose of the above classification is to determine which surfaces may have their geometry implied by that of the others. In what follows below it is assumed that the order of precedence for the classes of surface is as listed above. This implies that every Hard Interface must have its geometry (and the permitted variations therefrom) directly specified by the dimensioning scheme. When this has been done, the next priority is to remedy any deficiencies in the geometry of each Soft Interface. When the Soft Interfaces have been disposed of, the Guides should receive attention, etc. Finally, any surfaces classed as Signals are processed and they will suffer the greatest possible deviation from their ideal shape as a result of any permitted tolerances in the dimensions of previously processed surfaces.

The process of dimensioning a Hard Interface is one of selecting, from an infinite set of constraining parameters, a set which is both sufficient and relevant to the function of the surface. Fig. 4.19 illustrates the process for a simple triangular surface containing one circular hole. The infinite set of constraining parameters could include such impracticable quantities as the perimeter of the triangle or its second moment of area. In practice, a draughtsman will select from a finite set of usable parameters.

Each parameter has an associated tolerance which reflects the practical difficulties of manufacture and inspection. As most practical tolerancing schemes refer to some nominal value and unequal variations above and below it, it is convenient for what follows to refer to the maximum and minimum acceptable values of each parameter. For a line/arc graph that bounds an area, such as that shown in Fig. 4.19, the dimensioning problem may be stated thus:

Given a set of maximum values of r parameters:

$(u_1, u_2, u_3, \dots , u_r)$

and their respective minimum values:

$(v_1, v_2, v_3, \dots , v_r)$

select the valid subset of parameters which minimizes the deviation from the ideal shape.

A valid subset is one which is just sufficient to define the geometry. A line/arc graph consists of the following types of Edge:

a) straight lines, each of which requires four parameters,

b) complete circles, each of which requires three parameters, and

c) circular arcs, each of which requires five parameters.

Some of these parameters will be common to two adjacent Edges. A figure of merit which gives an overall indication of the sensitivity of the shape to the range of parameter values has practical value only if it takes account of the function of the particular surface. The selection of a sub-set of parameters is therefore constrained by:

a) The geometric complexity of the shape,

b) The precision of the measurement and/or manufacturing processes, and

c) The function of the surface.

One well-known result of this analysis is manifested in surveying. Land is surveyed by theodolite measurements of each angle between straight lines which bound the area of interest. The length of only one straight line is needed in order to establish the geometry of the polygon. As angles can be measured with greater precision than distances, the net effect yields a polygon which is geometrically similar to the true one. In the language of computer graphics: the measured polygon is a scale transformation of the true one.

The concept of transformation is theoretically attractive for dimensioning schemes. The transformations of translation, rotation and scale each result in a geometry that preserves both angles and tangency. Suppose that, for a hard interface, the following axioms were applied:

a) Parallel lines should remain parallel.

b) Perpendicular lines should remain perpendicular.

c) A Node at which there is no change of gradient should retain this property.

d) Lines normal to arcs should remain normal.

All of the above could be ensured simply by accepting only transformations of the basic shape. However, there are practical difficulties which preclude such a constraint. For example, if two circular holes in a hard interface are to mate with two pegs, then the distance between centres is of prime importance. Translation and rotation transformations would not affect this relationship but a scale transformation of the shape would not be acceptable because this critical distance could vary.

A modified form of constraint might be to allow transformations of Circuits, independently of the surface of which they are part. A hard interface may be regarded as a set of Circuits, each of which may undergo translation, rotation and scaling. This would overcome some of the practical difficulties with mating parts but leads to an intractable theoretical problem:

Given a set of maximum values of r parameters:

$(u_1, u_2, u_3, \ldots, u_r)$

and their respective minimum values:

$(v_1, v_2, v_3, \ldots, v_r)$

select the valid subset of parameters which yields only transformations of each Circuit and minimizes these transformations.

The conclusion reached by the author was that the selection of parameters for dimensioning was not a suitable subject for decision-making algorithms to handle. Decisions about these parameters are best left to the person who builds the geometric model. This is not to imply that the user should be required to dimension the views that are drawn. It does imply that the steps by which a surface is input to the geometric model are of as much significance as the final result. If the user is not constrained by the modelling software and may therefore define the geometry of a surface in his own way, then a record of the process (and not simply the end-result) is found to be highly relevant. This point is taken up again in Chapter 9.

CHAPTER 5

TECHNIQUES OF DECISION-MAKING

Having examined the apparatus with which models are built in Chapter 3 and the nature of a working engineering drawing in Chapter 4, both the input and the output for this project have been treated. It is now necessary to focus attention on some general principles of decision-making in order to investigate the tools and techniques that may be used for the design of decision-making algorithms. The actual process of decision-making will first be analysed. The rest of this Chapter describes some established mathematical tools and indicates ways in which they may be made to serve the objectives of this work.

5.1 Stages and Terminology in Decision-making

A general introduction to Decision-making by Adams, Gagg and Tayar [29] lists the following five stages in the process:

a) Purpose. It is necessary to decide on the general aim that is to be achieved and then to specify it in detail.

b) Information. This involves an investigation of what information is thought to be necessary, what information is already known and what information is yet to be obtained. It is then necessary to consider where the information can be found and what information is unlikely to be available at the time a decision has to be made. This stage also involves the organisation of the information that is gathered.

c) Evidence and Argument. Some of the Information gathered during the previous stage may not be relevant to the decision that has to be made. That which is relevant is termed Evidence and is used in logical Argument when possible solutions are examined.

d) Solution. Viable solutions to the problem have to be discovered in various ways. If there are several possible solutions then the implications of choosing each one must be examined in more detail. It is also important to decide whether any or all of the solutions will lead to other decisions if they are implemented. The performance of the solution, when implemented, should be monitored and the solution may need to be reconsidered.

e) Implementation. This is the process of putting into operation the various processes implied by the chosen solution.

The above stages are referred to in later Chapters whenever the design of a particular algorithm is discussed. The purpose of the overall project has been specified in Chapter 1 and the purpose of each separate algorithm is given before it is treated. The initial information is, of course, the geometric model and some essential additional information (the function of each surface) was discussed in Section 4.5. Further information may be required for some of the decisions that have to be made. Normally, this information may be obtained by processing the geometric model but the cost of obtaining this information, in relation to its value as a decision-making aid, is, at times, prohibitive. On such occasions, the information that is not available for input to the decision-making algorithm is specified.

The term "evidence" is used during the discussion of algorithm design to indicate the particular information which is considered to be relevant to the task in hand. The weight to be attributed to this evidence is arrived at by logical argument but also depends upon the mathematical probability of what the evidence implies. The word "strategy" is preferred to "solution" in what follows because it is well-established in the terminology of Decision Theory, which is treated in section 5.3. Implementation is a term that is widely used in connection with the conversion of an algorithm (a concept) into the actual program which is mounted on a particular computer.

5.2 Potential Tools

Dixon [30] states that there are several branches of engineering that may be called the study of decision making. He includes such branches of knowledge as Numerical Optimization, Probability, Statistics, Decision Theory and Game Theory. Probability and Statistics are such well-known and well-proven tools for engineering applications that their employment for the purposes of this project may be taken as read. References to the other branches of knowledge mentioned above appear less frequently in the literature of engineering design and so each is briefly introduced before considering some of them in their separate sections.

One of the principal concerns of Decision Theory is with the development of analytical techniques for guiding the choice of a single course of action from among a series of alternatives, in order to accomplish a designated goal. It is an area where currently a great deal of work is being undertaken and, as a result, an increasingly sophisticated range of tools is becoming available to enable the decision-maker to use the information at his disposal in the most efficient way.

Much of the literature of Decision Theory is business orientated but it is shown in Section 5.3 that many of the ideas can be applied directly to this project. The space devoted to treating this subject reflects the author's assessment of its significance.

Optimization may be defined as computing the values of certain constrained parameters which maximize (or minimize) the value of some "objective fuction", such as cost, heat transfer per unit area, etc. It was recognized early in this project that mathematical optimization could be a useful tool. However, the nature of the problem is such that the specification of an overall objective function is impracticable. Like most engineering projects, it does not involve just one major decision but a whole series of decisions: the early ones are broad and concern questions of principle, the later ones on questions of detail.

Numerical optimization may provide the means of making a decision when a situation presents itself in which a limited number of independent parameters are seen to be affecting some desirable or undesirable feature. If the number of parameters is very great or if the feature is not easily identified, or both, then optimization techniques are not suitable tools. We shall return to this point when numerical optimization is disussed in Section 5.4

Game Theory and Decision Theory are closely related. The decision-maker in a Decision Theory situation has a strategy which he implements, and then nature takes its course. In Game Theory, an intelligent opponent, who also has a strategy, is included in the situation. It may be argued that the Decision Theory model is effectively a "game against nature", with nature being cast in the role of opponent. Like most emerging areas of knowledge the boundaries are fuzzy, but many of the technical terms and principles of Game Theory appear in the literature of Decision Theory and vice-versa and, therefore, no special section has been devoted to the former.

5.3 Decision Theory

Kmietowicz and Pearman [31] preface their book by indicating that its starting point was the type of decision making problem faced by businessmen or managers. They then suggest that the methods that they describe may be extended to other decision situations, a view which the author would endorse. The introduction to Decision Theory and the definitions of some of the non-standard expressions which abound in this subject are particularly valuable features of the book cited.

An essential feature of a decision situation is the existence of alternative courses of action (or strategies), which are denoted by S_1 , S_2 , S_3 ,..., S_m There are always conditions which are outside the control of the decision-maker; they are referred to as "states of nature" and denoted by N_1 , N_2 , N_3 ,..., N_n

These states of nature are often continuous but such situations are handled by dividing them into arbitrary but convenient discrete states.

For each given combination of strategy and state of nature there is a consequence. This may be described in terms of monetary gain but may include many other items such as time, loss of resources, etc. All these factors are embraced by a single term called the "payoff". The payoff of strategy S_i given state of nature N_j is designated as X_{ij}. Thus the essential information of a decision problem may be conveniently summarised in a payoff matrix, as shown in Table 5.1.

There are three generally recognised approaches to decision making. The first approach assumes that the decision maker is working in conditions of complete uncertainty, i.e. no information about the probabilities of the states of nature occuring is available to him. This situation is referred to as "decision making under uncertainty". The second approach takes the view that probabilities on the states of nature can be specified uniquely. This situation is termed "decision making under risk". The third approach assumes that some information is available about the probabilities of the states of nature, but that it is not comprehensive enough to enable exact specification of the probabilities. This is referred to as "decision making under conditions of incomplete knowledge".

Many of the mathematical techniques of decision making have been developed with reference to activities such as capital investment, medical care, agriculture, etc. What are the "states of nature" with reference to geometic modelling? Quite simply, the solid object that

Strategy	State of Nature					
	N_1	N_2	...	N_j	...	N_n
S_1	X_{11}	X_{12}	...	X_{1j}	...	X_{1n}
S_2	X_{21}	X_{22}	...	X_{2j}	...	X_{2n}
.	.	.		.		.
.	.	.		.		.
S_i	X_{i1}	X_{i2}	...	X_{ij}	...	X_{in}
.	.	.		.		.
.	.	.		.		.
S_m	X_{m1}	X_{m2}	...	X_{mj}	...	X_{mn}

Table 5.1 A Payoff Matrix

is modelled is a "state of nature". This may appear to contradict the basic assumption that a state of nature is uncertain and, at best, may only be expressed as a probability. By definition, a geometic model is certain, unambiguous and complete.

Nevertheless, the situation does correspond to that descibed by decision-making theorists. A decision-making algorithm can only select from a finite number of options when processing a geometric model with some specified objective. The number of geometric models that may be presented to the decision-making algorithm is infinite but it will be shown in later Chapters that it is generally possible to devise a taxonomy of models: a set of "classes of model" which will be denoted by:

$$M_1 \ , \ M_2 \ , \ M_3 \ ,\ldots, \ M_n \ .$$

For each combination of strategy and class of model there will be some consequence which is yet to be defined.

For example, a decision is required on whether to draw an exterior view or a sectional view in some specified direction. At the lowest level of sophistication one would have just two classes of model: simple (those whose internal complexity did not exceed some quantifiable amount), and complex (all others). The consequences are attractive for the combinations (exterior view, simple) and (sectional view, complex).

It has been implied, thus far, that entries in the payoff (consequence) matrix are specified in terms of money, time, or some other objective scale common to all decision-makers. There can be no such objective scale for the decisions which are the subject of this project and, indeed, the same is often true of business, medical and sociological decisions. The standard technique of decision-theory is to convert, say, a monetary payoff into some "utility" which corresponds more closely to the decision-maker's subjective valuation. For example, when the marginal utility, U, of money decreases as the amount, M, increases, it would be reasonable to assume that $U = f(\log M)$

The concept of utility appears (often with a different name) in the literature of Computer-aided Design. Mischke [32] defines a "figure of merit" as a number whose magnitude is an index to the merit or desirability of a solution to a problem. He comments that, when a single factor (such as cost) dominates a solution the construction of a figure of merit function is simple, but when a multiplicity of interdependent factors vie for dominance, the fabrication of a meaningful figure of merit becomes exceedingly difficult.

The utility associated with a consequence X_{ij} will now be denoted $U(X_{ij})$. It is advantageous to have coherence when comparing utilities and therefore the existence of a pair of reference consequences is postulated. X_0 is the least desirable consequence and X_1 the most desirable. By definition, a consequence has a utility of zero if it is the least desirable outcome, but a utility of 1 if it is the most desirable result.

Thus $U(X_{ij}) = (X_{ij} - X_0)/(X_1 - X_0)$ and hence, utility is measured on an interval scale: unique up to a positive linear transformation. The choice of zero and unity for the range of the scale is a useful convention because the least and most desirable outcomes are easily recognised in any utility matrix.

If the class of model is known then the decision making process is reduced to the selection, from the appropriate column in the utility matrix, of the strategy with the greatest utility. If nothing, not even the probability that a model belongs to a particular class, is known then the situation is referred to as "decision making under uncertainty". In the literature there are various principles for decision making under such conditions. Three of these principles are particularly relevant to this project and will be considered in detail.

The MAXIMIN criterion suggests that the decision maker should examine only the minimum utilities of strategies and select the strategy with the largest of these, hence the name.

Using the above example of exterior view versus sectional view selection, the minimum utility, when opting for an exterior view, is the combination (exterior view, complex). The minimum utility when opting for a sectional view is associated with (sectional view, simple) and this is the worst possible outcome. Drawing an exterior view of a complex object satisfies the MAXIMIN criterion and is very attractive if one wishes to proceed with caution.

The MAXIMAX criterion adopts just the opposite viewpoint. It advises the decision maker to examine only maximum utilities of strategies and to select the strategy with the largest of these. If the decision maker was particularly concerned about clarity then he would probably decide to draw a sectional view because the utility of (sectional, complex) is greater than that of (exterior, simple).

The principle of MINIMAX REGRET was originally advanced by Savage [33]. He suggested that the utility matrix should be transformed into a "regret" matrix in which each element represents the positive difference between a specific utility and the highest utility possible under the corresponding class of model. The new numbers reflect the decision-maker's degree of regret - once the true class of model is known - for not having chosen the strategy that yields the most desirable consequence for that class.

Suppose that the decision maker accepts an equal interval between utilities and that he is primarily concerned with clarity. The utilities are:

U(exterior, simple) = 1

U(sectional, simple) = 0

U(exterior, complex) = 1/3

U(sectional, complex) = 2/3

If the class of model is "simple" then the regret at drawing an exterior view is zero and the regret at drawing a sectional view is 1. If the class of model is "complex" then the regret at drawing an exterior view is 1/3 against zero regret for drawing a sectional view. The maximum regret if the model is "simple" is 1 and the maximum regret if the model is "complex" is 1/3 and, hence the minimax regret is for drawing an exterior view of a "complex" model.

Colman [34] notes that the minimax regret principle may, in some instances, violate a condition of rationality known as the "independence of irrelevant alternatives". At its simplest, if a decision maker has selected strategy A in preference to the only other option, B, it would be irrational of him to change from A to B if he was made aware of a third strategy, C. The existence of C is irrelevant to his preference of A to B. Unfortunately, the minimax regret principle can lead to decisions of this nature when the third strategy has a utility that is higher than one that was available before. It is shown in later Chapters that, when decisions are made under "uncertainty", the minimax regret principle can be used, provided a check is made on irrelevant alternatives.

For the purposes of this project, very few decisions need to be made in conditions of complete "uncertainty". It is usually the case that the probability of membership of each class of model can be readily determined but the computational cost of classifying the model with certainty is unacceptably high. In such a case it is necessary to select the strategy which is most likely to produce desirable consequences. The theory developed by von Neumann and Morgenstern [35] suggests one method by which this may be done.

If the probability of occurrence of the jth class of model is P_j and there are n classes of model then the utility of the ith strategy is given by:

$$U(S_i) = \sum_{j=1}^{n} P_j U(X_{ij}) \qquad 5.1$$

In numerical terms, let the probability of a "simple" model be 0.7 against 0.3 for a "complex" model. Let the utilities of the various combinations be as stated above. Hence, using equation 5.1, the utility of the strategy which calls for drawing an exterior view is:

$$U(S_1) = 0.7x1.0 + 0.3x1/3 = 0.8$$

and the utility of the strategy which calls for drawing a sectional view is:

$$U(S_2) = 0.7x0.0 + 0.3x2/3 = 0.2$$

Thus, the von Neumann-Morgenstern theory suggests that, given the stated probabilities, the sensible strategy is to draw an exterior view. Of course, the theory can only be applied if it is possible to assign numbers to the probabilities and the utilities. In practice, it might be possible to determine utilities more accurately by assessing the clarity of a view in relation to the cost of drawing it, but the probabilities could only be found empirically. However, for decision-making, the absolute values of the utilities are not required; their relative magnitudes will suffice. If the expected utility of the first strategy exceeds or equals that of the second then the first strategy is said to be statistically dominant. This point is further investigated when Fishburn's theorem is treated below.

So far, all that has been demonstrated is that it may be possible to adapt the concepts and terminology of decision-theory to the purposes of this project. It has not been demonstrated that, whenever a decision has to be made, there will be a finite number of classes of model from which to construct a "consequence" matrix, but the author judged it to be a reasonable assumption from which to proceed. The next question which must be addressed concerns the exact nature of the conditions under which the decision is to be made.

If the geometric modelling software places few constraints on the models that can be generated then there will be no initial information about the probability that a model belongs to a particular class.

It might appear that all that is known is that the probability is 1 that a model generated using the SUPERMODEL package belongs to the class of objects bounded by flat and/or conical surfaces. In order to make sensible decisions at each stage at least two classes of model will need to be defined.

In the author's view, there is some initial information about the probable complexity of the model that is to be processed. In the terminology of statistics a geometric model is an isolated event in a continuum of 3-space. A model defined by using SUPERMODEL may have from three to thirty eight surfaces. It is not meaningful to ask a question such as: how many models do not have six surfaces? Therefore, it is unlikely that the Binomial distribution would provide usable information on the spectrum of models that need to be processed. The Poisson distribution was employed instead as it often provides a useful guide to probabilities of events of this type, provided the average number of occurrences of the event is known.

A shape with less than three surfaces will not be accepted by the modelling software so a "base" model may be defined as one with one conical surface and two circular flat surfaces. Model "complexity" will be defined as the number of additional surfaces above the "base" level. Quite simple objects typically have between ten and twenty surfaces and occur relatively

Total number of surfaces in model (model "complexity" + 3)	Probability
3	0.000000002
4	0.000000041
5	0.000000412
6	0.00000275
7	0.0000137
8	0.000055
9	0.000183
10	0.000523
11	0.00131
12	0.00291
13	0.00582
14	0.0106
15	0.0176
16	0.0271
17	0.0387
18	0.0516
19	0.0646
20	0.0760
21	0.0844
22	0.0888
23	0.0888
24	0.0846
25	0.0769
26	0.0669
27	0.0557
28	0.0449
29	0.0343
30	0.0254
31	0.0181
32	0.0125
33	0.0083
34	0.0054
35	0.0034
36	0.0020
37	0.0012
38	0.00068
39	0.00038
40	0.00021

Table 5.2 Poisson distribution of model "complexity"

frequently. Complicated objects occur less frequently although they consist of many more surfaces. It will therefore be assumed that the "average model complexity" is a convenient value of 20 above the "base" level, or 23 surfaces in all. All that this implies is that the decision-making algorithms are designed to cater for this average complexity.

The Poisson distribution for model complexity, based upon an average complexity of 20 surfaces, is shown in Table 5.2. The first comment that may be made is that the distribution does not contain any great surprises. The probability of being required to process a "base" model is so small that it need receive no special attention by the algorithms. Considering only those models that have probabilities in excess of 1/100th of that with the greatest probability would limit the range to those with between eleven and thirty-seven surfaces. This greatly assists the design of decision-making algorithms as it usefully reduces the number of strategies that need be devised.

If the above is accepted as a working hypothesis then some information is available about the probabilities of the classes of model. Although this information is useful it is certainly incomplete. Does the situation correspond to that described as "decision making under conditions of incomplete knowledge"?

It is convenient so to classify the situation, for two main reasons. Firstly, the ultimate objective of the decision-making algorithms descibed in this thesis is the production of a dimensioned engineering drawing, for which there is no objective measure of quality. The only realistic approach has to be based upon probabilities, e.g. one might conclude that the probabably of a sectional view being correctly interpreted is greater than for an exterior view of a particular model. Secondly, the computational cost of acquiring some data will, on occasion, be unacceptably high and so, again, the decision has to be based upon incomplete knowledge.

This immediately poses the question: what is the trade-off between data acquisition by processing the geometric model and the quality of the decision that is made?

Lindley [36] observes that a natural reaction of anyone having to make a decision under uncertainty is to try to remove the uncertainty by finding out the true state of affairs. We may be reasonably confident that the model has at least nine surfaces but how many does it have?
In the language and notation developed above, one way of selecting a strategy from the set $(S_1, S_2, \ldots, S_m)$ when one of the classes of model $(M_1, M_2, \ldots, M_n)$ is presented is to find out which M has been presented.

If this is done and the true situation found to be M_k then the decision-maker need only consider the combinations $X_{1k}, X_{2k}, \ldots, X_{mk}$ and select the one with the highest utility. In situations where the states of nature refer to the future but the decision has to be taken in the present there is, of course, no means of ascertaining the correct event. The possibility that the determination of the true situation is prohibitively expensive, because of computing time, has already been raised.

It is shown in Chapter 6 that it is possible to compute all the relevant data for a decision on which of two opposite viewing directions to select. However, the cost in either money or time taken to carry out such computations is quite unacceptable. Therefore, it is necessary to settle for a situation where, instead of being certain about the true state of affairs, one knows instead that there is a high probability that a particular class of model has been presented.

Suppose that a decision-maker has, initially, probabilities for each of the possible classes of model and these are denoted by $p(M_1)$, $p(M_2)$,..., $p(M_n)$. If he obtains complete information, one of the probabilities will go to 1 and the remainder will go to zero. Partial information will produce a less marked change. In the notation of conditional probability, if I denotes the partial information, the revised values will be:
$p(M_1| I)$, $p(M_2| I)$,..., $p(M_n| I)$. A question of vital importance for the design of decision-making algorithms is: how are the probabilities $p(M_j)$, available before

the information is obtained, related to the $p(M_j| I)$ available after it has been computed ?

All these values must satisfy the basic rules of probability and be coherent one with another. The coherence is guaranteed by using a theorem which Lindley [op cit] ascribes to Bayes, an 18th century English cleric. Suppose that K denotes the initial knowledge and that there are n uncertain classes of model with probabilities, given K, $p(M_1| K)$, $p(M_2 |K)$,..., $p(M_n |K)$. Let I denote additional information relevant to the uncertain classes of model. Bayes' theorem then states:

$$p(M_j| I \text{ and } K) \propto p(M_j| K)p(I| M_j \text{ and } K) \qquad 5.1$$

In order to apply Bayes' theorem to the design of an algorithm to process a geometric model, it may be useful to restate 5.1 in words:

The probability that a model belongs within a particular class, given initial knowledge K and additional information I, is proportional to the probability that it belongs within that class, given only the initial knowledge, multiplied by a "likelihood" term.

The "likelihood" term is the probability of the additional information, given membership of the jth class and the initial knowledge. In other words, it is the likelihood of membership of the jth class , given the initial knowledge and additional information.

The example used above has two classes of model: simple,

M_1 and complex, M_2. In order to make use of Bayes' theory the two classes must be precisely defined. Let membership of the set of simple models be restricted to those where at least 2/3 of the surfaces may be seen by suitably orientating the model. Given the initial knowledge, the probability of membership of either class is 1/2, i.e.:

$$p(M_1 | K) = 0.5 \qquad \text{and} \qquad p(M_2 | K) = 0.5$$

Additional information is required before a decision is made on whether to draw an exterior or a sectional view. Suppose that one surface is examined and found to be visible. If the model is simple the probability of this happening is at least 2/3; if complex, less than 1/3. Provided the number of surfaces is large, successive examination of different surfaces does not affect the probabilities and may be regarded as additional information, I. The multiplication law of probabilities will apply and it will be possible to compute a value proportional to the probability that the model is simple or complex, given I:

$$p(M_1 | I \text{ and } K) \propto 0.5p(I | M_1 \text{ and } K) \qquad 5.2$$

$$p(M_2 | I \text{ and } K) \propto 0.5p(I | M_2 \text{ and } K) \qquad 5.3$$

but $$p(M_1 | I \text{ and } K) + p(M_2 | I \text{ and } K) = 1 \qquad 5.4$$

Combining 5.2, 5.3 and 5.4 it is possible to compute the updated probability that the model is simple. The significance of this result is that it quantifies the value of additional information. The processing of geometric models is expensive and Bayes' theorem provides guidance on how sampling from the total number of model-surfaces will affect the confidence with which a model may be placed in a certain class.

In later Chapters it is frequently found that a decision needs to be made although something less than a precise estimation of probabilities of classes of model is available. Fishburn [37] assumed that the decision maker has sufficient information about the environment of his problem to be able to rank the probabilities. That implies, in the terms of this project, an ability to state for the n classes of model that are proposed that $P_1 > P_2 > \dots > P_n$.

Fishburn considered two strategies and proved the following theorem:

If $P_1 > P_2 > \dots > P_n$, then $U(S_1) > U(S_2)$ if

$$\sum_{k=1}^{j} U(X_{1k}) > \sum_{k=1}^{j} U(X_{2k}) \qquad 5.3.1$$

The principal drawback of the theorem is that the inequality 5.3.1 is a very stringent requirement that rarely occurs in practice.

Kmietowicz and Pearman [op cit] give an alternative proof of Fishburn's theorem which, by substitution, leads to a more practically useful decision-making rule. The substitutions are

$$Q_j = P_j - P_{j+1} \qquad (j = 1 \ldots (n-1)) \qquad 5.3.2$$

$$Q_n = P_n \qquad \text{(since, by implication } P_{n+1} = 0) \qquad 5.3.3$$

$$Y_{ij} = \sum_{k=1}^{j} U(X_{ik}) \qquad (i = 1,2; \quad j = 1 \ldots n) \; 5.3.4$$

Cannon and Kmietowicz [38] show that it is possible to derive expressions for the minimum and maximum expected utilities of any strategy, where the classes of model are subject to the above probability ranking constraint.

The formalised problem is:

$$\text{Maximize or minimize } U(S) = \sum_{j=1}^{n} P_j U(X_j)$$

$$\text{subject to} \qquad \sum_{j=1}^{n} P_j = 1$$

$$P_j - P_{j+1} > 0 \qquad (j = 1 \ldots (n-1))$$

$$P_j > 0 \qquad (j = 1 \ldots n)$$

The determination of these extreme values is greatly simplified by the substitution of equations 5.3.2, 5.3.3, and 5.3.4 into the above, when the problem becomes:

$$\text{Maximize or minimize } U(S) = \sum_{j=1}^{n} Q_j U(Y_j) \qquad 5.3.5$$

$$\text{subject to} \quad \sum_{j=1}^{n} jQ_j = 1 \qquad 5.3.6$$

$$Q_j > 0 \quad (j = 1...n) \qquad 5.3.7$$

A solution to this problem may be found by making use of the theory of numerical optimization. Such techniques are discussed in Section 5.4 before returning to the implications of the above optimization.

5.4 Numerical Optimization

An optimization problem involves minimizing a function (called the objective function) of several variables, possibly subject to restrictions on the values of the variables defined by a set of constraint functions.

For reasons of computational efficiency, optimization problems are classified into particular categories, where each category is defined by the properties of the objective and constraint functions. The following categories are given for completeness although only two are relevant for this project.

5.4.1 Unconstrained Minimization.

In such problems there are no constraints on the variables. The problem can be stated mathematically as follows:

$$\underset{x}{\text{minimize}}\ (F(x)),$$

where $x \in E^n$, that is, $x = (x_1, x_2, \ldots, x_n)^T$

5.4.2 Nonlinear Least-Squares Problems

Special consideration is given to the problem for which the function to be minimized can be expressed as a sum of squared functions.

5.4.3 Minimization subject to Bounds on the variables

These problems differ from the unconstrained problem in that at least one of the variables is subject to a simple restriction on its value but no constraints of a more general form are present.

5.4.4 Minimization subject to linear constraints

A general linear constraint is defined as a constraint function that is linear in more than one of the variables. The various types of linear constraint are reflected in the following mathematical statement of the problem.

$$\underset{x}{\text{minimize}}\ (F(x)), \quad x \in E^n$$

subject to equality constraints, inequality constraints, range constraints and bound constraints.

If F(x) is a linear function, the linearly-constrained problem is termed a linear-programming problem; if F(x) is a quadratic function, the problem is termed a quadratic-programming problem.

5.4.5 Minimization subject to nonlinear constraints

These problems are distinguished from 5.4.4 in that at least one constraint function is nonlinear.

The optimization problem expressed in 5.3.5 is a linear programming problem with only one functional constraint. If a finite optimal solution exists, it will correspond to the case of only one of the decision variables being non-zero. Clearly, if only one Q_j is non-zero then, applying equality constraint 5.3.6, is must take the value 1/j.

The objective function will therefore be maximized when Y_j/j is maximized and minimized when Y_j/j is minimized.

Hence, the extreme expected utilities of any strategy, given a ranking of the probabilities of the classes of model, may be found by computing the n partial averages

$$\frac{1}{j} Y_j = \frac{1}{j} \sum_{k=1}^{j} U(X_k) = \overline{U}(X_j)$$

The largest such partial average will be the maximum expected utility and the smallest will be the minimum expected utility. The implications of this result, for this project, may best be examined by means of another numerical example.

Let there be four classes of model for internal complexity, arranged in decreasing order of probability:

M_1	Internal details all in one normal plane
M_2	No internal details
M_3	Internal details in two normal planes
M_4	All other cases - complex internal details

Let the strategies be as above, i.e. exterior view or single-plane sectional view. The utility matrix is shown in Table 5.3 and is based on the assumption that the worst outcome is to draw a sectional view of an object with no internal details, whereas the best possible outcome is to draw an exterior view of such an object.

The partial averages for each strategy are shown in Table 5.4. For the first strategy the maximum utility is 0.85 and the minimum is 0.50. For the second strategy the maximum utility is 0.90 and the minimum is 0.45

If there were no information about the probabilities of each class of model then the maximax criterion would indicate the drawing of an exterior view to be the better strategy. Using ranked probabilities changes the attractiveness of the second strategy: it is now indicated by the maximax criterion. In general, there will be decision-making situations where the attractiveness of a strategy appears different by using ranked probabilities, whatever criterion is adopted.

Strategy	Class of model 1	2	3	4
S_1 Draw exterior view	0.7	1.0	0.2	0.1
S_2 Draw sectional view on single plane	0.9	0	0.8	0.5

Table 5.3 Utility Matrix with ranked probabilities

$\overline{U}(X_{11}) = (1/1)(0.7) = 0.70$

$\overline{U}(X_{12}) = (1/2)(0.7 + 1.0) = 0.85$

$\overline{U}(X_{13}) = (1/3)(0.7 + 1.0 + 0.2) = 0.63$

$\overline{U}(X_{14}) = (1/4)(0.7 + 1.0 + 0.2 + 0.1) = 0.50$

$\overline{U}(X_{21}) = (1/1)(0.9) = 0.90$

$\overline{U}(X_{22}) = (1/2)(0.9 + 0) = 0.45$

$\overline{U}(X_{23}) = (1/3)(0.9 + 0 + 0.8) = 0.57$

$\overline{U}(X_{24}) = (1/4)(0.9 + 0 + 0.8 + 0.5) = 0.55$

Table 5.4 Computation of Partial averages

In later Chapters, it is shown that the computational cost of ranking probabilities is slight when compared with more precise computations of probabilities. As the cost of computing partial averages is trivial, the decision-making algorithm can be improved by adopting the techniques of "incomplete knowledge".

Some techniques which may be of assistance to a decision-maker have been examined and those of particular utility to the present work have been identified. The entire groundwork for the design of decision-making algorithms for producing a dimensioned engineering drawing has now been laid. In Chapter 6, the first such algorithm is examined: its function is to judge which combination of exterior views of the model will be most suitable for dimensioning.

CHAPTER 6

SELECTION OF EXTERIOR VIEWS FOR DIMENSIONING

6.1 Nature of the problem

A draughtsman using conventional techniques has to make a decision about which views to draw in orthographic projection. He should consider which views will most clearly represent the solid object, whether they have details visible or hidden, and whether they show surfaces in their true shape or projected onto some non-parallel projection plane. He will select sectional views which give a clear picture of the internal details of the object, and auxiliary views which assist the reader to comprehend the nature of any inclined surfaces.

Pictorial views of the object are not widely used in British drawing offices although they can be helpful in the understanding of complex shapes. A draughtsman may well have to moderate his desire for clarity because of the high cost of manual draughting. The scope of a manually-produced engineering drawing represents a compromise between labour-cost and lead-times on the one side and ease of interpretation on the other.

When producing a dimensioned drawing from a geometric

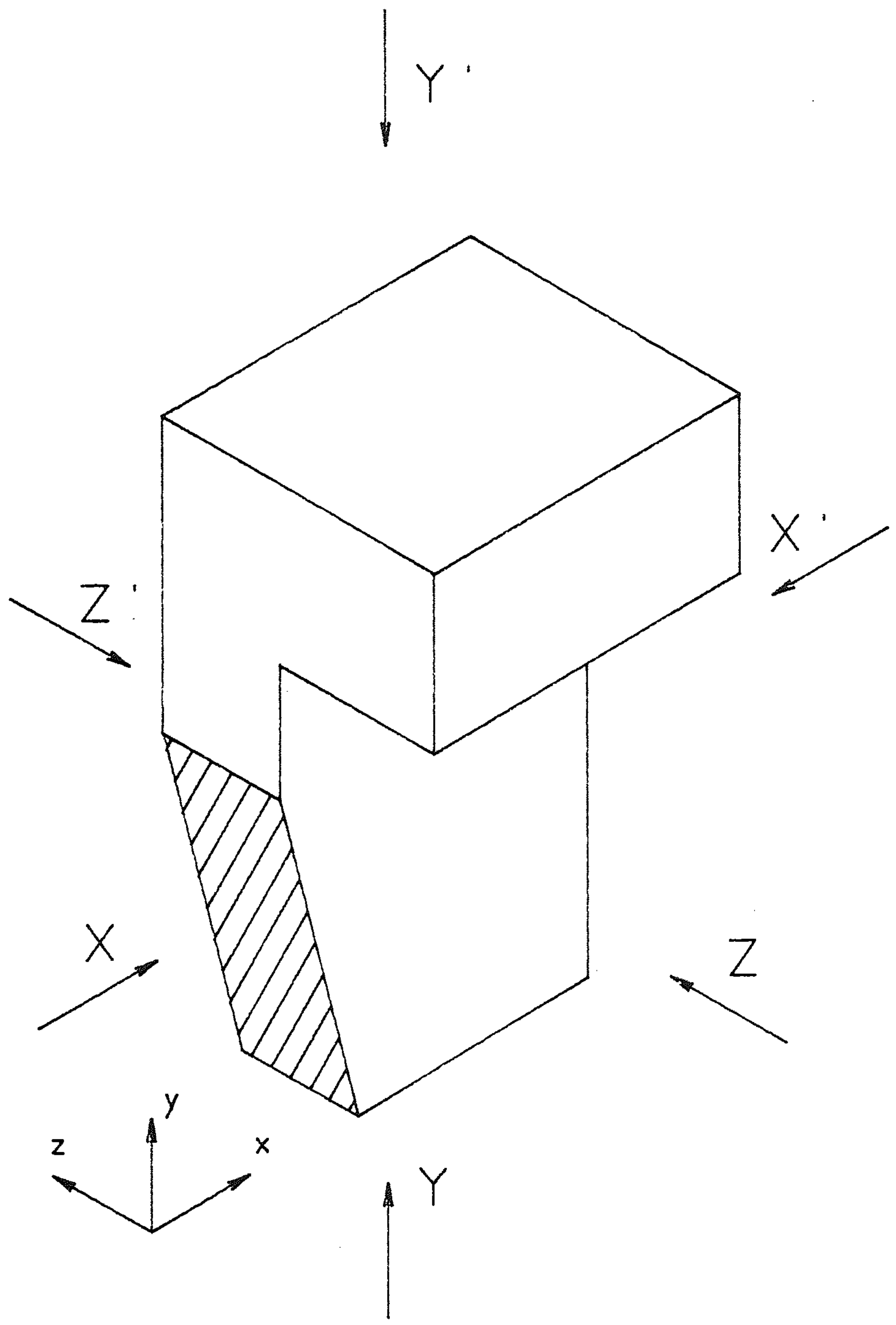


Fig. 6.1 Possible viewing directions for orthographic projections of a simple solid

model, although the views must be chosen carefully, the number of such views is only one of many factors which determine the total cost of the final drawing. It may be possible to produce illustrative pictorial views of the object at no significant extra cost. This point is taken up again in Chapter 8.

The scope of this Chapter is restricted to an examination of possible methods of selecting a set of exterior orthographic views that is well-suited for later dimensioning. The questions of when and where to draw sectional views are addressed in Chapter 7. Two algorithms are proposed; the first, Algorithm A, was not fully implemented for reasons that are given. The second, Algorithm B, was implemented in full and forms the starting point for work described in later Chapters.

6.2 Algorithm A

Fig. 6.1 shows a pictorial view of a simple polyhedral component. Clearly, no consideration need be given to drawing a sectional view of such an object. The analysis which follows is based upon the axiom that , for the purposes of dimensioning, a viewing direction must be normal to at least one of the object's surfaces. The arrows on Fig.6.1 indicate viewing directions which are normal to several surfaces. No arrow is shown normal to the shaded surface as no other surface is parallel (or normal) to it.

Strategy No.	Views					
	X	X'	Y	Y'	Z	Z'
1	0	0	0	1	0	1
2	0	0	0	1	1	0
3	0	0	1	0	0	1
4	0	0	1	0	1	0
5	0	1	0	0	0	1
6	0	1	0	0	1	0
7	1	0	0	0	0	1
8	1	0	0	0	1	0
9	0	1	0	1	0	0
10	0	1	1	0	0	0
11	1	0	0	1	0	0
12	1	0	1	0	0	0
13	0	1	0	1	0	1
14	0	1	0	1	1	0
15	0	1	1	0	0	1
16	0	1	1	0	1	0
17	1	0	0	1	0	1
18	1	0	0	1	1	0
19	1	0	1	0	0	1
20	1	0	1	0	1	0

Table 6.1 View Selection Matrix

A trained draughtsman would rapidly decide which combinations of views will be the most useful for dimensioning. In the terminology of Decision Theory: if the objective is to produce a dimensioned drawing then there will be some strategies which, when processing the appropriate class of model, give Utilites which approach unity. In the terminology of Artificial Intelligence, the draughtsman is employing a generate-and-test paradigm.

In order to limit the number of possible strategies, two further axioms are proposed:

a) At least two views are required of any object which cannot be modelled by one translational sweep of an Area Set.

b) No two views may be in opposite viewing directions.

The possible Strategies may then be arranged in a View Selection matrix, **S**, which is shown in Table 6.1. The presence of a "1" in a row indicates that a view in that direction is drawn. e.g. Strategy No. 6 would be to draw two views, as seen in the direction of arrows X' and Z.

For the purposes of dimensioning, some measurements will be made from one or more datum surfaces and such surfaces should be normal to a viewing direction. The dimensions themselves will be placed around a surface which is viewed normally and, preferably, has none of its Edges hidden by other surfaces. It is therefore suggested that the following counts provide evidence for the suitability of a viewing direction:

a) number of visible normal surfaces,

b) number of hidden normal surfaces,

c) number of visible parallel surfaces and

d) number of hidden parallel surfaces.

An additional tally is included in what follows as it may be regarded as evidence of the ease with which a view may be interpreted:

e) number of visible inclined surfaces.

For the object shown in Fig.6.1 the six directions score points which are arranged in a Visibility Matrix, **V**, shown in Table 6.2. Multiplying the View Selection matrix, **S**, by the Visibility matrix, **V**, gives the Consequence matrix, **C** = **SV**, shown in Table 6.3. The Consequence matrix may be used to derive Utilities, which must be based upon subjective judgements of the least and most desirable consequences.

In any situation where a subjective assessment is required, undesirable characteristics are more easily identified than desirable ones. Most people can recognize bad acting when they see it even though they might not be able to analyse what makes a good actor. Therefore, the task of assessing Utilities is begun by identifying the consequence with zero Utility. In this case the question is posed: what is the least desirable consequence that may be identified on the Consequence Matrix? One response might be to condemn Strategy No.17 on the grounds that the return on drawing those

View along arrow	Number of visible normal surfaces	Number of hidden normal surfaces	Number of visible parallel surfaces	Number of hidden parallel surfaces	Number of visible inclined surfaces
X	1	1	6	0	1
X'	1	1	6	0	0
Y	2	1	5	0	1
Y'	1	2	5	0	0
Z	2	1	6	0	0
Z'	1	2	5	1	0

Table 6.2 Visibility Matrix for the object shown in Fig. 6.1

particular three views is one of the smallest numbers of visible normal surfaces and the maximum number of hidden normal surfaces.
The author has observed that an effective test of the judgement of zero Utility is to ask if the opposite qualities imply a Utility which approaches unity. In the present case the question is: would two views showing the largest total of visible normal surfaces and the smallest total of hidden normal surfaces be the most desirable consequence ? In other words, a subjective judgement is required as to whether Strategy No. 4 has a Utility of 1.
Before discussing the design of Algorithm A, based upon the above, it is useful to review the stages which led to the present position, as reference is made to this process in later Chapters.

First, certain axioms were accepted. These axioms were then applied in order to constrain some of the problem variables. Next, and of vital importance, certain quantifiable properties were proposed as evidence for the suitability of a view for dimensioning. In the above example, the evidence provided by the last three columns in the Consequence Matrix is not required for decision-making and need not have been generated. Conversely, new evidence emerged when assessing the Consequence with zero utility: the marginal utility of an additional view. Lastly, a value judgement is required in order to

Strategy No.	Total of visible normal surfaces	Total of hidden normal surfaces	Total of visible parallel surfaces	Total of hidden parallel surfaces	Total of visible inclined surfaces
1	2	4	10	1	0
2	3	3	11	0	0
3	3	3	10	1	1
4	4	2	11	0	1
5	2	3	11	1	0
6	3	2	12	0	0
7	2	3	11	1	1
8	3	2	12	0	1
9	2	3	11	0	0
10	3	2	11	0	1
11	2	3	11	0	1
12	3	2	11	0	2
13	3	5	16	1	0
14	4	4	17	0	0
15	4	4	16	1	1
16	5	3	17	0	1
17	3	5	16	1	1
18	4	4	17	0	1
19	4	4	16	1	2
20	5	3	17	0	2

Table 6.3 Consequence Matrix for the object shown in Fig. 6.1

identify zero Utility and, by examination of opposites, the most desirable Consequence.

SPECIFICATION OF ALGORITHM A

1. Set up a View Selection matrix, **S**, based upon the stated axioms.

2. Compute a Visibility matrix, **V**, based upon numbers of visible and hidden normal surfaces.

3. Compute the Consequence matrix, **C** = **SV**

4. From the Consequence matrix select the row(s) with the maximum total of visible normal surfaces. If there is only one such row then exit. Otherwise:

5. Select the row(s) with the minimum total of hidden normal surfaces. If there is only one such row then exit. Otherwise:

6. Select the row(s) with only two views. If there is more than one then choose one at random. Exit.

6.3 Principles of Algorithm Design

Algorithm design is an iterative process. Algorithm A may discover one of the better Strategies for a limited set of rather simple geometric models. The algorithm may be judged less than satisfactory when processing more complicated models and will need to be modified and retested. Just how much testing is required is a topic which is taken up in Chapter 11.

Of more immediate concern is the question of the cost of computing those properties which have been established as evidence for or against the suitability of a viewing direction. Some initial work was carried out on an implementation of Algorithm A in order to determine the value and cost of the evidence that is generated. The Strategy whose Utility is unity can only be discovered after the numbers of visible and hidden normal surfaces have been computed. The SUPERMODEL data were processed by using the hidden-surface algorithm which is described in 8.2. The outputs from this algorithm are continuous (visible) Edges and dashed (hidden) Edges. An Edge represents the intersection of two surfaces or a boundary between the object and the surrounding air. Many Edges represent both situations. It is possible to decide from this output whether an entire surface is visible, hidden, or partially hidden. There is no need to process all six views because every surface which is completely visible in one viewing direction will be hidden when viewed in the opposite direction.

The major problem with this approach was the time required to process just one orthographic view and then count hidden surfaces, visible surfaces and partially visible surfaces. This computational expense may be regarded as an investment if most of the information obtained will be used at a later stage. Less data are required in order to display an orthographic view than to determine the visibility of each surface. This is because visible Edges may hide other Edges and not all the hidden detail may be required on the view that is to be dimensioned. Moreover, only two of the three views processed may be required. Taking these factors together, the computation of the visibility of every surface is a poor investment.

In order to make an informed choice of viewing direction some information is required but how much will suffice? This question was first raised in Chapter 5 and it is now addressed by reference to a new decision-making procedure - Algorithm B. Instead of being based on the certainty of the numbers of hidden and visible surfaces Algorithm B is based upon the probability of seeing (or not seeing) a normal surface in one viewing direction. There is no way to prove that certainty is essential; the best that can be done is to investigate the performance of an implementation of Algorithm B and this is reported in Chapter 11.

As well as the cost of computation, there is another consideration, suggested above, which renders Algorithm A impracticable. This is the problem of partially visible surfaces.

Fig. 6.2 shows an isometric view of a bracket which might be used to attach two surfaces at 45°. A view in the direction of arrow Y' reveals only part of the shaded surface. The hidden-surface algorithm can handle this situation but the question immediately arises of

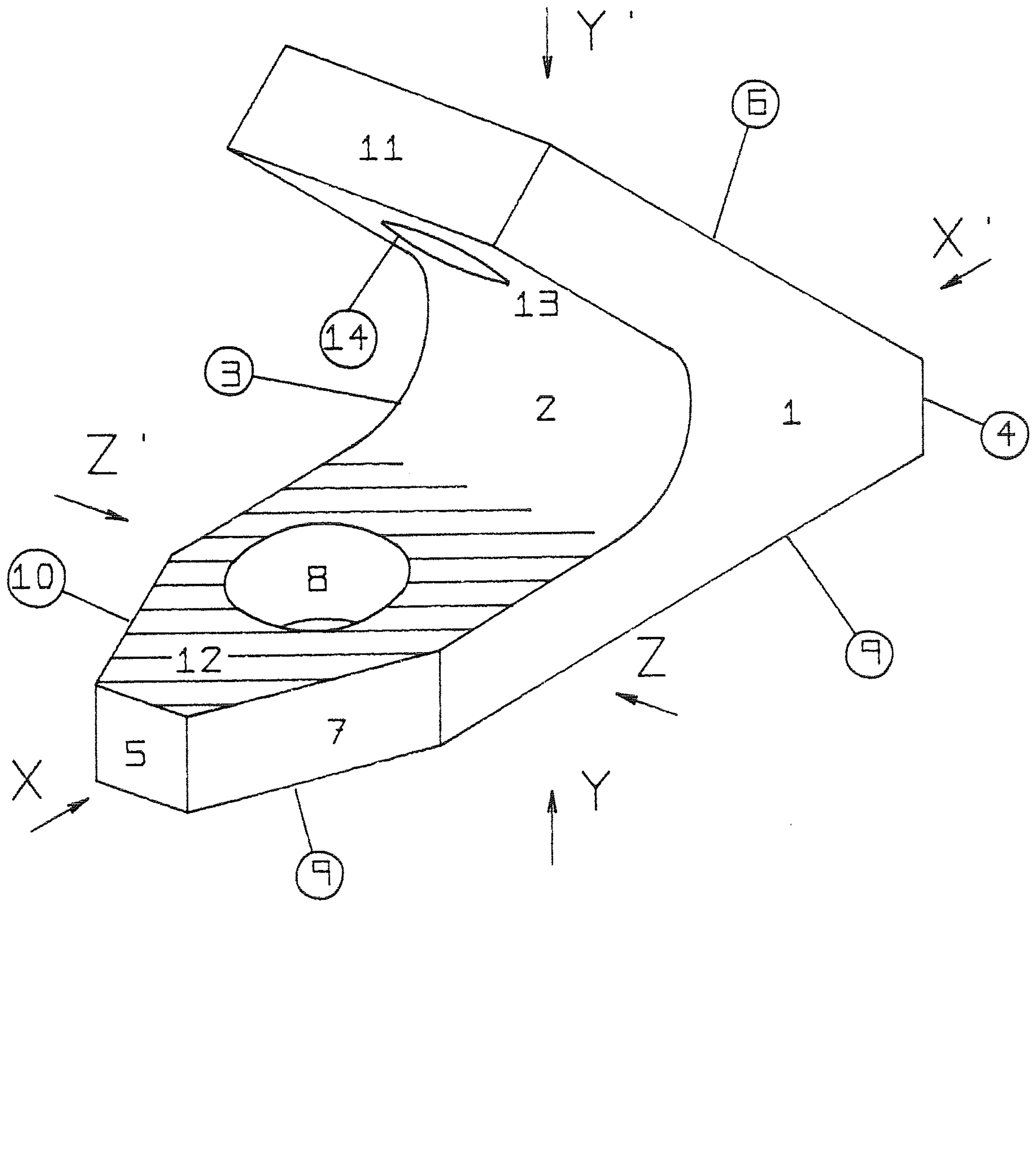


Fig. 6.2 Isometric view of a bracket

how to award points to such a surface. One could argue that, as some of the surface is visible, this enhances the Utility of the viewing direction. There is, however, a valid contrary argument: the dimensioning of a surface which has some full lines and some hidden detail lines is bound to be untidy and could lead to confusion. Therefore, a partially visible surface is evidence which argues against a viewing direction for dimensioning purposes. It was decided that the counting of visible, hidden, and partially visible surfaces was not a fruitful line of investigation.

6.4 Algorithm B

The SUPERMODEL modelling package described in Chapter 4 allows the user to create a Boundary Representation of the solid object. The geometry of each flat and curved surface is established in (x,y,z) coordinates which are then subjected to a rotational transformation if the user has specified a line about which the surface has been rotated.

It is a relatively straighforward operation to determine which surfaces are parallel to others within the object. Fig. 6.3 illustrates the general principles. If the first surface has been rotated through angle B, about an axis that is at angle A to the horizontal, then a second surface is parallel if it has been rotated about an axis, also at angle A, through the same angle B.

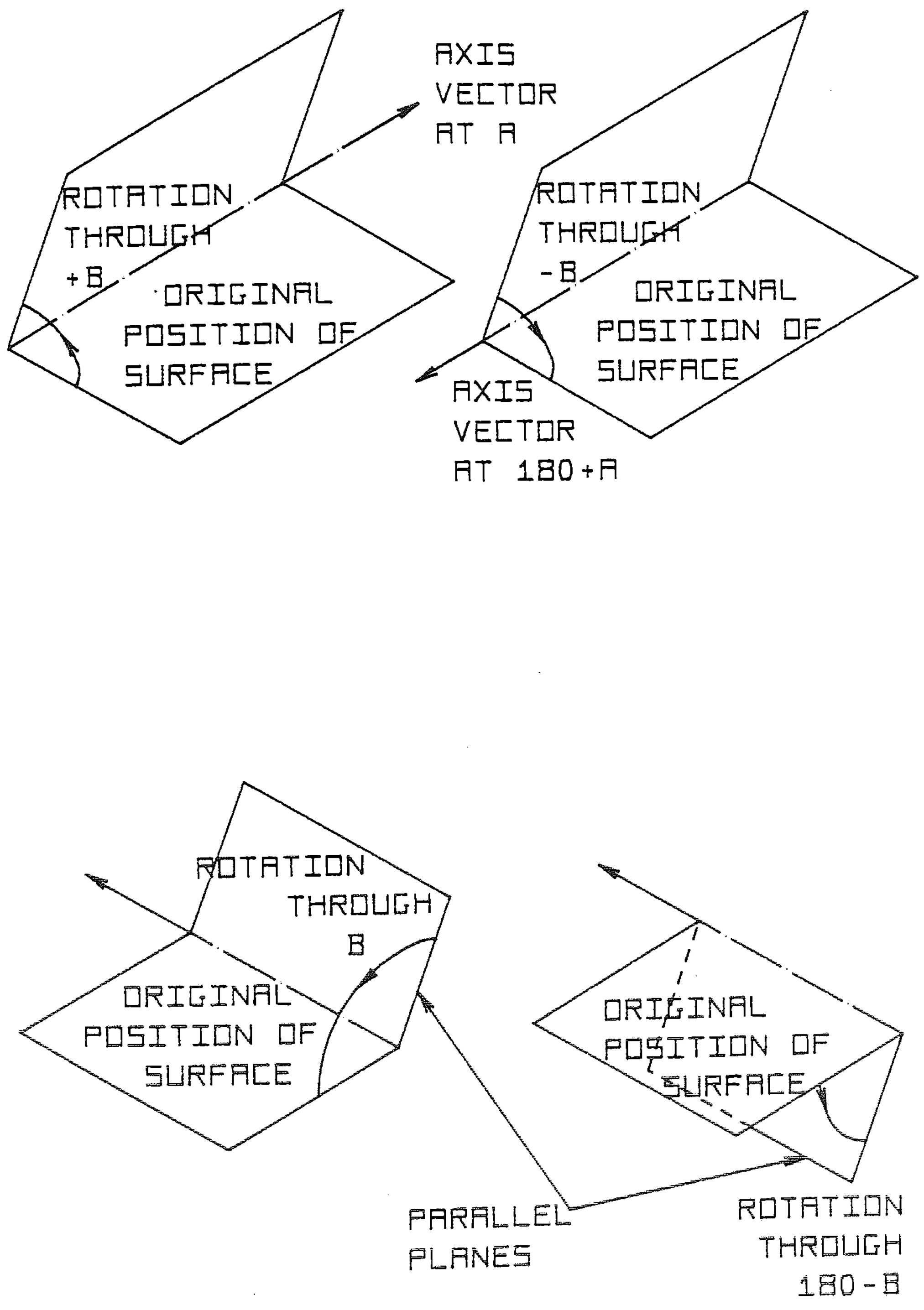


Fig. 6.3 Methods of testing if two SUPERMODEL surfaces are parallel

Alternatively, the second surface is parallel to the first if it has been rotated about an axis, at angle 180+A, though -B. Two planes rotated about parallel axes are parallel if the angles of rotation are equal or if their algebraic sum is 180 degrees.
Thus, it is not computationally expensive to identify parallel planes within the model and to count the number of such planes. Table 6.4 gives details of the parallel planes that were computed for the bracket illustrated in Fig. 6.2. It is an axiom for what follows that the suitability of a viewing direction for dimensioning correlates positively with the number of normal object-surfaces.

A good candidate viewing direction is often one which is not only normal to many surfaces but also one from which these surfaces are entirely visible. However, the computation of visible, or partially visible surfaces is rejected on grounds stated above. It is therefore necessary to investigate alternative mechanisms for computing parameters which could provide a basis for view selection.
The first parameter to be considered is related to the perpendicular distances to each of the normal planes. Fig. 6.4 shows a more complicated object than Fig. 6.1 and 6.2. The similarly shaded planes are parallel. Arrow X1 indicates a viewing direction normal to seven planes and it is possible to compute the minimum (xmin) and

Number of parallel surfaces	Fraction of total %	Surface Numbers	Inclination of axis of rotation degrees	Rotation around axis degrees
3	21.43	8,9,12	0	90
3	21.43	6,13,14	135	90
3	21.43	1,2,3	0	0
2	14.29	4,5	90	90
1	7.14	11	45	90
1	7.14	7	90	150
1	7.14	10	90	30

Table 6.4 Ranking of parallel surfaces for the bracket illustrated in Fig.6.2

maximum (xmax) values of coordinate x at points on the surface of the object. A relative (dimensionless) distance, r is defined as:

r = (x - xmin)/(xmax - xmin)

Intuitively, a normal plane within a solid object that has a relative distance of zero will be entirely visible whereas a normal plane with a relative distance of unity must be entirely hidden. The surfaces which are input to SUPERMODEL are orientable and it follows that the open side of a normal flat surface at relative distance zero is towards the viewing position. Conversely, the open side of a normal flat surface at relative distance 1 is away from the viewing position. In other words, the probability that a normal surface is entirely hidden is 0 at relative distance 0 and is 1 at relative distance 1.

There is no such simple correlation at intermediate values, as surfaces may be oriented independently of their relative distance. A view in the direction of arrow X1 is considered to be more suitable for dimensioning than a view in the direction of X1' if more normal surfaces are (partially) visible along X1 than along X1'. In Fig. 6.4 there are five surfaces (14,1,8,9,3) with their open sides towards X1 and only two surfaces with their open sides towards X1'. The same numbers apply to Fig.6.5, but here one observes that more surfaces would be visible along arrow X2'.

In general, the probability of viewing a normal surface

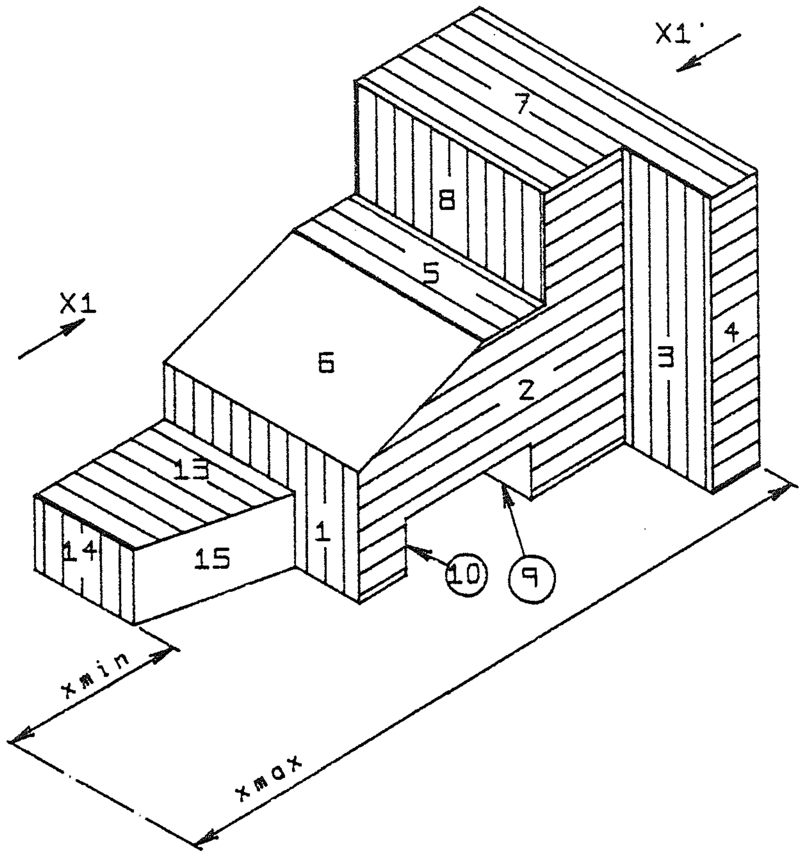


Fig. 6.4 Distances to normal planes

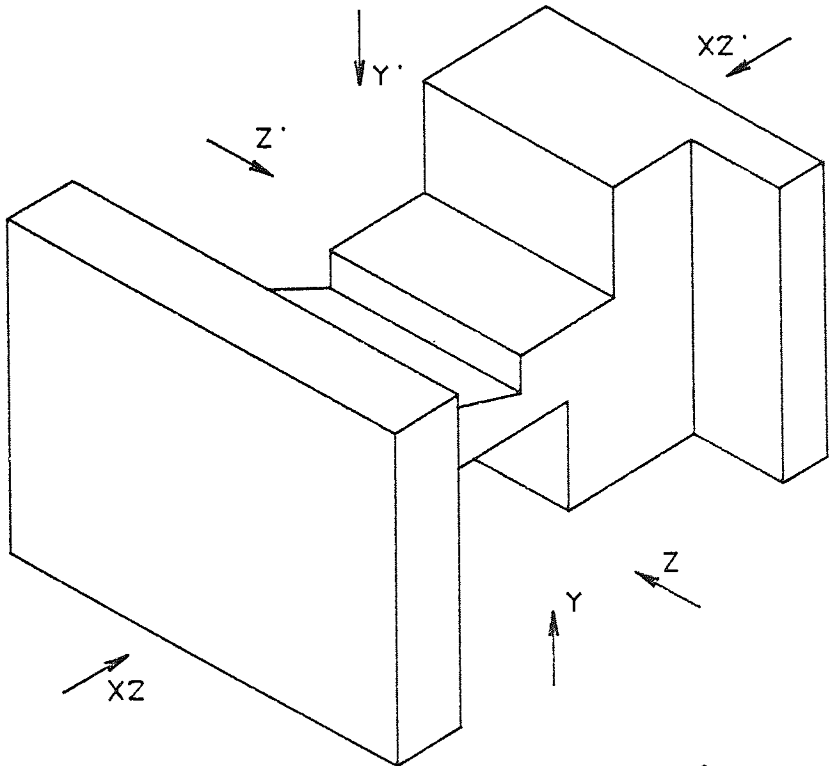


Fig. 6.5 Visibility of normal surfaces

whose open side is away from the viewing direction is zero. In Fig.6.5, two normal surfaces have their solid sides away from Y but three are away from Y'. Two normal surfaces have their solid sides away from Z but only one is away from Z'.

The position is still more complicated for an object with relatively large flat surfaces that are not normal to the viewing direction. Fig.6.2 illustrates such a case. On the basis of normal surfaces only, there is one with its solid side away from X and one with its solid side away from X'. This argues that there is nothing to choose between X and X' but, if the visible inclined surfaces, 7,10,11 & 13, are taken into account it is clear that a view along arrow X reveals more of the object.

The convention adopted for the orientation of surfaces in SUPERMODEL is that the normal vector to a flat surface points to the solid side of that surface. Suppose that the magnitude of this vector is equal to the area of the surface. If the component of this vector, resolved in the direction of, say, arrow X, is negative then the surface could definitely not be seen along arrow X but might be visible along arrow X'. If such a component is positive then its magnitude is a measure of the area that could appear in a view along arrow X.

If it is simply the number of surfaces orientated towards the viewing position that matters then the area

Surface No.	Angle between normal vector and vector X (degrees)	Positive Component X	Negative Component X'
1	-90	0	0
2 conical			
3	90	0	0
4	180	0	1
5	0	1	0
6	135	0	0.7071
7	60	0.5000	0
8 conical			
9	90	0	0
10	-60	0.5000	0
11	45	0.7071	0
12 conical			
13	-45	0.7071	0
	Totals	3.4142	1.7071
	Exposure bias	0.6667	0.3333

Table 6.5 Components of Normal Vectors for Fig. 6.2

of each may be set to unity. Table 6.5 shows the results obtained for the bracket illustrated in Fig. 6.2. The totals of positive and negative components argue strongly that more flat surfaces could be exposed by viewing along arrow X, rather than along arrow X'.

The only undeniable fact in the above is that surfaces with negative vector components cannot be seen under any circumstances. It would be improper to draw other general conclusions from what amounts to anecdotal evidence. However, as a working hypothesis, it is assumed that totals such as those shown in Table 6.5 indicate a "bias" that the indicated viewing direction will expose the greater number of surfaces. To conform to the same laws as probability, each total has been divided by the overall total.

The computational cost of obtaining the information shown in Table 6.5 is not great. The orientation of a SUPERMODEL flat surface is input directly by the user of the software and a normal vector may easily be computed and written to diskette as part of an existing file. The angle between the viewing vector and each normal vector and, hence, the positive and negative components may be rapidly computed, even for a large number of flat surfaces.

It is, however, possible to envisage models in which many flat surfaces are orientated positively with the viewing vector and yet few of them are visible. Fig. 6.6 illustrates such an object. None of the numbered

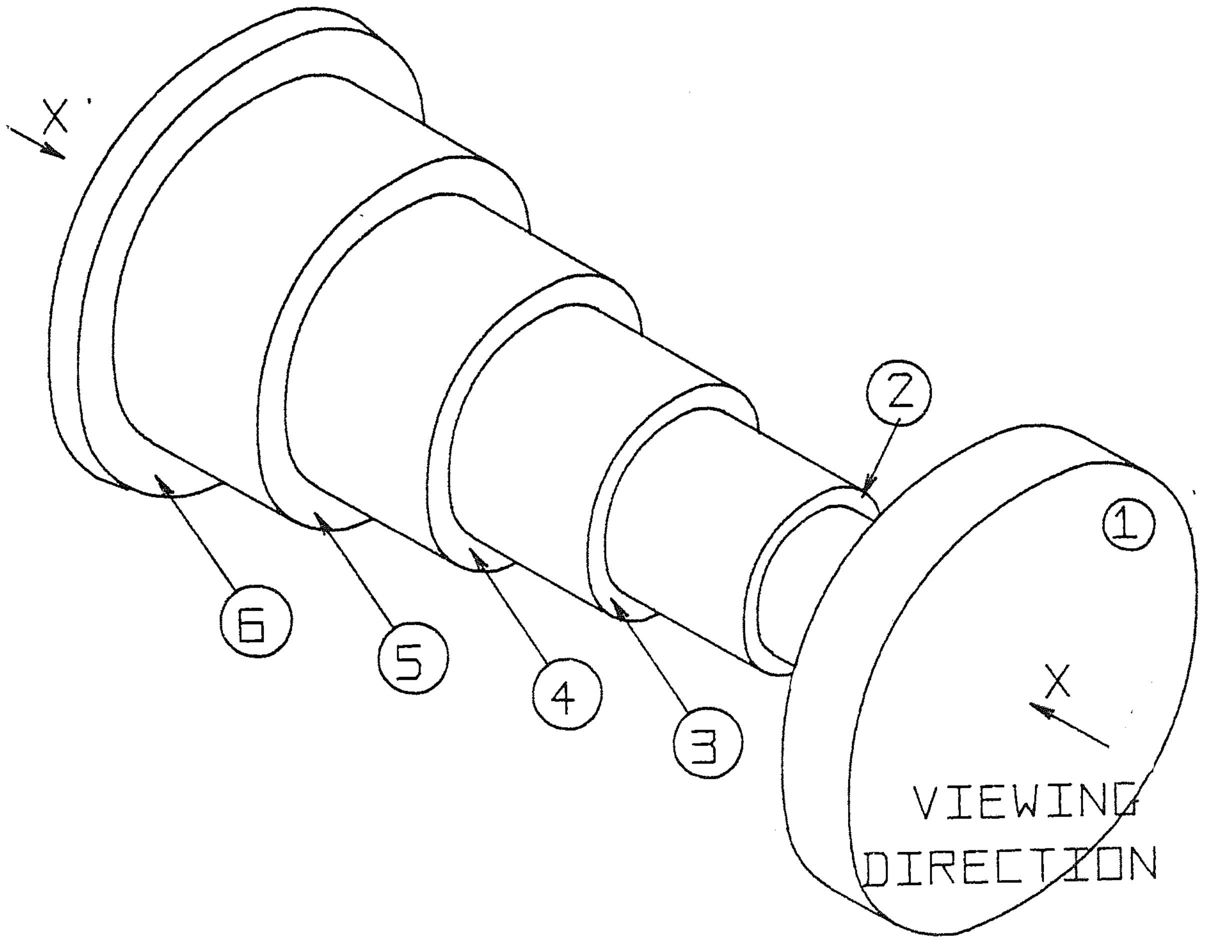


Fig. 6.6. Object with many positively orientated flat surfaces, few of which are visible

surfaces, except No.1 (which is larger than No.6), would be visible along arrow X and a clearer dimensioned view might result from viewing the object along arrow X'. The question arises: how much more information is required in order to make a decision about viewing direction with reasonable confidence?

Such a question was raised when Bayes' theorem was discussed in Section 5.5. The theorem is now applied to assist the design of Algorithm B. First, we limit the visibility of a surface to two categories:

a) those on which at least half of the Nodes (not Centres) are visible, which are dubbed "clear" surfaces and

b) those on which less than half of the Nodes are visible, which are referred to as "unclear" surfaces.

Having computed the directions which are normal to the greatest number of surfaces, one such viewing vector, X, is selected for further consideration. If the number of clear surfaces seen along X is i and the number of clear surfaces seen along X' is i' then two classes of model are considered:

M_1 for which $i > i'$ and

M_2 for which $i \leq i'$

If N surfaces are orientated towards X and N' surfaces are orientated towards X' then, as each may be either clear or unclear, the number of combinations is:

$$C = 2^{N+N'} \qquad 6.1$$

As the number of clear surfaces seen along X is i, and $0<i\leq N$, the number of combinations is $\dfrac{N!}{(N-i)!i!}$

For each value of i there may be 0,1,...,n clear surfaces seen along X' if the model is to qualify for the first class, where n = min(i-1,N')

Therefore, the number of combinations that would qualify for membership of class of model, M_1 is:

$$Q = \sum_{i=1}^{N} \left(\frac{N!}{(N-i)!i!} \sum_{j=0}^{\min(i-1,N')} \frac{N'!}{(N'-j)!j!} \right) \qquad 6.2$$

Combining equations 6.1 and 6.2 gives the probability, given the initial knowledge, that the model belongs to class M_1:

$$p(M_1 \mid K) = Q/C \qquad 6.3$$

and, as there are only two classes of model:

$$p(M_2 \mid K) = (C-Q)/C \qquad 6.4$$

In order to apply Bayes' theorem it is necessary to answer the question: what is the probability of finding that one particular surface is clear when seen along X, given that the model belongs within class M_1 and the initial knowledge?

There are Q combinations which confer membership of the first class. For each of these combinations it is necessary to investigate the number of ways in which a particular surface could be clear and, at the same time, the rest of the complement of clear surfaces could be made up. If i surfaces are clear out of a maximum of N then the number of ways in which a particular surface would be clear is:

$$\frac{(N - 1)!}{(N - i)!(i - 1)!}$$

This expression may be summed for each of the N numbers of clear surfaces which could exceed the number of clear surfaces seen along X'. Hence, the number of ways in which a particular surface could be clear given membership of the first class, is:

$$m = \sum_{i=1}^{N} \left(\frac{(N - 1)!}{(N - i)!(i - 1)!} \sum_{j=0}^{\min(i-1,N')} \frac{N'!}{(N'-j)!j!} \right) \qquad 6.5$$

Hence, the probability that a particular surface is clear when seen along X (additional information I), given membership of the first class is:

$$p(I \mid M_1 \text{ and } K) = m/Q \qquad 6.6$$

The total number of ways in which a particular surface can be clear is exactly half of the total combinations, hence:

$$p(I \mid M_2 \text{ and } K) = (2^{N+N'-1} - m)/(C-Q) \qquad 6.7$$

Using Bayes' theorem for the case where only two classes of model are considered and combining equations 6.3, 6.4, 6.6 and 6.7:

$$\frac{p(M_1 \mid I \text{ and } K)}{p(M_2 \mid I \text{ and } K)} = \frac{m}{(2^{N+N'-1} - m)} \qquad 6.8$$

The implications of the above are best examined by means of a numerical example. Suppose that there are 4 surfaces orientated towards the X direction and 2 surfaces orientated towards X'

$C = 64 \qquad Q = 42 \qquad m = 26$

$$\frac{p(M_1 \mid I \text{ and } K)}{p(M_2 \mid I \text{ and } K)} = \frac{26}{(2^5 - 26)} = \frac{26}{6}$$

In other words, the odds in favour of the first class of model start at 42/22 but, if one surface is found to be clear when seen along X, then the new odds are 26/6 in favour. This tends to the conclusion that it is well worth checking one surface (whose relative distance is not zero) before making a decision about the viewing direction.

Therefore, the following strategy is proposed for Algorithm B.

Compute the probabilities using the initial knowledge. If the initial odds are very high then draw the view which has the higher probability of showing more clear surfaces. Otherwise, calculate the revised probabilites and select one surface at random. If it is clear then look at the revised odds and make a choice.

SPECIFICATION OF ALGORITHM B

6.4.1 Determine the numbers of parallel surfaces and sort them in order of magnitude. Select the viewing direction that has the greatest number of normal surfaces.

6.4.2 Count the number of normal surfaces, N, orientated towards the viewing direction and the number, N', orientated away from it. If N'> N reverse the viewing direction.

6.4.3 Using equations 6.1 and 6.2, determine the odds, D1, in favour of more surfaces being clear in the viewing direction, X, than in the opposite direction, X'.

6.4.4 If D1 < 3 (previously determined by experiment) go to step 6.4.6, otherwise select at random a normal surface that is orientated towards the viewing direction and has a relative distance greater than zero. Use the hidden-surface algorithm (described in Section 8.2) to determine if the surface is clear.

6.4.5 If the surface is unclear then reverse the viewing direction and go to step 6.4.6, otherwise use equations 6.5 and 6.8 to determine the new odds, D2, in favour of more surfaces being clear in the viewing direction. If D2 < 3 reverse viewing direction.

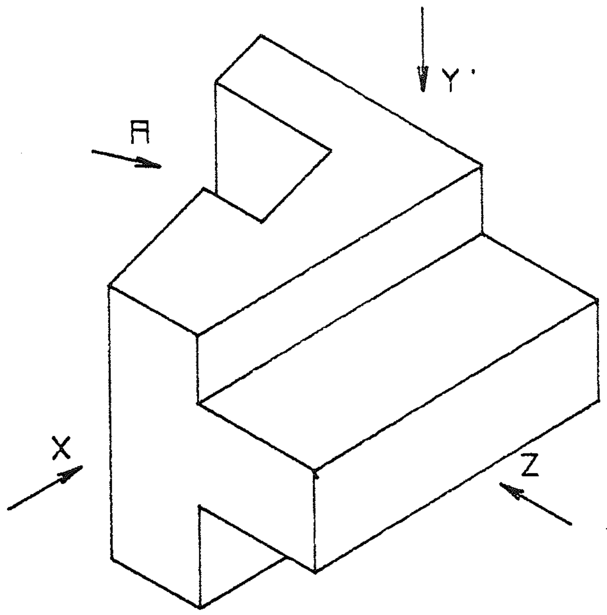
6.4.6 Draw an orthographic view in the current viewing direction. If other views are to be drawn select that with the next highest number of normal surfaces and go to step 6.4.2, otherwise exit.

The results obtained from an implementation of the above algorithm are discussed in Chapter 11.

6.5 Auxiliary Views

The assessment procedure described in Section 6.2 sometimes gives results which show a significant number of parallel surfaces that are not normal to the x, y or z axes. If such surfaces are ranked first or second in a tabulation such as Table 6.2 then viewing directions normal to these surfaces must be chosen. The situation is less easy to resolve when such a set of parallel surfaces is ranked third or below but forms a significant fraction of the total number of surfaces.

An object which falls into this category is illustrated in Fig.6.7. There are four surfaces normal to arrow Y' and four surfaces normal to arrow X. Thus, these viewing directions have been chosen for the working drawing shown beneath the pictorial view. There are three surfaces normal to arrow Z and the same number normal to arrow A. The question is: should a third view be drawn and, if so, which one should be selected, Z or A?



VIEWS ALONG X & Y' IN THIRD ANGLE PROJECTION

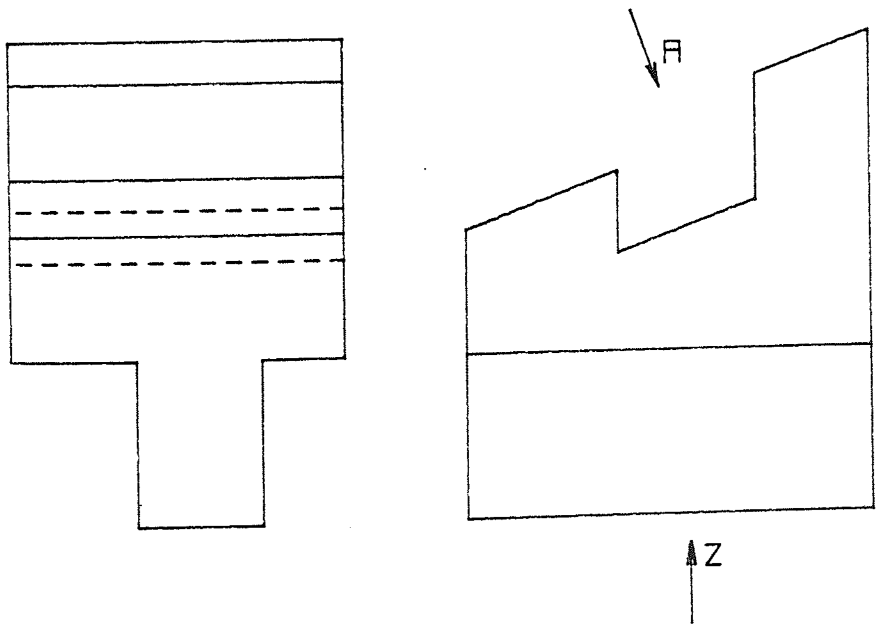


Fig. 6.7 Object whose working drawing may require an auxiliary view

There are several practical difficulties connected with the drawing of auxiliary views. The layout of the total drawing is complicated by such views. It is not sufficient merely to position the views according to the rules of First of Third angle projection. There is a well-established convention that the view from which an auxiliary view is projected has an arrow appended in order to indicate the viewing direction, the exact viewing angle may need to be dimensioned and the auxiliary view itself is labelled to indicate how it relates to the adjacent view.
Many of the dimension lines on an auxiliary view will be aligned at angles other than 0^o and 90^o. This makes for a more complicated dimensioning algorithm but, more important, the dimension figures and text should normally be written parallel to the dimension lines and this may be an impossibility with certain computer displays and plotters. Some commercial systems have an even more severe restriction on text: it must all read from left to right. It was decided that neither of these restrictions was acceptable for this project. In consequence it was necessary to write a special set of character generation routines for the SuperBrain computer and their selection and use are briefly described in Appendix C.

The presence of even one auxiliary view tends to place the drawing in a special category as far as certain users of the drawing are concerned. The person manufacturing the object that is represented is sometimes less well-versed on the finer points of engineering drawing than the designer/draughtsman. Moreover, if there is a significant amount of detail in other planes which is grossly foreshortened on the auxiliary view then it becomes a source of "visual noise" which is unhelpful to the reader of the drawing. These are additional reasons for caution when deciding if an auxiliary view should be drawn.

There is one overwhelming piece of evidence in favour of drawing an auxiliary view: if the normal surface(s) have been classed as hard interfaces then their dimensioning must be explicit and this is best done on a view which shows the true shape of the surface.

6.6 Algorithm C

In order to design a decision-making algorithm to cope with the problem of auxiliary views, the above points provide the following evidence:

a) If a viewing direction is ranked first or second by counting the number of normal surfaces, then it should be drawn without further consideration.

b) If a viewing direction is ranked third or fourth and is not normal to either of the top-ranked directions, this is evidence against drawing the view.

c) If a viewing direction is normal to one or more surfaces which have been classified as hard interfaces, this is evidence which strongly suggests drawing the view.

d) The number of Nodes on the normal surfaces is evidence of the need for the view.

e) The number of circular arcs on the normal surfaces is evidence of the need for the view.

f) The number of circular arcs on surfaces that are neither normal nor parallel to the viewing direction is evidence against the need for a view.

The ultimate decision that is required is whether a view should be drawn in a direction that is ranked third or lower. The nature of the problem is such that the use of a generate-and-test paradigm, such as used for Algorithm A, is inappropriate. The above criteria may, however, be used as the basis for a rule-based paradigm.

RULES FOR ALGORITHM C

6.6.1 If the viewing direction is normal to one or more hard interfaces then the view is to be drawn.

6.6.2 If the viewing direction is normal only to boundaries then it is rejected.

6.6.3 If the viewing direction is normal to less than three surfaces then it is rejected.

6.6.4 Count the number of Nodes/Centres on the normal surfaces. If it is not less than that for the views which have been accepted then the view is not to be rejected yet.

6.6.5 Count the number, N_1 of circular arcs that will project ellipses in the viewing direction. Count the number, N_2, of circular arcs that will project arcs or straight lines in the viewing direction. Let $Q = N_1/N_2$. If Q exceeds some arbitrary value (set for test purposes at 0.25) then the view is rejected.

A decision-making algorithm based upon the above rules may be implemented at little computational cost. The selection of a value for Q must be based upon an analysis of the performance of a prototype implementation. Indeed, the definition of quotient Q may be called into question when experimental evidence becomes available. Algorithm C is the first example of this type of design problem: a rule-based paradigm whose rules and parameters cannot be completely established from first principles. All such examples are discussed together in Chapter 11.

Various problems connected with the selection of exterior views for dimensioning have been considered, but many of the models that may be input have such internal complexity that an exterior view with hidden detail lines is not a suitable graph to which to add dimensions. In Chapter 7 the questions of when and where to draw sectional views are addressed.

CHAPTER 7

HIDDEN DETAIL AND SECTIONAL VIEWS

The discussion on the selection of views for dimensioning in Chapter 6 concentrates on views of the exterior of the object, but many of the objects that may be modelled have interior surfaces which cannot be seen when viewed normally from outside. Engineers have developed two complementary techniques which assist the interpretation of orthographic projections of such an object: hidden detail lines and sectional views, both of which are discussed in this Chapter.

7.1 Strategies for representing internal surfaces

Historically, the concept of a sectional view was introduced both in Mechanics and other applied sciences at an early stage of Western Civilization. The usefulness of such a concept depends upon the scale and the function of the artifact or natural object that is represented. Architects' and anatomists' drawings would be of no practical use if they showed only exterior views.

The situation is slightly different in engineering design because the internal details of some objects may be satisfactorily represented by the convention of drawing them with dashed lines. It is by no means the case that all components require a sectional view in order easily to be interpreted by the reader of the drawing.

The hidden detail lines that appear on a manually draughted exterior view are not drawn merely because they happen to be present within the object represented. Barnes and Tilbrook [39] put the case succinctly: "It is a basic rule of projection that whenever detail is visible it must always be drawn in irrespective of the fact that it may have been previously indicated on other views. There is, however, no such rule appertaining to Hidden Detail. In practice it is left to the discretion of the draughtsman to insert as much as he thinks necessary for a clear and correct interpretation of the drawing. A good draughtsman will concentrate on inserting the Hidden Detail wherever it appears and will only omit it when, in his opinion, its insertion will confuse the drawing and render it difficult to interpret."

The types of situation that can arise are illustrated in Fig. 7.1 & 7.2. The former is a drawing with insufficient Hidden Detail. The holes are not shown on the view labelled "?" and, as a result, it could be interpreted in any of the seven ways shown below it. Fig.7.2 illustrates excessive Hidden Detail. The view labelled "Full detail", although accurate, is difficult to interpret. The bottom view has the confusing Hidden Detail omitted and is far easier to read.

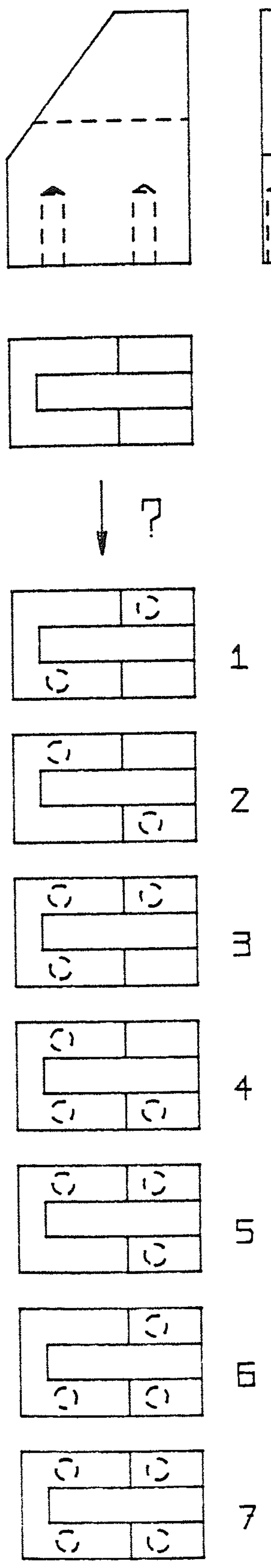


Fig. 7.1 Insufficient hidden detail

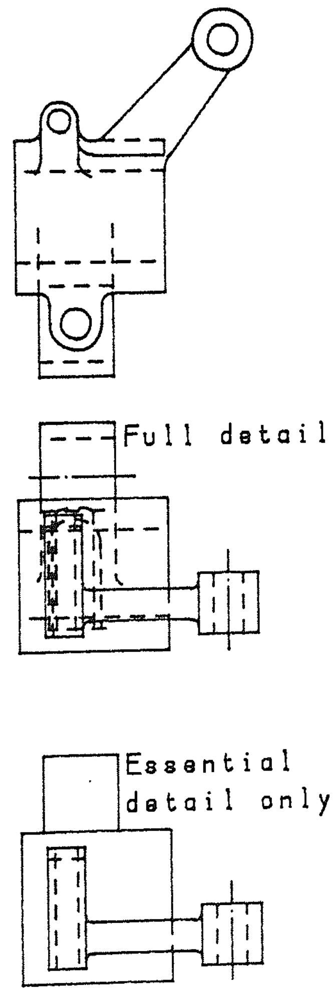


Fig. 7.2 Excessive hidden detail

For this project, the considerations described above pose several questions. Is there a reliable measure of internal complexity which is computable within a reasonable time and at an acceptable cost? If such a measure can be found, at what point does it become evidence for drawing a sectional view? If a sectional view is required, where should the section plane be positioned? Will one section plane be satisfactory or does the situation call for a compound section, or more than one sectional view? Can a sectional view (with its mandatory cross-hatching) be dimensioned more clearly than an exterior view with hidden detail lines? Each of these questions is addressed in this Chapter and the design of algorithms for sectioning is discussed.

7.2 Evidence for sectioning

Algorithm B, which is detailed in Chapter 6, is designed to output two or more orthographic views of the geometric model. A viewing direction is chosen on the basis that it will probably reveal more normal surfaces than the opposite viewing direction. A sectioning algorithm is required to make a decision on whether an exterior view or a sectional view will better contribute to the overall objective of producing a comprehensible working drawing.

The object shown in Fig. 6.6 illustrates some of the points that need to be considered. Most of the normal surfaces have the same orientation as No.1, but that is

the only one visible along arrow X. This, of course, is an extreme case but the conclusions are no less valid.
The "regret" associated with drawing an exterior view arises from the quantity of hidden-detail that it contains. This may be because the hidden-detail lines are densely packed, making interpretation of the view more difficult. Alternatively, the hidden surfaces may be highly-ranked in the hierarchy of surfaces (set out in Section 4.5) and require to be dimensioned explicitly. Dimensioning to hidden-detail is less readily comprehended than dimensioning to visible Edges. Maximum "regret" would occur if densely packed hidden-detail had to be dimensioned.
If all parts of the model from surface No.1 to an infinitesmal distance before surface No.2 were to be removed, surfaces No.2,3,4,5 & 6 would then be visible. The "regret" associated with this strategy would be the absence of a graphical description of surface No.1, and the amount by which the computational cost of a sectional view exceeds that of an exterior view.
The "regret" at the absence of surface No.1 may be reduced if:
a) its function is ranked rather low in the hierarchy of functions, in which case its dimensions may be implied from those of adjacent higher-ranked surfaces. Or,
b) another surface, having some dimensions common with surface No.1, is still visible.

Fig. 7.3 illustrates several of the questions raised above. The object has sufficient internal details to produce a significant number of hidden-detail lines when viewed along arrow X but it would still be possible to dimension the exterior view with reasonable clarity. There are two sensible alternative positions for a single section plane and the one that is shown could be clearly dimensioned. A compound section plane gives a sectional view which reveals all the internal details but such a view would not necessarily make for a clearer dimensioning scheme, nor would it be easier to interpret.

The evidence in favour of drawing a sectional view may be quantified by defining the following parameters:

P_1 The ratio of the number of hidden Edges to the total number of Edges within the geometric model.

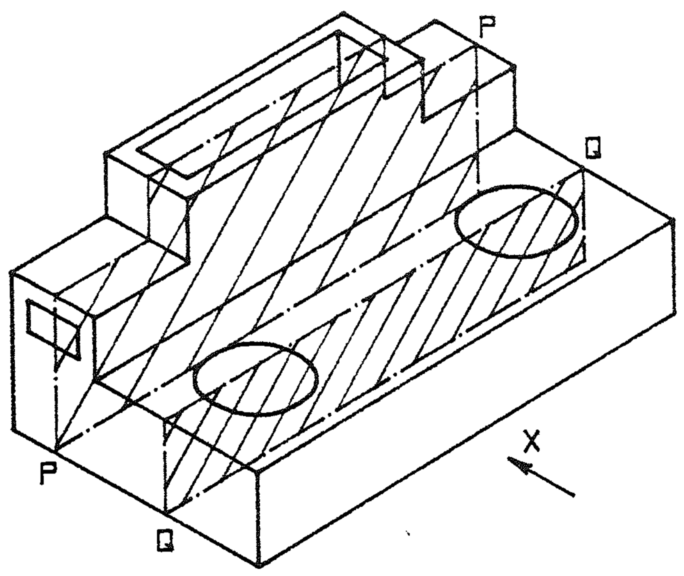
The value of this parameter may be computed by executing the hidden-surface algorithm which is described in Section 8.2.

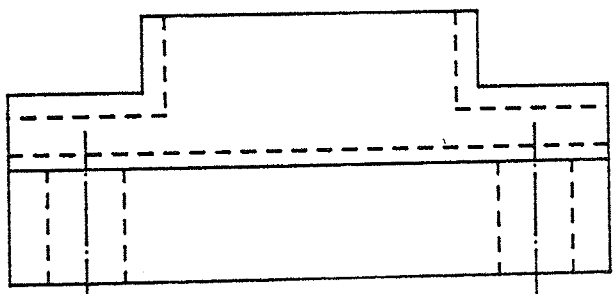
P_2 The number of normal surfaces, requiring explicit dimensioning (because their function is highly ranked, as specified in Section 4.5), that are hidden if an exterior view is drawn.

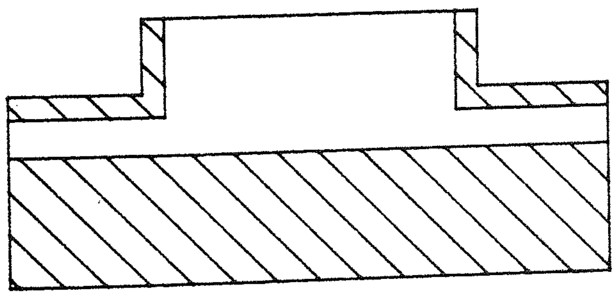
Computation of this parameter is possible by executing the hidden-surface algorithm.

P_3 The number of surfaces (both normal and inclined) that would be revealed per normal surface removed.

This parameter would be be prohibitively expensive to




Exterior view along arrow X


Sectional view on plane P-P

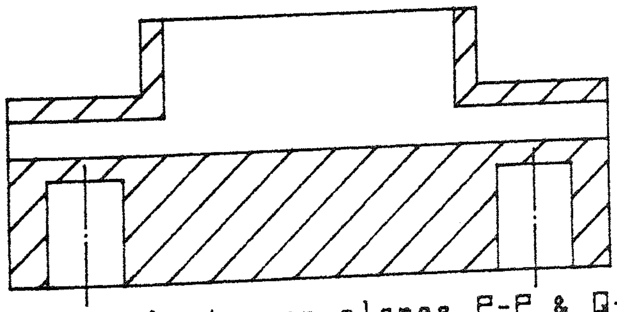

Sectional view on planes P-P & Q-Q

Fig. 7.3 Criteria for a sectional view

compute. It would require one execution of the hidden-surface algorithm for each of the normal surfaces.

P_4 The number of Nodes/Centres that are superposed in the orthographic view.

This parameter is readily computed without executing the hidden-surface algorithm.

The parameters which provide positive evidence in favour of drawing an exterior view are:

P_5 The number of normal surfaces, requiring explicit dimensioning, that are visible if an exterior view is drawn.

One execution of the hidden-surface algorithm would determine the value of this parameter.

P_6 The number of different relative distances to normal surfaces which have their "open" sides towards the viewing direction.

This parameter is readily computed without executing the hidden-surface algorithm.

7.3 Algorithm D

The objective of this algorithm is to decide whether an exterior view contains so much hidden detail that a sectional view should be drawn instead. If the exterior view is rejected then there are some important decisions to be made about the precise nature of the sectional view that is to be drawn. These questions are addressed in Section 7.4 below.

The decision-making algorithms described in Chapter 6 select from strategies which can be implemented with equal ease. All of the strategies must be available or else the overall objective (producing a dimensioned drawing) is frustrated. Such is not the case when considering a sectional view. It is possible (although perhaps not very elegant) to produce a working drawing without generating a sectional view.

It is demonstrated in Section 5.3 that, if the decision had to be made under conditions of "uncertainty", then both the MAXIMIN and the MINIMAX REGRET criteria suggest that an exterior view is the sensible strategy. This raises the question: how does one justify the provision of an algorithm which can output a sectional view? If the strategy is never selected then there is no point in making it available. Such an algorithm may be justified as follows.

Let there be two classes of model:

M_1 an exterior orthographic view in the selected direction is comprehensible and can be clearly dimensioned and

M_2 an exterior orthographic view in the selected direction is either incomprehensible or cannot be clearly dimensioned, or both.

Let the two strategies be:

S_1 draw an exterior view, and

S_2 draw a sectional view.

Undoubtedly, the most desirable consequence is X_{11} and the least desirable is X_{21}. Arguably, a sectional view of the second class of model has a utility close to unity and an exterior view of this class has a utility which is assumed to be equally close to zero. Therefore, let:

$U(X_{11}) = 1, \quad U(X_{12}) = c, \quad U(X_{21}) = 0, \quad U(X_{22}) = 1 - c$

If the probability of the first class of model is P, the von Neumann-Morgenstern theory gives the following utilities for the two strategies:

$U(S_1) = P + (1 - P)c$

$U(S_2) = 0 + (1 - P)(1 - c)$

and these utilities are equal when:

$P = (1 - 2c)/(2 - 2c)$ or

$P = 1 - .5/U(X_{22})$

For example, if $U(X_{22})$ were subjectively set at 0.9 then the strategies are equally attractive when $P = 4/9$. Qualitatively, if the utility of drawing a sectional view of the second class of model is rated highly and the odds favour the second class of model then the drawing of a sectional view is an attractive strategy. This is precisely the situation for manual draughting and hence the provision of an algorithm for sectioning is justified on both empirical and theoretical grounds.

There is another new factor in the design of Algorithm D which was not present in those described above. Hitherto, the computational costs of the various strategies proposed have been roughly equal. When choosing between an exterior view and a sectional view, the additional cost of the latter must be taken into account. In the terminology of Decision Theory: the utility of a particular combination of strategy and class of model must now be related to both the usefulness of the consequence and the cost of achieving it.

If an algorithm for the production of sectional views is available, then the primary criterion for employing it must be the comprehensibility of its output when compared with an exterior view. If a definitive working drawing is required and time is not of the essence then the unit computational cost of operating such an algorithm is of secondary importance. If a new engineering drawing is required rapidly by an on-line user, then the question of computational cost (and hence, speed) is of greater significance.

In practice, it is the relative cost of producing a sectional view, compared with an exterior view, which is the important factor. The absolute cost of computing an exterior view depends directly on the cost of operating the hidden-surface algorithm which is described in Section 8.2. In order to output a single-plane sectional view, it is first necessary to modify the geometric model by computing the boolean difference of the model and a half space whose surface is the section plane. An examination of the techniques descibed by Braid [op cit] indicates that this would be approximately twice the cost of an exterior view.

Therefore, the design of Algorithm D is constrained by the unfortunate situation whereby much of the quantitative evidence for or against a sectional view may only be obtained by generating an exterior view of the model. If this evidence is overwhelmingly in favour of a sectional view then the computational cost of the final output will be three times the cost of outputting the exterior view (which has been analysed and rejected). Such a cost multiplication is most significant if an on-line user is waiting for the working drawing, and so this factor is included in the decision table which is set out below.

If the evidence from the hidden-surface algorithm favours a sectional view, it is probable that the exterior view will include a significant quantity of hidden-detail, some of which will require dimensioning. The policy of removing non-essential hidden-detail, although attractive for manual draughting, is illustrative of what has been dubbed "the Concorde fallacy". Colman [op cit] defines this as: "Continuing to invest in a project simply because so much has already been spent on it." If there is hard evidence that drawing a sectional view (either single or multiple plane) will produce a clearer dimensioning scheme, then there can be no justification for investing in improvements to the exterior view.

The selection of essential hidden-detail may be considered if there is no hard evidence of the superiority of a sectional view, or if the sectional view itself includes confusing hidden-detail. It is shown in Section 8.4 that such a procedure may incur abnormally high additional costs. The one Strategy that may be justifiable is to suppress all hidden-detail which is not dimensioned.

The important initial input to Algorithm D is therefore whether or not the user is on-line. If the ultimate user is off-line then the hidden-surface algorithm is executed and provides data for later decisions. Otherwise, the most highly ranked normal surface with its "open" side towards the view point is the only one to be processed by the hidden-surface algorithm and the number of visible Nodes/Centres is counted. If this number exceeds a fraction (Q, which is set by experiment) of the total for the surface, then the decision is to draw an exterior view. Otherwise, the Nodes and Centres of surfaces that have lower relative distances than the surface of interest are checked for superposition with surfaces that have higher relative distances. If a significant fraction (R, set by experiment) are not superposed, then the decision is for an exterior view. Otherwise, the model is sectioned at the surface of interest.

SPECIFICATION FOR ALGORITHM D

Constants for prototype implementation:

Critical fraction for visible Nodes, Q = 1/2

Critical fraction for superposed Nodes, R = 2/3

Critical weighted hidden Node score, Gmax = 100

Experimental Weightings, W, for surfaces:

Hard interface	10
Soft interface	6
Guide	4
Dam	3
Boundary	2
Signal	1

Operations:

7.3.1 If user is off-line go to Step 7.3.8

7.3.2 Arrange normal surfaces with their open sides towards the viewing direction into hierarchical order. Select the most highly ranked normal surface, N, or, if there is more than one, the surface with the lowest relative distance. Count the total number, P, of Nodes and Centres surface N.

7.3.3 Execute the SUPERMODEL hidden surface algorithm for surface N only and count the number, V, of visible

Nodes and Centres. If V/P> Q then go to Step 7.3.7.

7.3.4 Count the total number, T, of Nodes and Centres on all surfaces with a lower relative distance than surface N. Compute the number of these Nodes/Centres, S, that are superposed on the surface N and those with a higher relative distance. If S/T < R then go to Step 7.3.7

7.3.5 Position a half-space normal to the viewing direction and just before surface N. Compute the boolean difference of this half-space and the total geometric model. Pass the modified geometric model to Algorithm E (Section 7.4)

7.3.7 Use Hidden-surface algorithm (Section 8.2) to draw an exterior view. Exit.

7.3.8 Execute SUPERMODEL Hidden-surface algorithm for the complete geometric model. For each of the n normal surfaces with their open sides towards the viewing direction count the number of hidden Nodes, H_i, that are not superposed with visible Nodes. Compute the weighted hidden Node summation:

$$G = \sum_{i=1}^{n} W_i H_i$$

If G < Gmax then go to Step 7.3.7

7.3.9 Use the same criteria as in Step 7.3.4 to select the surface, N. Go to step 7.3.4

7.4 Algorithm E

The purpose of this algorithm is to make detail decisions about a sectional view. Algorithm D (Section 7.3) is designed to output either an exterior view of the model or to select a half space at an appropriate section plane and compute its boolean difference with the model. This boolean difference model is the input to Algorithm E.

In some cases the input model will be quite satisfactory for dimensioning but there can be cases where there are other normal planes requiring dimensioning which are still substantially hidden. The reason is that the object is of such internal complexity that two or more sectional planes are required in order to generate a suitable view for dimensioning.

A decision is required on whether to generate a compound section, to draw an entirely separate sectional view on a plane different from the input model, or to employ some other technique. British Standard 308: Part 1 contains recommendations for various strategies to overcome these difficulties, such as revolved or removed sections or the drawing of parts located in front of a cutting plane using chain lines.

The strategies available to a decision-maker are:

S_1 Do nothing. Leave the input model as it is.

S_2 Remove another part of the model to reveal normal surfaces that require dimensioning

S_3 Generate a new, additional boolean difference model

S_4 Draw a revolved section, superimposed on the view from which the section would normally be projected

S_5 Draw a removed section, displaced slightly from the view from which the section would normally be projected

S_6 Make an irregular break in the model so as to reveal the hidden features that require to be dimensioned

If there are no other normal hard interfaces in the viewing direction then the model clearly requires no further processing. Otherwise, it may be necessary to consider further sectioning if there are several "unclear" hard interfaces or if such a surface is very complicated or both. The presence of one unclear hidden hard interface is evidence which argues strongly in favour of an additional section. The following classes of model are used to set up a utility matrix for the algorithm.

M_1 No other normal hard interfaces

M_2 One "unclear" normal hard interface

M_3 One "clear" normal hard interface

M_4 Several normal hard interfaces with at least half of the total number of Nodes visible

M_5 Several normal hard interfaces with less than half of the total number of Nodes visible

A utility matrix for Algorithm E is shown in Table 7.1 The author's subjective assessment of utilities is based upon the following assumptions:

a) Strategies S_2 and S_3 will incur approximately equal computational cost.

b) Strategies S_4 and S_5 will be exceedingly difficult to implement and costly to operate.

c) Strategy S_6 will be marginally more expensive to operate than S_2 and S_3.

The relative values of the Utilities given in Table 7.1 have little influence on the design of a practicable algorithm. As the class of model that is presented may be ascertained with certainty, the decision making is reduced to selecting, from the appropriate column, the strategy with the highest utility.

Algorithm E has been not been implemented but the above analysis is intended as a basis for further work. In the case of the other algorithms that have been described, sufficient progresss has been made to analyse

Strategy	Class of model				
	M_1	M_2	M_3	M_4	M_5
S_1	1	.5	1	.9	.5
S_2	0	.8	.1	.6	.6
S_3	0	.7	.1	.2	.5
S_4	0	.1	.1	.2	.2
S_5	0	.2	.1	.2	.2
S_6	0	.6	.2	.4	.7

Table 7.1 Utility matrix for sectioning strategies

the geometric model and to select exterior and single-plane sectional views.

Various problems posed by the internal complexity of an engineering component have been examined. The algorithms described in this Chapter are dependent upon the output from the hidden-surface algorithm that is described in Chapter 8. The SUPERMODEL hidden-surface algorithm is designed specifically to produce orthographic views for dimensioning and this aspect is emphasized in the following Chapter.

CHAPTER 8

PREPARATION OF ORTHOGRAPHIC VIEWS

The selection of combinations of exterior views of a solid object is discussed in Chapter 6. The criteria for drawing a sectional view and the associated question of where to position sectional planes are examined in Chapter 7. It is now necessary to consider the actual process whereby a single orthographic view may be output by processing the geometric model.

The basic process is the classical "Hidden-Surface" problem, the literature of which is reviewed in the first section below. Some details of the special Hidden-Surface algorithm that was written to process SUPERMODEL data are given in Section 8.2.
The three special problems associated with this project are then addressed. The output from the hidden-surface algorithm is not always well-suited for inclusion in a working drawing. Section 8.3 discusses decisions which are required in order to clarify the orthographic view or to make it conform more closely to draughting practice.
The theme of decision-making is continued in Section 8.4, where the selection of essential hidden detail is discussed, and Section 8.5, where some questions relating to the cross-hatching of sectional views are addressed.

8.1 Hidden-Surface Algorithms

Sutherland, Sproull and Schumacker [40] presented a taxonomy of Hidden-Surface Algorithms in which they refer to three primary classes: those that compute a solution in "object-space", those that perform calculations in "image-space" and those that work partly in each. Those in the third group were referred to as "list-priority" Algorithms.

The "object-space" algorithms compute a solution which, within the limits of the precision available in the particular computer, is "exactly" what the image should be. Such a solution will be correct even when it is greatly enlarged. The "image-space" algorithms compute a solution which is suitable only for the resolution of the screen (or incremental plotter) on which it is to be displayed. The goal of an "image-space" Algorithm is simply to compute an intensity for each of the hundreds of thousands of resolvable dots on the screen.
The "list-priority" Algorithms work partly on object-space and partly on image-space. For the purposes of this project, the object-space Algorithms are of the most direct interest. A geometric model is to be viewed in two or more directions and these views will be displayed in orthographic projection with various scale factors. It is essential to determine orthographic views which are as precise as the computer accuracy will

permit.

The survey by Sutherland, Sproull and Schumacker places emphasis on the distinction between "object-space" and "image-space" algorithms. Foley and Van Dam [41] have opted to classify hidden-surface algorithms according to the strategy followed. In particular, they refer to one group as "area-subdivision" algorithms. The original area-subdivision algorithm was published by Warnock [42] in 1969 and it has since been adapted and improved by several other workers.

An area of the projection plane image is examined by, in Warnock's terminology, the "thinker" part of the algorithm. Then, if the "thinker" can decide which polygon is visibile in the area, it is displayed. Otherwise, the area is subdivided into four smaller areas and the "thinker" examines each of these in turn. Ultimately a decision about what to display will be possible but, using the image-space approach, the decision may simply be to illuminate the one pixel which then lies inside a very small sub-division of the total picture.

Such an algorithm was seriously considered for this project in spite its "image-space" classification. The Edges that are shown as a full line or a dashed line on an orthographic view can be extracted from the output of an area-subdivision algorithm. A technique that was investigated was to fill in unresolved parts on an Edge by assuming coherence (the tendency of an Edge to continue until it meets a Node) in zones where a great deal of area-subdivision had taken place. The advantage of this approach would have been the ease with which curved surfaces could be processed, but it proved to be very slow and was finally abandoned when another method showed better promise for this particular project.

Sutherland, Sproull and Schumacker [op cit] proposed a sub-division of the class of object-space Algorithms. All such algorithms test relevant edges to determine what parts of the edges are visible but the invisibility criteria are different. Algorithms in the first sub-division (termed "volume criterion") test whether an edge is obscured by the volume of an object that lies between the edge and the viewpoint. The second type of algorithm tests edges against edges and often makes use of the tendency of an edge to have coherent visibility, particularly at the nodes that terminate the Edge. This type is called an "edge criterion" Algorithm.

Roberts [43] was the first to publish a solution to the hidden-line problem. His algorithm tests each relevant edge to see if it is obstructed by the volume occupied by some object in the environment. The testing technique is to set up a parametric equation for a line from a point on the edge to the viewpoint and discover if any of the edge is obscured by the object. The algorithm requires that objects be planar and convex.

The "Edge-Intersection" Algorithms discussed by

Sutherland, Sproull and Schumacker are those by Appel [44], Loutrel [45] and Galimberti and Montanari [46]. Appel's algorithm was found to be particularly relevant to this project as it is typical of a whole group that compute line drawings and some of the techniques employed were adapted for the SUPERMODEL hidden surface algorithm .

Appel defines the "quantitative invisibility" of a point as the number of relevant faces that lie between the point and the viewpoint. The problem is solved by computing the quantitative invisibility of every point on each relevant edge. Use is made of the concept of edge coherence to reduce the time taken to compute a solution.

The "quantitative invisibility" of an edge can change only where the projection of that edge onto the picture plane crosses the projection of some contour edge. At such a crossing point, the quantitative invisibility must be incremented or decremented by one. When all contour edges have been considered the relevant edge has been divided by the intersections into a number of segments. If the quantitative invisibility of the initial vertex of the edge is known then the visibility of each segment can be determined.

It is instructive to note how these concepts and techniques permeate later work. Not surprisingly the terminology is often rather different: Loutrel refers to an "order of invisibility" which is equivalent to

Appel's "quantitative invisibility". Galimberti and Montanari refer to the "nature" of a point when deciding whether it is visible or obscured by the set of faces that have been processed.

The algorithm required for this project must be able to process curved surfaces, but all those referred to thus far are for planar polygonal faces. Objects containing curved surfaces must be approximated by many small facets before any of the above algorithms can be used. This creates a large number of additional planes to be stored and processed. Worse, these facets then have to be removed in order to generate a credible orthographic view.

Braid [47] published one of the earliest papers to refer to a broader class of objects. The algorithms descibed in his paper restrict object faces to lie in surfaces which are either planes or circular cylinders. He considers two types of Edge: those formed by the intersection of two planes and those at the intersection of a plane with a cylinder.

Weiss [48], Woon and Freeman [49], and Mahl [50] extended the class of objects still further by developing algorithms for removing the hidden surfaces of objects defined by quadratic surfaces. These surfaces are in general non-planar and have the form:

$$Ax^2 + By^2 + Cz^2 + 2Dxy + 2Eyz + 2Fzx + 2Gx + 2Hy + 2Jz + K = 0$$

Familiar objects composed of quadratic surfaces include the closed cylinder (3 surfaces), the closed cone (2 surfaces) and the sphere (1 surface). There is a small, but potentially confusing difference in terminology within this topic. Some workers describe the surfaces as "quadric" whereas Foley and Van Dam term them "quadratic"; the latter is preferred on grounds of comprehensibility.
Algorithms for processing quadratic surfaces always have to address the problem that the intersection of two such surfaces yields a fourth-order equation in x,y and z. The roots of the equation must be found numerically and this is very demanding of computing time. Levin [51] reduced these fourth-order problems to second-order by parameterizing the intersection curves. This not only facilitates an analytical solution but also requires a mere twenty or so numbers to represent an intersection curve, rather than storing the coordinates of several dozen points on such a curve.

Apart from listings of source-code in some theses, very little has been published on the actual implementation of the above algorithms. Presumably, this is because of the commercial value of much of the software. The details that are available generally relate to early work on rather simple algorithms of limited application. The algorithms presented by Angell[op cit] are typical: the most advanced will handle only convex polyhedra.

Each surface of the model must consist entirely of straight Edges and have no re-entrant angles and no holes. Angell's commentary on the "General Hidden Line Algorithm" (p.89) is instructive and provides a usable starting-point. The actual algorithm appears to based upon work by Appel[op cit] but is inefficient because the homogeneous coordinates of a surface are computed each time that the surface is processed. The algorithm has clearly been tailored to the arithmetic precision of a particular computer-system.

8.2 SUPERMODEL hidden surface algorithm

It is noted in Section 6.4 that, if a SUPERMODEL surface is orientated with its solid side towards the viewer, then none of it can be seen. Only those surfaces whose normal vectors are inclined at an angle of absolute value less than 90° to the viewing direction need to be processed by the hidden surface algorithm. Such surfaces are referred to as "open" in the description below.

The basic process may be described quite simply. Each open surface is examined by considering its Edges one at a time. Initially, it is assumed that the entire Edge is visible. Each Edge is compared with every other surface of which the model is composed. The Edge/surface comparison is illustrated in Fig. 8.1, where the Edges of the open, cross-hatched surface are compared with surface ABCD. There are three possible results:

Cross-hatched surface is 'open'

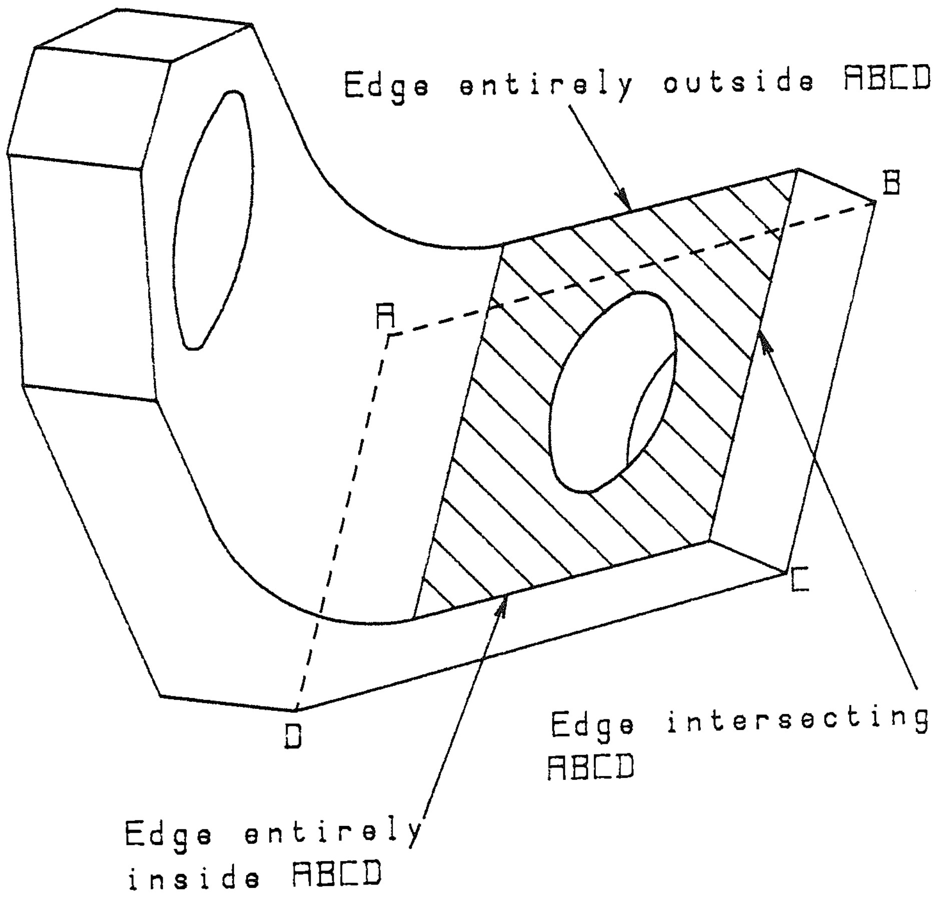


Fig. B.1 Edge/surface comparisons for hidden surface algorithm

Cross-hatched surface is 'open'

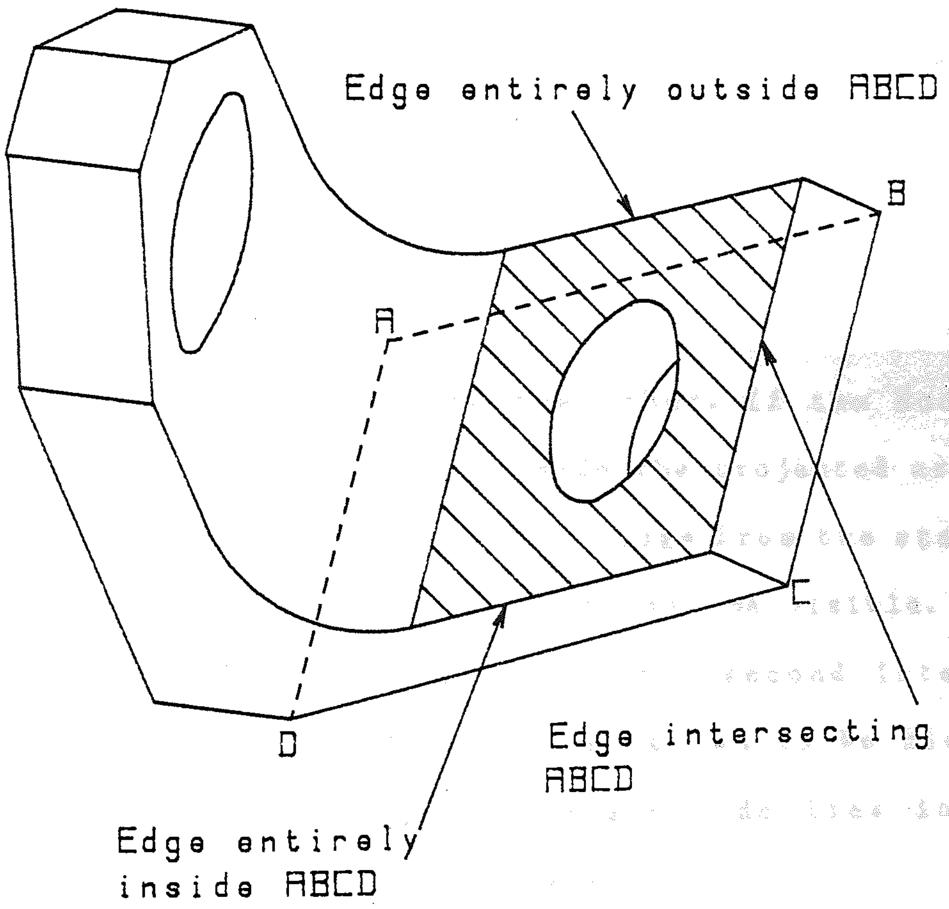


Fig. B.1 Edge/surface comparisons for hidden surface algorithm

8.2.1 The Edge lies entirely outside the projected area of the surface.

8.2.2 The Edge lies entirely within the projected area of the surface.

8.2.3 The Edge intersects the projected area of the surface at one or more points.

If 8.2.1 applies, then no change needs to be made to the current state of visibility of the Edge. If 8.2.2 applies and the surface is nearer to the viewer than the Edge, then the entire Edge is known to be hidden. Otherwise, there is no change to the current state of visibility of the Edge. If the surface is closer to the viewer than the Edge and 8.2.3 applies then it is necessary to arrange the intersections in sequence from one end of the Edge to the other. If the Node at the start of the Edge lies outside the projected area of the surface then that part of the Edge from the start to the first intersection point may still be visible. From the first intersection point to the second intersection point (or the finish Node) is known to be hidden. The opposite is true if the start Node lies inside the projected area of the surface.

When the Edge has been compared with every surface, its true status is known and it may either be drawn as a full line, a dashed line, or a set of alternating full and dashed lines. The algorithm then processes the next Edge from the current "open" surface until there are no more Edges left to process.

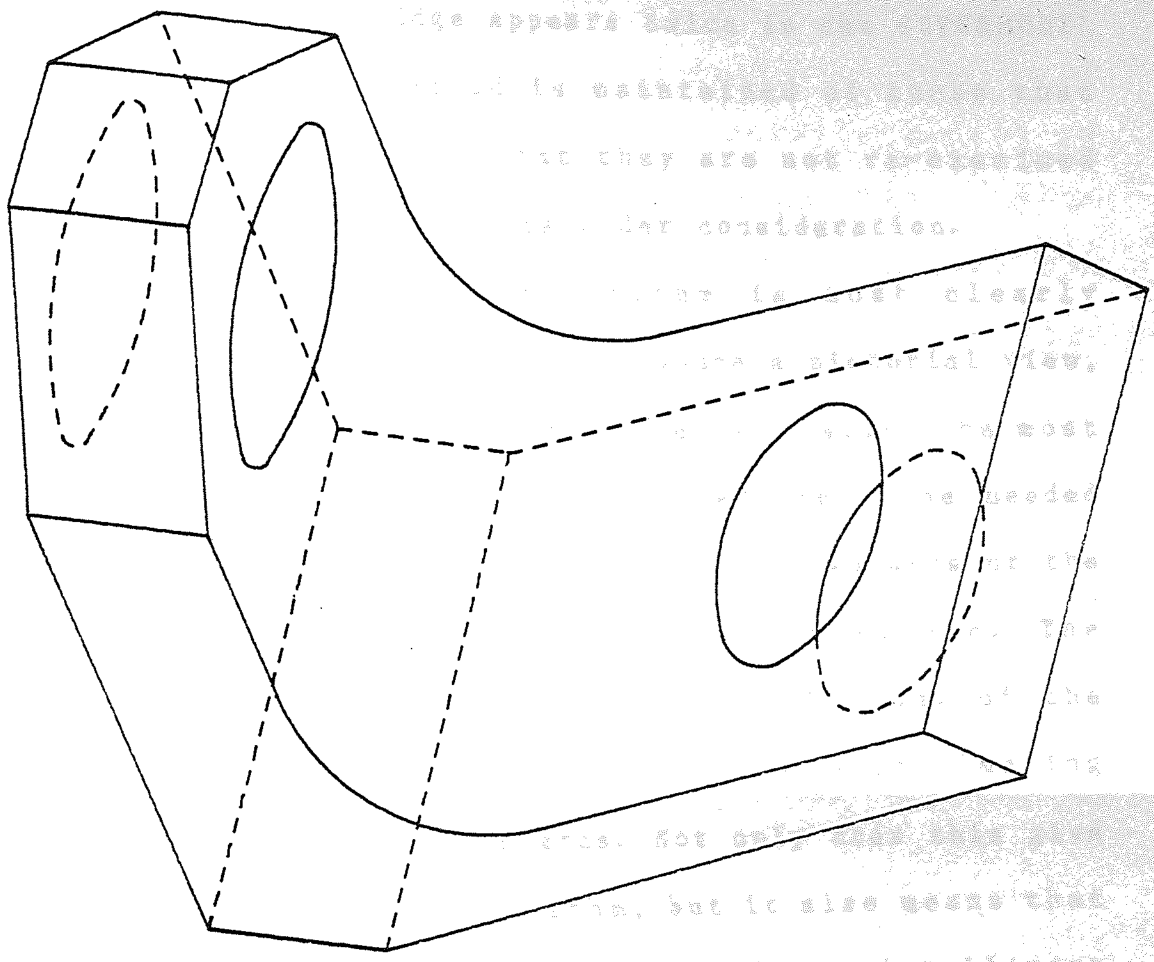


Fig. 8.2 Output from SUPERMODEL hidden surface algorithm

Next, the algorithm examines the remaining model surfaces and, if they are "open", their Edges are processed. As every Edge appears twice in the SUPERMODEL data structure, a record is maintained of those that have been processed so that they are not re-examined when the adjacent surface is under consideration.

The output from this algorithm is most clearly illustrated when it is used to produce a pictorial view, such as that shown in Fig. 8.2. Theoretically, the most complex geometrical routines required are those needed to compute the apparent intersection of two arcs or the apparent intersection of a line and an arc. The algorithm whose output is shown avoids many of the problems noted by Levin [op cit] by approximating ellipses to four circular arcs. Not only does this give a simpler and faster algorithm, but it also means that the output can be easily processed by an intelligent plotter. As the objective is to produce a view for an engineering drawing rather than an high precision curve, this tactic is entirely justified.

A line/line intersection is easily computed by using homogeneous coordinates. The homogeneous coordinates of each flat surface are computed once, at the start of the routine, using the method given in Section B.3.5.

The view shown in Fig. 8.2 would not be suitable for dimensioning as none of the surfaces appears in its true shape. Each Edge is clearly shown without being covered (partially or completely) by closer Edges.

This is not the case when an orthographic view is drawn by selecting a viewing direction which is normal to many of the flat surfaces. Such views inevitably result in the superposition of Edges, a possible source of confusion if their computed positions differ by an amount that is within the resolution of the plotter that produces the final drawing. Another difficulty is that even "correct" orthographic views may exhibit features which a human draughtsman would modify in order to clarify the view. It is convenient to post-process the output from the hidden-surface algorithm with software that is designed to clarify it, a process which is now addressed.

8.3 Algorithm F

The purpose of this algorithm is to clarify the output from the hidden-surface algorithm. There are several reasons for doing this but four in particular have been selected for the impact that they have on the quality of the final drawing:

8.3.1 Edges which should be superposed cannot always be computed with sufficient precision and are resolved by the plotter into apparently separate features. This algorithm is required to reintegrate them.

8.3.2 The recommendation contained in British Standard 308: Part 1 that visible outlines should be shown with lines that are from two to three times the thickness of projection lines, dimension lines and hidden-detail lines. One way to achieve this is by instructing a multi-pen plotter to use a suitable pen but the effect may be achieved by drawing parallel edges very close together. This method is preferred as it is suitable for all plotters.

8.3.3 The recommendation contained in British Standard 308: Part 1 that exceptions to the general rule for indicating sections should be made where the cutting plane passes longitudinally through ribs or webs. When there is only a small amount of material behind the section plane it is considered that a misleading impression of bulk is given if the entire area is cross-hatched. The convention is to indicate such features in outside view, even though the cutting plane passes through them.

8.3.4 The recommendations contained in British Standard 308: Part 1 for the conventional representation of external and internal screw threads. A cylindrical surface that is input to SUPERMODEL may be designated as having a thread. This classification is ignored by the hidden-surface algorithm but acted upon by Algorithm F.

The problem of nearly-superposed Edges (8.3.1) is resolved by computing the homogeneous coordinates of each line and the rectangular coordinates of the centre together with the radius of each arc. If there is no difference between two such sets of parameters or if the difference could have arisen from the arithmetic precision of the computation, then the Edge with the greater relative distance is eliminated.

The thickening of visible edges (8.3.2) is achieved by making use of the parameters that were computed for 8.3.1. Each computed line is represented by two parallel lines whose displacement either side of the "true" line is set according to the fineness of the pen that the plotter uses. Typically, this displacement is equal to the average line-thickness. Each computed arc is represented by two concentric arcs whose radial displacement either side of the "true" arc is equal to the line displacement.

The clarification of ribs (8.3.3) calls for some fine-tuning in the selection of what is a reasonable thickness to qualify as a rib. Fundamentally, the decision is a simple one as the putative rib has to pass three tests:

a) It must be a flat surface with its open side away from the viewing direction,

b) It must lie entirely within the area that has been cut by the section plane, and

c) Its relative distance (section plane = 0) must not exceed some critical value, R, dubbed the "rib factor".

If all three tests are passed then the outline of the rib is converted from dashed edges to continuous edges. This adds one or more circuits to the graph in the surface of the half-space that represents the sectioning plane, and it effectively changes the output from the cross-hatching routine which is described in Section 8.5.

The conventional representation of screw threads (8.3.4) is implemented by looking up the surface orientation that was input to SUPERMODEL. A cylindrical surface that was specified as convex implies an external screw thread which, when viewed axially, is represented by two concentric circles, the outer having a small gap. When viewed longitudinally, the thread is represented by parallel lines. There are similar conventions for internal screw threads.

8.4 Algorithm G

The purpose of this algorithm is the selection of essential hidden detail. Fig. 7.2 illustrates the confusing nature of densely-packed hidden detail. It is a simple matter to detect if the output from the hidden-surface algorithm contains such densely-packed detail but decision-making algorithms for resolving the problem are necessarily complex.

Two approaches to the problem were considered:

a) Avoidance of ambiguity. Fig. 7.1 shows how various interpretations of the drawing are possible if the hidden detail is insufficient. In order to implement a strategy based upon avoiding this ambiguity, it would be necessary to investigate the interpretation of the various views as a solid object. Work by Preiss [52] on edge-following techniques suggests that, although technically possible, the computational cost involved would be prohibitive.

b) Provision for dimensions. Arguably, an edge need only appear on an orthographic view if it joins two Nodes, one or both of which are to be dimensioned. These Nodes may be connected to other visible edges, in which case the hidden-detail edge in redundant for dimensioning purposes, although not necessarily for interpretive purposes.

Similarly, if the position of the Centre of an arc is to be specified and no concentric visible arcs are present, then something of the arc should be shown. Hence, a practicable criterion for retaining hidden edges in a region where they are densely packed is the dimensioning requirement.

The dimensioning algorithm described in Chapter 10 is the arbiter for hidden detail. The surfaces that require to be dimensioned explicitly are processed first and the views on which they appear in their true shape have hidden detail added each time that a Node or Centre could not be indicated otherwise. Lower-ranked surfaces are dimensioned in order until all surfaces have been processed. At this point there may still be hidden edges, generated by the hidden-surface algorithm, which have not been added to a particular view.

Algorithm G then decides if this "missing" hidden detail may be added by investigating the density of hidden detail already in place. The decisions made are based upon three types of hard evidence against the proposed addition:

a) A "missing" line would be close to a parallel hidden-detail line.

b) A "missing" arc would be close to a concentric hidden-detail arc.

c) A "missing" edge would be tangential (or nearly so) to a hidden-detail edge.

8.5 Cross-hatching

Once decisions have been made concerning the angle and pitch of the hatching lines, the solution to any cross-hatching problem is completely constrained by the geometry of the cross-section. The selection of a suitable angle is the more difficult decision and will be addressed first.

Observation suggests that draughtsmen normally opt for an angle of 45^o (or 135^o) and only select another angle in order to make the hatching more distinctive. The initial choice of strategy is therefore between:

S_1 draw hatching lines at 45^o to the horizontal and

S_2 draw hatching lines at an angle other than 45^o

For aesthetic reasons, as well as clarity, hatching lines should not be parallel or normal to a substantial number of bounding lines. Circles and circular arcs need not be considered because of the constantly changing angle of intersection. Therefore, two classes of orthographic view are defined:

M_1 All views which are not in class M_2

M_2 A view in which the majority of straight lines are at an angle of between $40-50^o$ or between $130-140^o$

Employing the usual nomenclature of decision-making:

$U(X_{11}) = U(X_{22}) = 1$; $U(X_{12}) = 0$; $0 < U(X_{21}) < 1$

The probability of the first class of model is the greater of the two, hence, for the first strategy statistically to dominate the second (see Section 5.4) it is necessary that:

$U(X_{11}) > U(X_{21})$ which is true, and

$U(X_{11}) + U(X_{12}) > U(X_{21}) + U(X_{22})$ which is false.

This tends to the conclusion that the second Strategy is worthy of serious consideration. Hence, a procedure is required which will determine the angle of inclination of each straight line within the view. Fortunately, such a procedure is easily formulated and computationally inexpensive. If the view is a member of M_1 then the angle for cross-hatching is set to 45^o, otherwise it is set to zero.

Experience suggests that a reasonable pitch for the cross-hatching is 5% of the normal distance to be covered. Hence, the pitch of the cross-hatching is determined after the bounds of the cross-section have been computed.

The SUPERMODEL cross-hatching routine was designed to process a line/arc graph of any complexity that can be stored within one graphics surface. It operates subject to the following restrictions:

8.5.1 The complete surface is cross-hatched. If a section through a solid object is to be processed then the sectional view only must be drawn in a particular surface. Additional continuous lines must not be present.

8.5.2 The routine ignores dash, chain and arrow lines and arcs. Such features may be included in a surface that is to be cross-hatched but only the continuous lines and arcs are considered to define the area(s) that are to be processed.

8.5.3 The continuous lines and arcs must "bound" one or more closed shapes.

8.5.4. The outermost boundary is regarded as the boundary between the material cross-section and the surrounding medium. Hatching lines terminate at this outer boundary.

Fig. 8.3 shows a typical surface to be cross-hatched. It

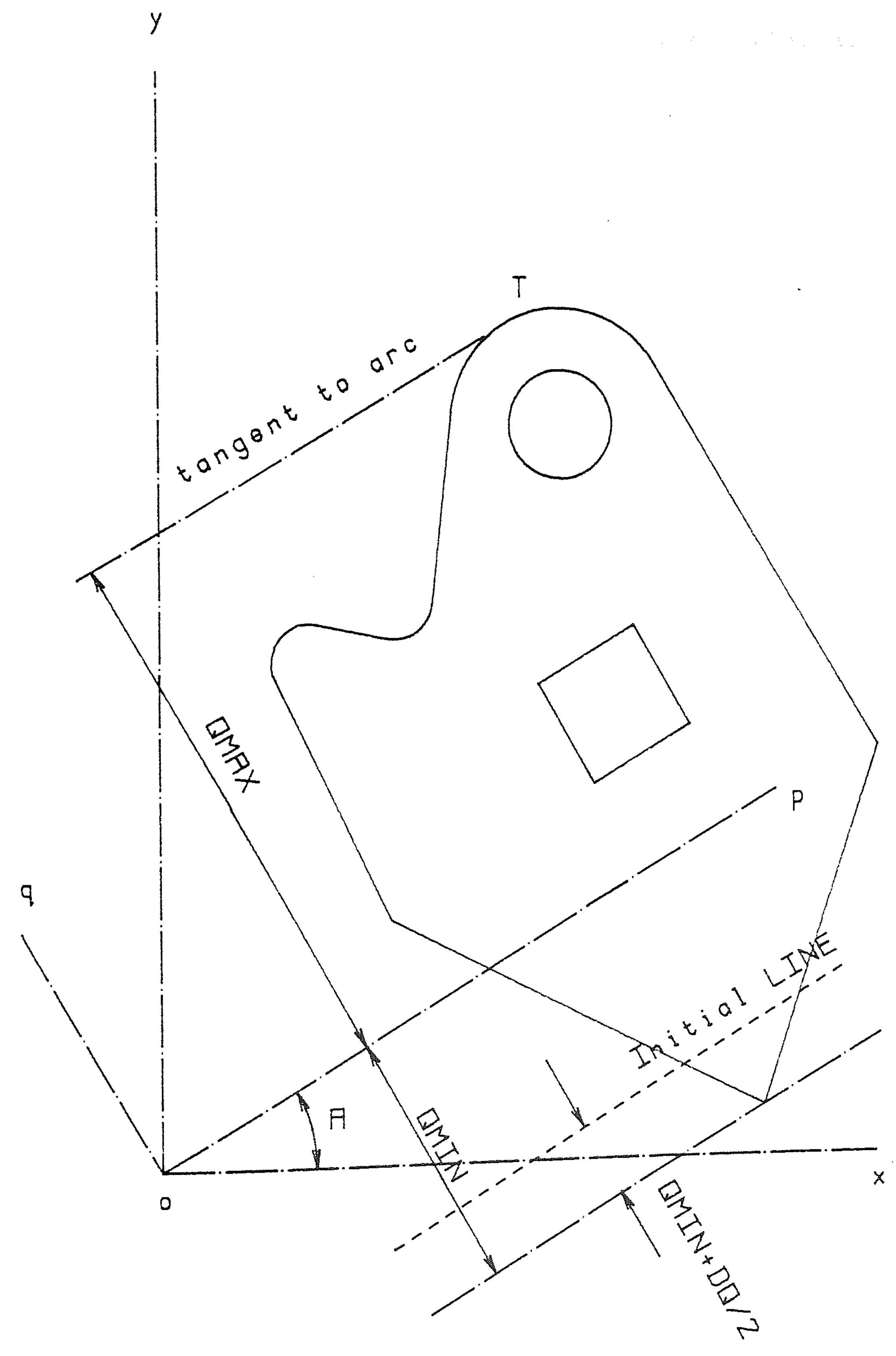


Fig. 8.3 Parameters for cross-hatching algorithm

consists of an outer boundary with two holes. The inputs to the following Algorithm are the Geometry and Topology arrays of the active graphics surface plus the angle, A, and pitch, DQ, of the hatching lines. The Topology array is first examined to determine if the surface can be cross-hatched. This is done by checking that it consists of one or more closed loops in which the last Node of the loop is the same as the starting Node. Complete circles are a special, acceptable case.

The surface is then processed in order to compute the values QMIN and QMAX shown in Fig. 8.3 This is done by transforming the coordinates of each Node to axes at angle A. It is also necessary to examine the points of tangency of a hatching line and each circular arc. Point T on Fig. 8.3 is such a point and happens to define the value of QMAX.

The pitch of the cross-hatching, DQ is set at a constant value of 0.05(QMAX - QMIN) and the algorithm then operates as follows:

```
STEP 1. Set initial position of hatching line at QMIN +
DQ/2
STEP 2. Compute homogeneous coordinate for an infinite
LINE at current hatching line position.
STEP 3. Set Edge counter I = 1
STEP 4. Set intersection counter N = 0
STEP 5. If Ith Edge is an arc go to Step 9
STEP 6. Compute intersection of LINE and Ith Edge (X,Y)
STEP 7. If intersection does not lie on Edge go to Step 14
STEP 8. Add one to intersection counter $(N = N + 1)$ and update intersection arrays $U(N) = X$, $V(N) = Y$.
    Go to Step 19
STEP 9. If LINE does not intersect Ith Edge go to Step 14
STEP 10. Compute one intersection of LINE and Ith Edge $(X,Y)$
STEP 11. If intersection does not lie on Edge go to Step 13
STEP 12. Add one to intersection counter $(N = N + 1)$ and update intersection arrays $U(N) = X$, $V(N) = Y$.
STEP 13. If only one intersection has been processed go to Step 10
STEP 14. Add one to Edge counter $(I = I + 1)$. If I does not exceed the number of Edges go to Step 4
STEP 15. If number of intersections, $N = 0$, then go to Step 18
STEP 16. Sort intersections into order according to their distance from PMIN
STEP 17. Draw a line from 1st intersection to 2nd, from 3rd to 4th....(N-1)th to Nth
STEP18.Increment position of hatching line $(Q = Q+DQ)$
STEP 19. If Q is less than QMAX then go to Step 3, otherwise exit.
```

Once the various views of a working drawing have been decided upon and most of the details of each view have been computed, the actual process of dimensioning can begin. Some hidden detail may not be added until the process of dimensioning is nearly completed.

The manual process of dimensioning is more akin to a craft than a science and, as such, it is difficult to analyse the steps and decisions which make up the process. The objective of the work reported in Chapter 9 was to generate feedback from experienced draughtsmen and thereby provide data for the design of an automatic dimensioning algorithm.

CHAPTER 9

PLACEMENT OF DIMENSIONS ON A SINGLE VIEW

9.1 Scope of the dimensioning algorithms

The process of producing a dimensioned drawing may be stated in the following way. The primary input represents a solid object which is composed of flat and conical surfaces. The secondary inputs are data concerning its function, manufacture and inspection. The outputs are to be orthographic views (discussed in Chapters 6, 7 and 8) plus dimension lines and text which completely define the geometry of the object and facilitate its manufacture and inspection.

Two alternative approaches to the problem were investigated:

a) An orthographic view is regarded as a "line/arc Graph", output by the hidden-surface algorithm, in which constraints must be placed upon the position of each Node and Centre by specifying dimensions.

b) Each surface within the model is regarded as a parameterised shape and each parameter is to be specified by means of dimension lines, some of which may be common to two or more shapes.

The first approach is discussed in this Chapter where details of an interactive algorithm is given and its implementation is described. The second approach is treated in Chapter 10.

The software described in this Chapter (Algorithm I) is designed to arrange dimensions around one orthographic view of a solid object that has been modelled using the software described in Chapter 3. The purpose served by Algorithm I was the generation of useful feedback from professional users.

The dimensioning process is without reference (by the computer) to adjacent orthographic views. The view that is dimensioned is a Graph whose Edges are straight lines, circular arcs and ellipses. The ellipses are the result of projecting a circle onto some non-parallel plane. They are deliberately excluded from the dimensioning process on the grounds that any feature should not be dimensioned on a view in which it does not appear in its true shape.

The dimensioning conventions adopted for the algorithms described both in this Chapter and in Chapter 10 are those recommended by the British Standards Institution [op cit]. These conventions are described in Section 4.2. The tolerancing of dimensions follows the recommendations of British Standard Specification 4500 [52].

Section 9.2 below examines the criteria which influenced the specification of an interactive procedure, Algorithm I. This was designed to assist an experienced draughtsman with the insertion of projection lines, dimension lines and toleranced dimensions. The specification is given in Section 9.3 and some sample output is illustrated. The way in which this algorithm was employed provided a useful insight into the relationship between the function and method of manufacture of a component and the dimensions by which it is specified.

Section 9.4 gives details of the conclusions that were reached after examining the feedback from users of the interactive dimensioning software. The conclusions were that:

a) no further work on the single-view approach to dimensioning should be undertaken but, more significantly,

b) the dimensioning process frequently reflects the procedures that were used to input surfaces to the geometric model.

9.2 Computer-assisted dimensioning of line/arc Graphs

Many objects are manufactured using dimensions taken from datum surfaces. Such surfaces are likely to be flat and either to form part of the boundary of the shape or else be physically accessible from the boundary. One of the objectives for the algorithm described in this section was to provide data on how the human operator selects datum surfaces.

Fig. 9.1 shows two dimensioning schemes for the same object. The upper view is dimensioned with its left-hand

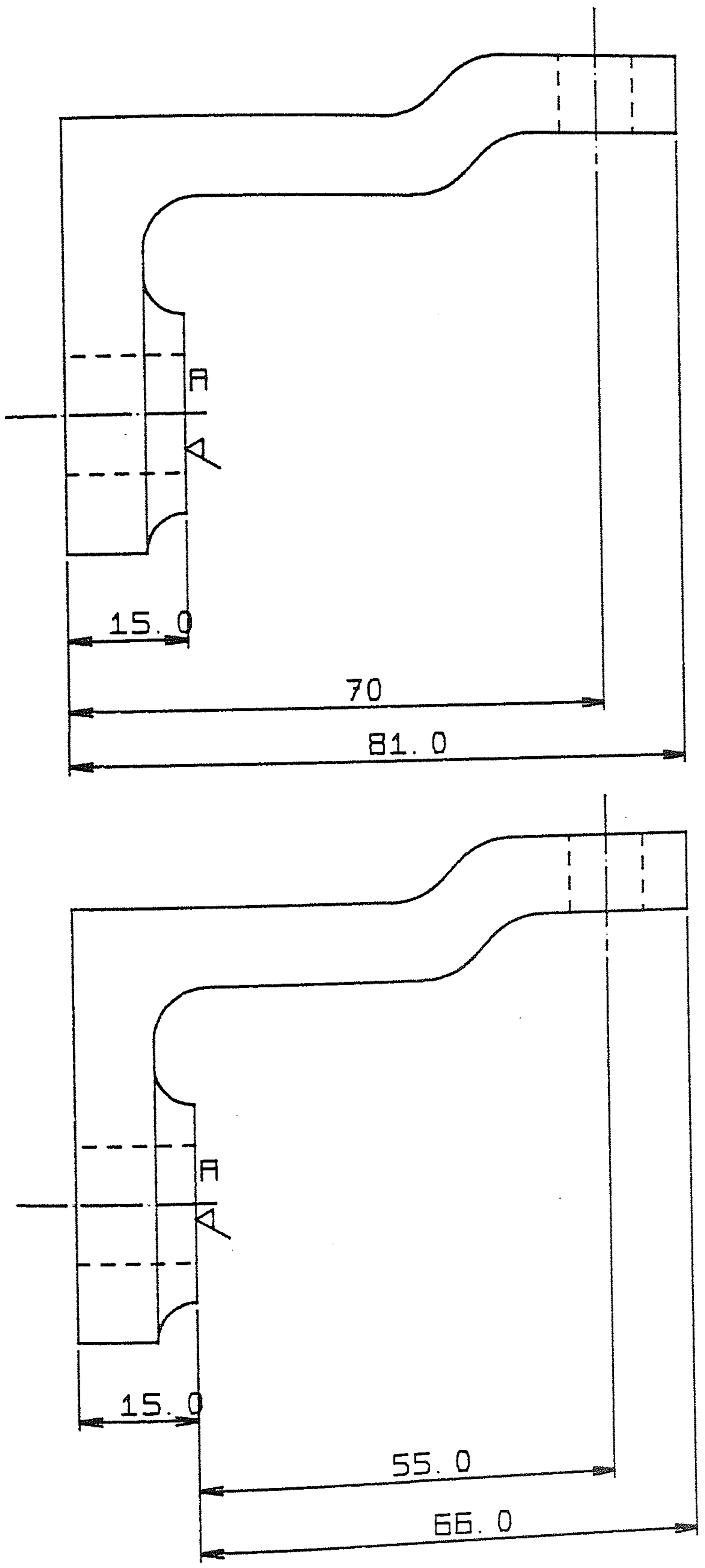


Fig. 9.1 Alternative dimensioning schemes for the same object

surface as the datum whereas, in the lower view, the machined surface A is the datum. The upper view would not be acceptable on a working drawing as distances from a machined surface need to be calculated, instead of directly read.

The sub-set of two-dimensional shapes that have been considered may be dimensioned by indicating the positions of their Nodes and Centres. An arc has two Nodes (a start-point and finish-point) and a Centre. In the case of a complete circle the start-point and finish-point are coincident and may be chosen arbitrarily unless some other Edge ends at this Node.

Assuming, for the moment, that the position of a Node or a Centre is to be described in terms of horizontal and vertical distances only, there are two Strategies that may be adopted:

a) the distance from some datum surface is given (a datum dimension) or

b) the distance from some adjacent Node or Centre is given (a size dimension).

Although practice varies, there is a tendency to avoid placing dimension lines within the boundaries of a view. Obviously, there will be cases where the projection lines start within the boundary of a view and would have to be extended excessively in order to clear the view. In such cases the dimension line may be placed within the boundary of the view. For clarity and aesthetic reasons the longer dimension lines are placed further from the view boundary than the shorter ones. These points are illustrated in Fig. 4.7

Evidence against dimensioning the end-Nodes of an arc is provided when the arc blends with its neighbouring line or arc. The following axiom is proposed and will be referred to as the Dormant Node axiom:

Any Node within the Graph which has $C^{(1)}$ continuity should not be dimensioned.

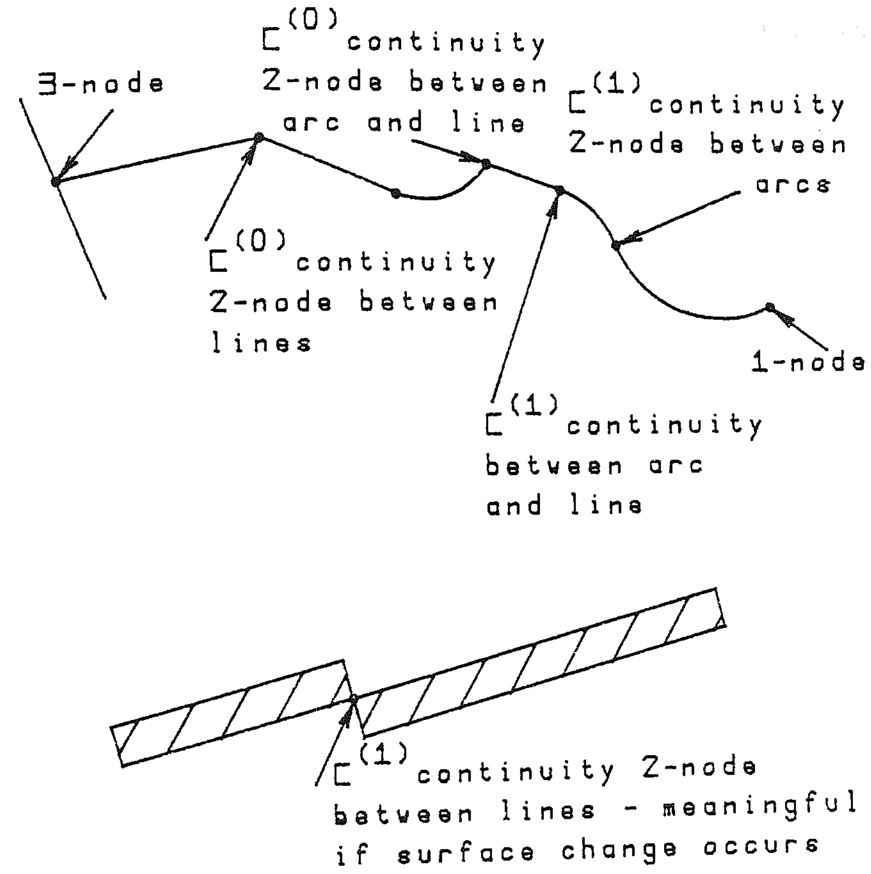
This situation usually occurs at a 2-Node, as illustrated in Fig. 9.2, but it can occur at Nodes of higher degree. The notation $C^{(i)}$ is used to describe continuity at a Node. A Node of degree two and above is $C^{(0)}$ continuous by definition because there is no discontinuity at that point.

It is said to be $C^{(1)}$ continuous if, in addition, the tangents to the two Edges are continuous. In other words, there is no change in the first derivative of the shape at this Node.

The 2-Node which is arbitrarily positioned on a complete circle has this degree of continuity and it has already been decided that it should not be dimensioned.

The corollary to the above is that all Nodes without $C^{(1)}$ continuity should be dimensioned.

A 2-Node with $C^{(1)}$ continuity can be completely ignored



$C^{(0)}$ continuity 2-node between arc and line
3-node
$C^{(1)}$ continuity 2-node between arcs
$C^{(0)}$ continuity 2-node between lines
1-node
$C^{(1)}$ continuity between arc and line
$C^{(1)}$ continuity 2-node between lines - meaningful if surface change occurs

Fig. 9.2 Continuity at Nodes

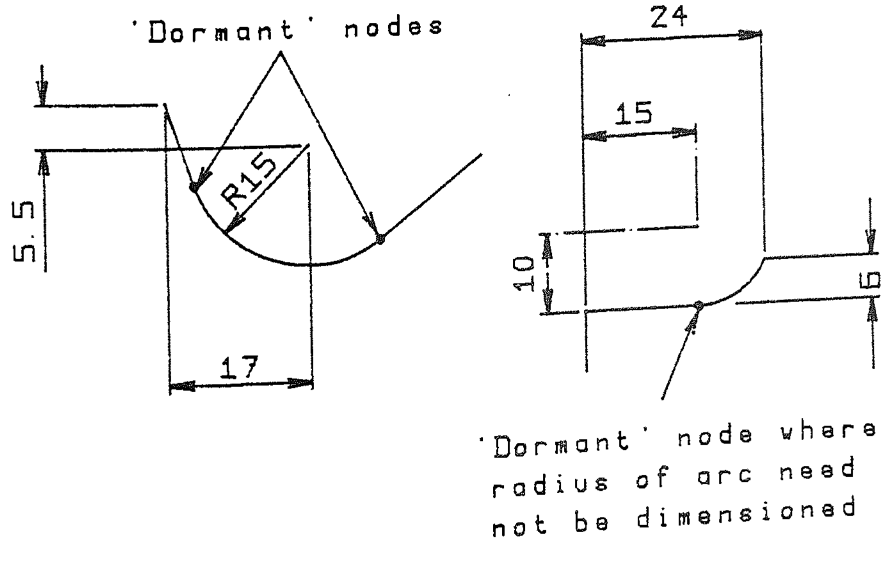


Fig. 9.3 Definition of 'Dormant' Nodes

if two straight lines meet there. It has arisen accidentally or erroneously in the course of creating the geometrical model. It is a point in 2-space where nothing happens. However, if the straight lines have additional attributes, such as roughness, machining instructions, etc. which are input to the dimensioning algorithm in addition to the geometric model, and one of these attributes changes at the Node then it is meaningful. Otherwise it need not be considered by the dimensioning algorithm.
If one or both of the Edges at a Dormant Node is an arc then the Centre of the arc is a point which must be dimensioned. Moreover, it will be necessary to state the radius of the arc unless the Node to which the arc is connected is dimensioned. These rules are illustrated in Fig. 9.3.

An interactive dimensioning algorithm is of little practical use if it is constrained to produce horizontal and vertical dimensions only. An auxiliary view typically shows the object with several parallel surfaces inclined to the horizontal. It is therefore necessary to examine the number of occurences of a particular angle in order to determine if it is of such significance that the dimensions should be orientated other than horizontally and vertically.

An algorithm for investigating the orientation of a view was designed. Its specification is as follows:

STEP 1. For each line within the graph determine the angle from the start point to the finish point.

STEP 2. For each arc within the graph determine the direction of the tangent at its start and at its finish point. Ignore complete circles (because their start/finish point is arbitrary).

STEP 3. Adjust each angle that has been computed so that it lies within the first quadrant. e.g. Angles of 45^o, 135^o and -45^o are all stored as examples of a 45^o edge or tangent.

STEP 4. Count the number of times that each significantly different angle occurs. A line scores two for the same angle and an arc scores one for the tangent angle at the start and one for the tangent angle at the finish.

STEP 5. Set "dominant angle" to zero. If a particular angle occurs more times than any other and the number occurances is at least one third of the total (a fraction determined by experiment) then reset "dominant angle" equal to this angle.

9.3 Algorithm I

By reference to the above criteria it was possible to specify an algorithm whose purpose is to assist the user to dimension a line/arc Graph. The specification given below is the result of several iterations.

SPECIFICATION OF INTERACTIVE DIMENSIONING ALGORITHM

9.3.1 Compute "dominant angle" using the algorithm above. If angle is not zero then invite user to accept or reject it. If applicable, rotate axes of coordinate-system through "dominant angle".

9.3.2 Compute the bounds of the two-dimensional shape: its minimum and maximum x coordinates and its minimum and maximum y coordinates, so as to define a rectangle inside which there are to be no dimension lines.

9.3.3 Invite the user to specify the Node or Centre which is to be used as the datum point for dimensioning (XREF,YREF).

9.3.4 Compute the coordinates of the centroid of the above rectangle (XMEAN,YMEAN).

9.3.5 Arrange the x coordinates in order of magnitude, ignoring points on the circumference of complete circles and any Dormant Nodes. Maintain a record of the addresses of all circles and circular arcs.

9.3.6 If more than one x coordinate has the same value, determine the minimum (YL) and maximum (YU) values of (y-YMEAN) for all such x coordinates. Set all such y coordinates that are less than YMEAN equal to YMEAN+YL and all that are greater than YMEAN to YMEAN+YU.

9.3.7 Starting at the first Node or Centre whose x coordinate value is not equal to XREF, draw a vertical projection line from a Node or a vertical chain line from a Centre that is not coincident with a previously processed one, i.e. ignore all repetitions of (x,y) coordinates. The distance that the projection (or chain) line extends beyond the boundary of the view is to be directly proportional to its position in the x coordinate ranking. A line which starts from a Node situated below the centroid of the enclosing rectangle is to be projected downwards. A line which starts from a Node situated at the same level or above the centroid is to be projected upwards.

9.3.8 Draw horizontal dimension lines from the minimum x coordinate to the projection (or chain) line drawn in step 9.3.6. If the length of the dimension line is less than that of three arrow-heads then split it into two parts each of two arrow-heads length and draw each part pointing towards the other from the projection line. Give user the option to reject this dimension.

9.3.9 Let the user opt to move the projection line to the opposite side of the Graph.

9.3.10 Let the user move the dimension line further away from the view (in order to leave room for tolerancing).

9.3.11 Let the user opt to have the Node or Centre dimensioned from an adjacent datum dimension, rather than as a new datum dimension.

9.3.12 Display computed value of dimension and let user opt to modify this value.

9.3.13 Let user opt to have dimension toleranced.

9.3.14 Display cursor alongside the horizontal dimension lines or in the space between inward pointing arrows if this is sufficient. Inform user that this is where text will be written and let user adjust this position, if required.

9.3.15 Write the dimensioning text.

9.3.16 If there are any Nodes or Centre without a vertical projection line, go to step 9.3.8

9.3.17 If one or more upward projection lines have been drawn during step 9.3.8 draw an upward projection line through the datum point as far out from the boundary as the furthest upward projection line that was drawn during that stage. If one or more downward projection lines have been drawn during step 9.3.8 draw a downward projection line through the datum point as far out from the boundary as the furthest downward projection line that was drawn during that stage.

9.3.18 Arrange the y coordinates in order of magnitude, ignoring points on the circumference of complete circles and Dormant Nodes.

9.3.19 If more than one y coordinate has the same value, determine the minimum (XL) and maximum (XU) values of (x-XMEAN) for all such y coordinates. Set all such x coordinates that are less than XMEAN to XMEAN+XL and all that are greater than XMEAN to XMEAN+XU.

9.3.20 Starting at the first Node or Centre whose y coordinate value is not equal to YREF, draw a horizontal projection line from a Node or a horizontal chain line from a Centre that is not coincident with a previously processed one, i.e. ignore all repetitions of (x,y) coordinates. The distance that each projection (or chain) line extends beyond the boundary of the view is to be directly proportional to its position in the y coordinate ranking. A line which starts from a Node situated to the left of the centroid of the enclosing rectangle is to be projected to the left. A line which starts from a Node situated on the same vertical or to the right of the centroid is to be projected to the right.

9.3.21 Draw vertical dimension lines from the datum point to the projection (or chain) line drawn in step 9.3.20. If the length of the dimension line is less than that of three arrow-heads then split it into two parts each of two arrow-head length and draw each part pointing towards the other from the projection line. Give user the option to reject this dimension.

9.3.22 Let the user opt to move the projection line to the opposite side of the Graph.

9.3.23 Let the user move the dimension line further away from the view (in order to leave room for tolerancing).

9.3.24 Let the user opt to have the Node or Centre dimensioned from an adjacent positional dimension, rather than as a new positional dimension.

9.3.25 Display computed value of dimension and let user opt to modify this value.

9.3.26 Let user opt to have dimension toleranced.

9.3.27 Display cursor alongside the vertical dimension lines or in the space between inward pointing arrows if this is sufficient. Inform user that this is where text will be written and let user adjust this position, if required.

9.3.28 Write the dimensioning text.

9.3.29 If one or more rightwards projection lines have been drawn during stage 9.3.20 then draw a projection line to the right through the minimum y coordinate as far out from the boundary as the furthest rightwards projection line that was drawn during that stage. If one or more leftwards projection lines have been drawn during stage 9.3.20 then draw a projection line to the left through the minimum y coordinate as far out from the boundary as the furthest leftwards projection line that was drawn during that stage.

9.3.30 If there are no complete circles then go to Step 9.3.36

9.3.31 Starting at the first circle to be dimensioned compute its diameter. Draw dimension lines along a diameter at 45°. Let user opt to rotate dimension lines clockwise or anticlockwise.

9.3.32 Display computed value of diameter and let user opt to modify this value.

9.3.33 Let user opt to have diameter toleranced.

9.3.34 Display cursor alongside the diameter dimension lines. Inform user that this is where text will be written and let user adjust this position, if required.

9.3.35 Write the diameter text. If there are more complete circles to be dimensioned go to Step 9.3.31

9.3.36 If there are no circular arcs to be dimensioned then exit. Otherwise, starting at the first arc to be dimensioned compute its radius. Draw dimension line along a radius which bisects the angle subtended by the arc. Let user opt to rotate dimension line clockwise or anticlockwise.

9.3.37 Display computed value of radius and let user opt to modify this value.

9.3.38 Let user opt to have radius toleranced.

9.3.39 Display cursor alongside the radius dimension line. Inform user that this is where text will be written and let user adjust this position, if required.

9.3.40 Write the radius text. If there are more circular arcs to be dimensioned go to Step 9.3.36

The above algorithm was encoded in FORTRAN and implemented on the Superbrain QD. It was tested by the author and a wide variety of interested correspondents, eighteen of whom agreed to provide a subjective assessment. Reaction from these users is discussed in Section 9.4.

Typical output from this algorithm is shown in Fig. 9.4 (and also Fig. 9.5). The final result is reasonably acceptable but it is achieved by making extensive use of the various reject and modify routines that are provided. The draughtsman has to exercise significant

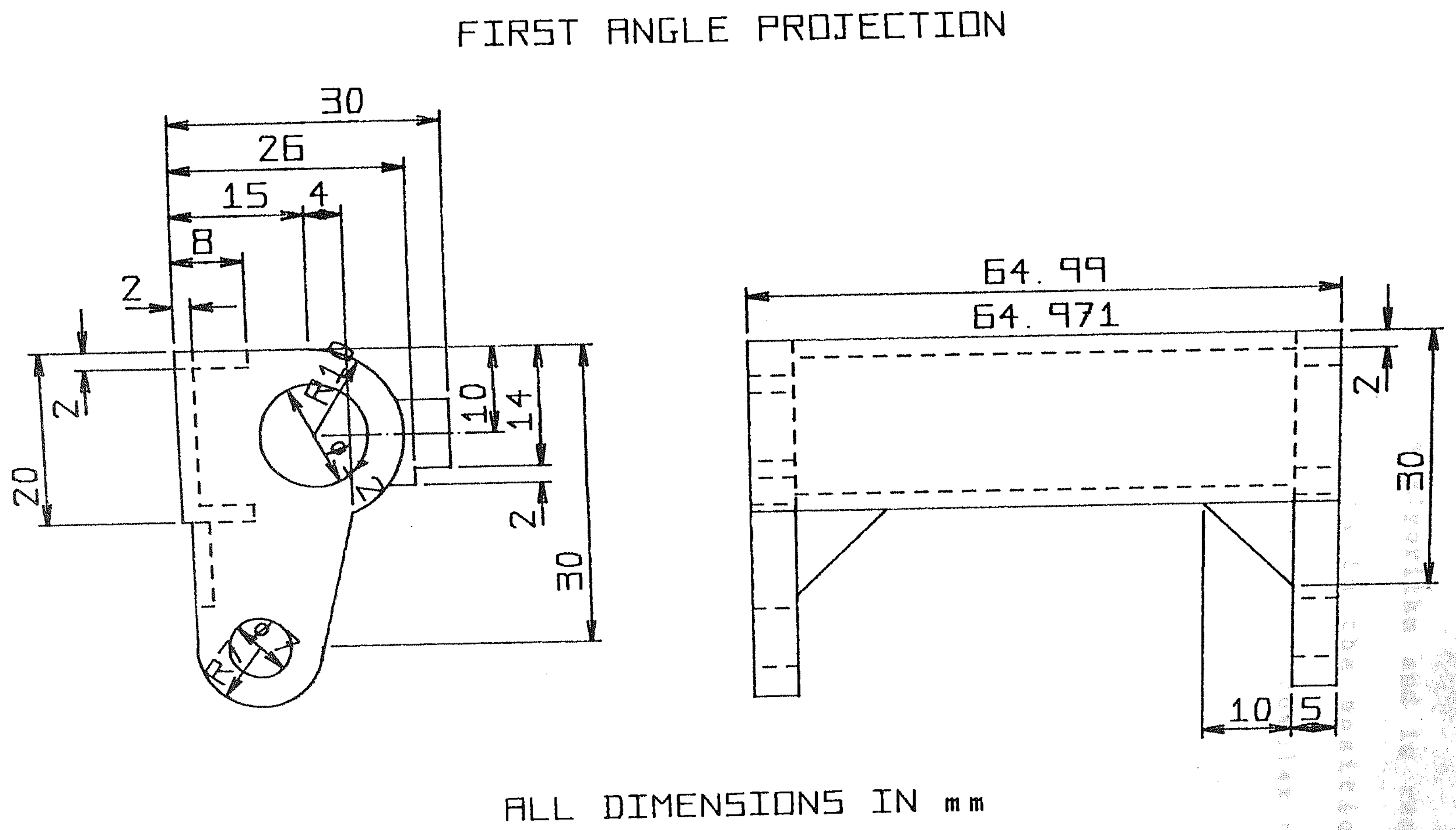


Fig. 9.4 Example of use of interactive dimensioning algorithm

judgement while using the algorithm and is required to make many fine adjustments in the positioning of dimension text. In short, despite the complex nature of the algorithm, many of the decisions that are built into it are overruled by the user who is better able to judge the clarity of the output.

9.4 Feedback from users of the interactive dimensioning algorithm

It was not possible to set up a controlled experiment. However, the objective was to gather qualitative information for planning the next step in the project by taking account of advice and suggestions from a group whose opinions were respected. The interactive dimensioning algorithm has been implemented on the SUPERBRAIN and several other microcomputers (e.g. IBM PC, Compustar). It has been used for the draughting of products as various as machine broaches and relief valves. Other potential users have provided valuable comments. Every volunteer user was asked to assess features of the software on a five-point scale. The responses are shown in Table 9.1

The concensus may be summarized as follows:

9.4.1 The provision of a single datum point for dimensioning seriously affects the flexibility of the dimensioning scheme.

9.4.2 The "dominant angle" approach, although

SUBJECTIVE SCALE:

0 =	Unhelpful. Impracticable. Should be abandoned.
1 =	May be useful to others, but not to me.
2 =	Occasionally useful but not of significant benefit.
3 =	Useful facility which should certainly be retained.
4 =	Particularly useful for my type of work.

18 Respondents	ASSESSMENT				
SOFTWARE FEATURE	0	1	2	3	4
	Number of times scored				
Single datum point	5	6	6	1	0
Alignment at dominant angle	3	6	7	1	1
Move Dimension to other side of view	1	3	5	7	2
Change Dimension from datum to size	4	4	7	3	0
Reject Dimension line	0	1	4	11	2
Look up tolerances	1	1	2	7	7
Vertical/Horizontal dimensioning	1	0	1	9	7
Dimension parallel to two specified Nodes	0	1	3	7	7
Angle dimension between two specified lines	0	1	4	8	5
Insertion of a diameter at a specified angle inside or outside circle	0	1	4	10	3
Insertion of a radius at a specified angle inside or outside arc	0	2	5	7	4
Overall assessment	3	4	7	3	1

Table 9.1 Responses from self-selecting sample of users of interactive dimensioning algorithm

representing a considerable advantage compared with horizontal and vertical dimensioning, is still restrictive. It should be possible to vary the alignment of dimensions.

9.4.3 The facility to move dimension lines to the other side of the view is particularly useful and frequently exercised.

9.4.4 The facility to change a datum dimension to a size dimension can be useful at times but the user should be able to specify directly which Nodes are to be sized.

9.4.5 The facility to reject dimension lines is frequently exercised because of deficiencies in the algorithm, such as 9.4.4. Another reason is the lack of an angular dimensioning capability.

Fig. 9.5 illustrates a single view which is particularly difficult to dimension because of deficiencies in the algorithm. The encircled features were added by selecting the normal arrow-head lines available within SUPERMODEL.

9.4.6 The automatic look-up and insertion of toleranced dimensions is particularly helpful in improving draughting productivity.

9.4.7 If the object represented by the drawing is basically one with a uniform cross-section or has axial symmetry then the computer-assisted dimensioning plays a useful role. When draughting more complex objects, however, the need to treat one view at a time is irksome and is unacceptable to many professional users.

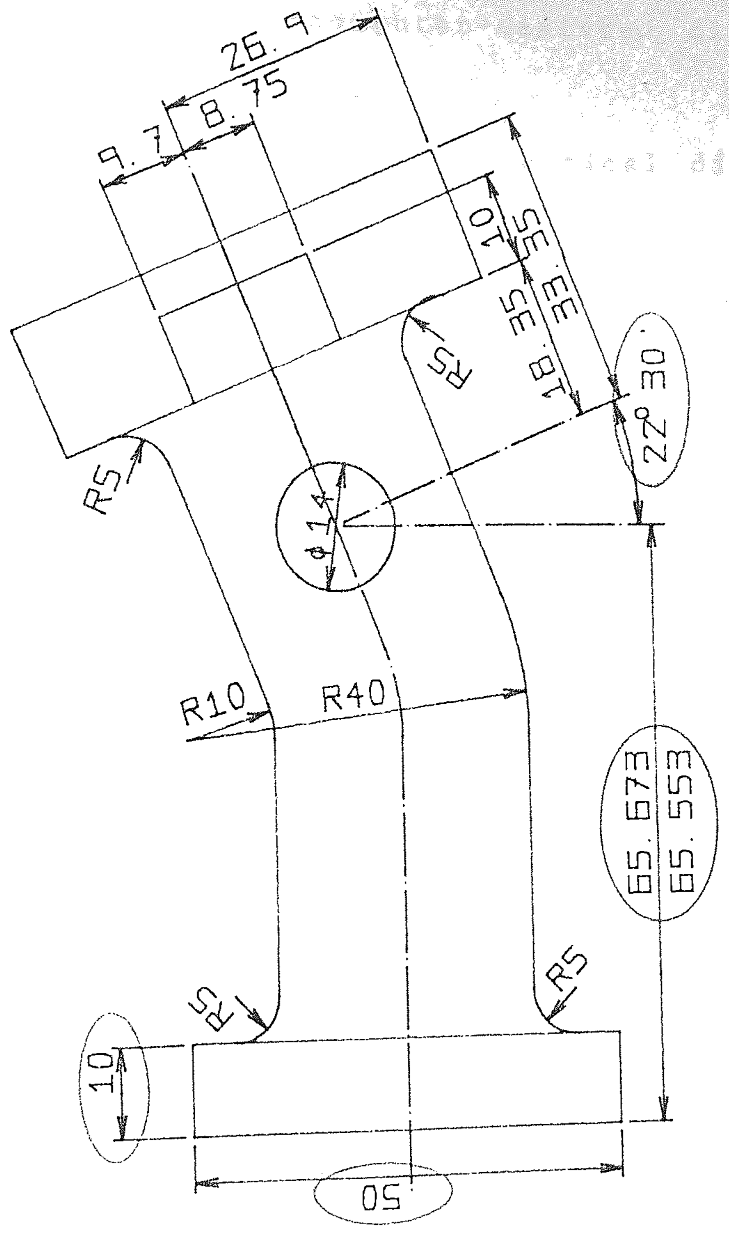


Fig. 9.5 View which could not be completely dimensioned by interactive algorithm

9.4.8 The most useful computer-assisted dimensiong routines are perceived to be:

a) Insertion of horizontal or vertical dimensions between two specified Nodes.

b) Insertion of a dimension parallel to two specified Nodes.

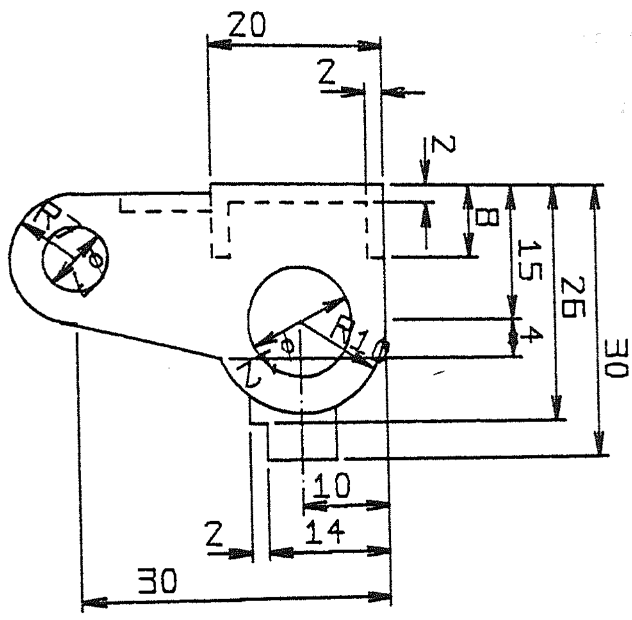
c) Insertion of an angle dimension between two specified lines.

d) Insertion of a diameter at a specified angle inside or outside the circle.

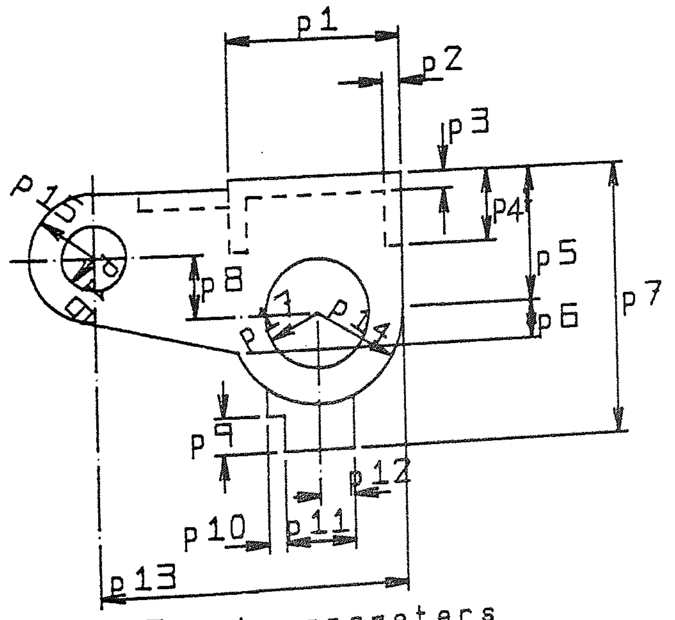
e) Insertion of a radius at a specified angle inside or outside the arc.

The first conclusion to be drawn from the above is that the approach whereby one view is dimensioned without reference to the others is not a fruitful line of investigation. With hindsight, this might appear rather obvious, but the implementation of the algorithm was important for several reasons.

Firstly, there is often a wide difference between the most sincere opinions of a potential software user before he has had "hands-on" experience and his reactions to that experience. This is the classical dilemma in many situations where a choice has to be made: only by making a committment is it possible to judge what one has chosen. Some users who were initially highly sceptical of the practicality of the dimensioning algorithm commented later that, with certain enhancements, it could provide a useful facility for



Dimensioning scheme



Input parameters

Fig. 9.6 Comparison of input to SUPERMODEL and dimensioning scheme adopted by same user

their particular application.
Secondly, the comments from users, although frequently destructive to the future development of the single-view approach, were constructive with regard to the development of the algorithm described in Chapter 10.
Finally, and perhaps most important, the user feedback frequently revealed the following trend. The dimensioning scheme that is produced for a flat surface reflects the draughting routines that were employed in order to define that surface. This phenomenon is illustrated in Fig.9.6, where the draughting operations are shown below the ultimate dimensioning scheme that was adopted.

The interactive dimensioning algorithm described in this Chapter is based upon the assumption that a view may be dimensioned by treating it as a Line/Arc graph whose linear dimensions are to be inserted. Assistance is provided by the computer by displaying projection and dimension lines in their proposed position and a draughtsman selects and manipulates these dimensions. Trials have indicated that the single-view approach to dimensioning should not be pursued. Thus, the final decision-making algorithm, which is the subject of the next Chapter, processes the entire geometric model.

CHAPTER 10

DIMENSIONING OF TOTAL GEOMETRIC MODEL

Feedback from users of the interactive dimensioning algorithm (described in Chapter 9) indicates that the most practicable method of dimensioning is to treat the model as a whole, rather than processing one view at a time. It is also clear that, whatever the imperfections of the hierarchy of surfaces proposed in Section 4.5, the general approach has some merit.

In general, a draughtsman is more concerned with establishing the geometry of a Hard interface than with any other class of surface. He also tends to deal with, say, Guides before he establishes the geometry of, say, Signals. In some cases where this trend was not apparent, it was found to be due to a wrong classification of the surface. For example, surfaces which are internal to a mould may be classed as Dams because they initially react pressure from a fluid. It would be more correct to class them as Hard interfaces because, once the moulding material has set, the geometry of the two contacting surfaces should be nominally identical.

The other important conclusion from the use of the interactive dimensioning algorithm is that there is significance in the way in which a surface is input to the geometric model. It would appear, although it cannot be proved, that an experienced draughtsman constructs surfaces in a manner which reflects his understanding of the function and method of production of the object. In effect, he lays down a set of parameters which are well-suited to the manufacture, inspection and purpose of the part.

In view of this, it is unnecessary to require a draughtsman to select the same parameters during a process of interactive dimensioning. If a decision-making algorithm were capable of positioning and spacing dimensions for optimum legibility then it would be possible to generate a working drawing automatically.

This chapter describes how the above conclusions were used in the design of an algorithm which processes the complete geometric model, and dimensions the various views that have been selected by the algorithms described above.

10.1 Foundations for an automatic dimensioning algorithm

The procedures described below are based upon two hypotheses:

10.1.1 The user of SUPERMODEL has input each surface in a manner which accords with his judgment of the optimum parameters for specifying the surface.

10.1.2 The user is aware of the implications of the ranking of surfaces, and has correctly identified the function of each.

The implications of the ranking of surfaces are that the top-ranked surfaces (i.e. Hard interfaces) will be the first group to be (explicitly) dimensioned. When all Hard interfaces have been dimensioned, the Soft interfaces will be processed, etc. Once an Edge has been positioned, it is unnecessary to position the same Edge when it is re-examined as part of an equal, or lower-ranked, surface.

The parameterization of surfaces may best be examined by means of a typical surface, Fig. 10.1, and the manner in which it was input to SUPERMODEL, Table 10.1. Starting from the origin of the system of (x,y) coordinates, the user has first specified the centre and radius of circle A1. The centre of A2 is set out as a radius and angle from the centre of A1 and the radius of A2 is specified. The user then specifies external common tangents to the two circles and removes a part of each circle to create the pear-shaped circuit. Finally, a circular hole is drawn concentric with A1 and it is modified with flat sides a specified distance apart.

Fig. 10.2 shows the parameterization that the draughtsman has effectively imposed on the surface. There are two features of particular importance. Firstly, the absence of dimensions to the four Nodes around the pear-shape implies tangency. Each of these Nodes has $C^{(1)}$ continuity and, as with the interactive dimensioning algorithm, it does not require dimensioning. Secondly, the dimensions to the hole render the symmetry explicit.

There are several other SUPERMODEL facilities available

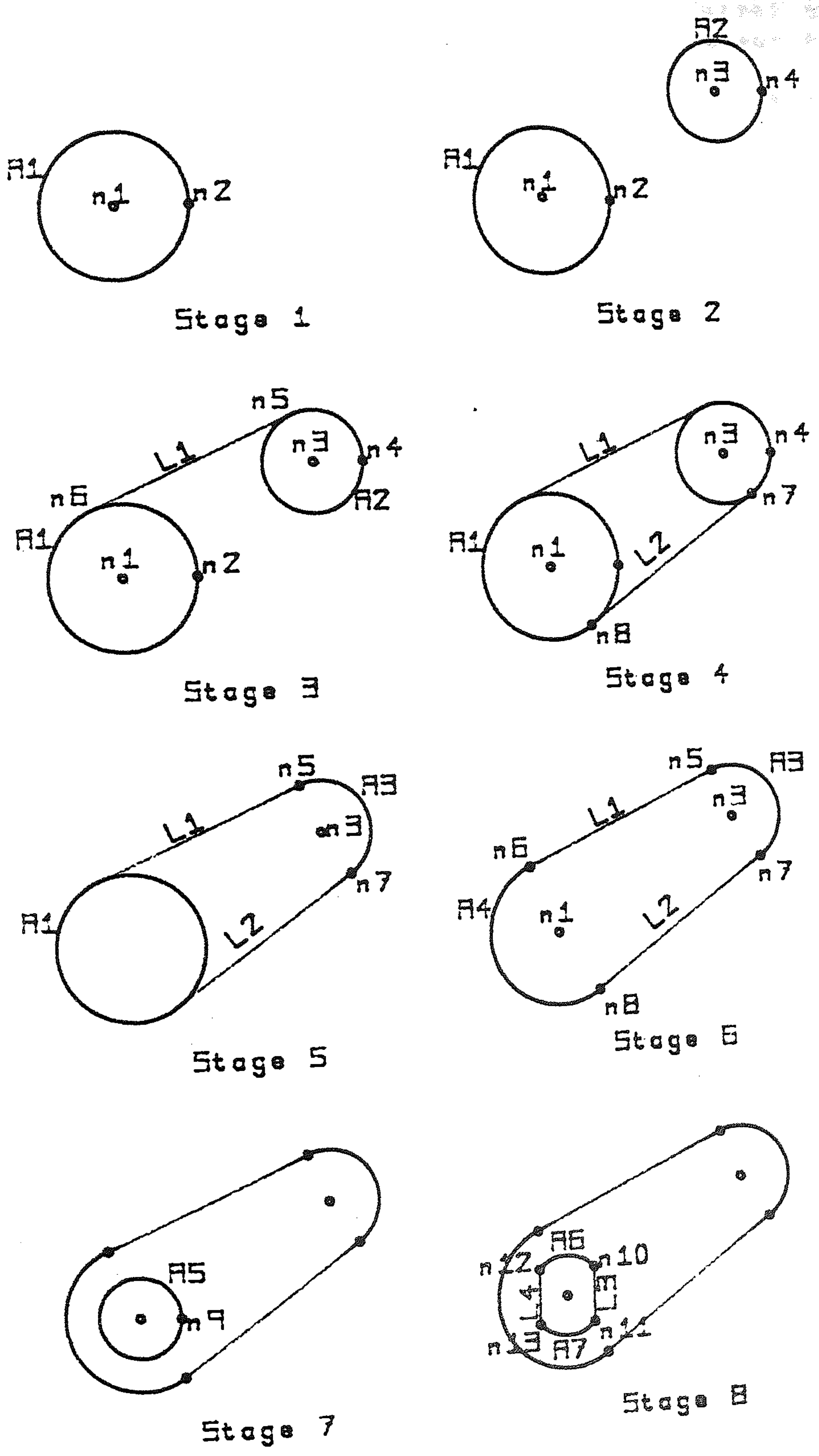


Fig. 10.1 Stages in the input of a flat surface to SUPERMODEL

Cursor x y	Menu Item	Description	Result
256 128 0 0	40 41	Cursor to datum point Input rect. coords	Cursor to (0,0) Cursor to (20,30)
20 30	44	Circle radius 10 with cursor as centre	Circle A1 Centre n1 (20,30) n2 at (30,30) Start, Finish n2
20 30	41	Input polar coords (30,30)	Cursor to (46,45)
46 45	44	Circle radius 7 with cursor as centre	Circle A2 Centre n3 (46,45) n4 at (53,45) Start, Finish n4
53 45	54	First common tangent	Line L1 Start n5 Finish n6
At n6	54	Second common tangent	Line L2 Start n7 Finish n8
At n8	35	Delete last arc	A2 removed
At n8 At n6	1 2	Specify start Specify finish	Start at n7 Finish at n5
At n6	7	Arc	Arc A3 Centre n3 Start n7 Finish n5
At n5 At n7 At n1	1 2 3	Specify start Specify finish Specify centre	Start at n5 Finish at n7 Centre at n1
At n6	32	Find arc	A1 is last arc
At n6	35	Delete last arc	A1 removed
At n6	7	Arc	Arc A4 Centre n1 Start n6 Finish n8
At n8	44	Circle radius 7	Arc A5 Centre n1 Start, Finish n9
At n1	41	Input rect. coords	

......etc. Construct lines L3 and L4......

Table 10.1 Sample of input to SUPERMODEL in order to generate the shape in Fig. 10.1

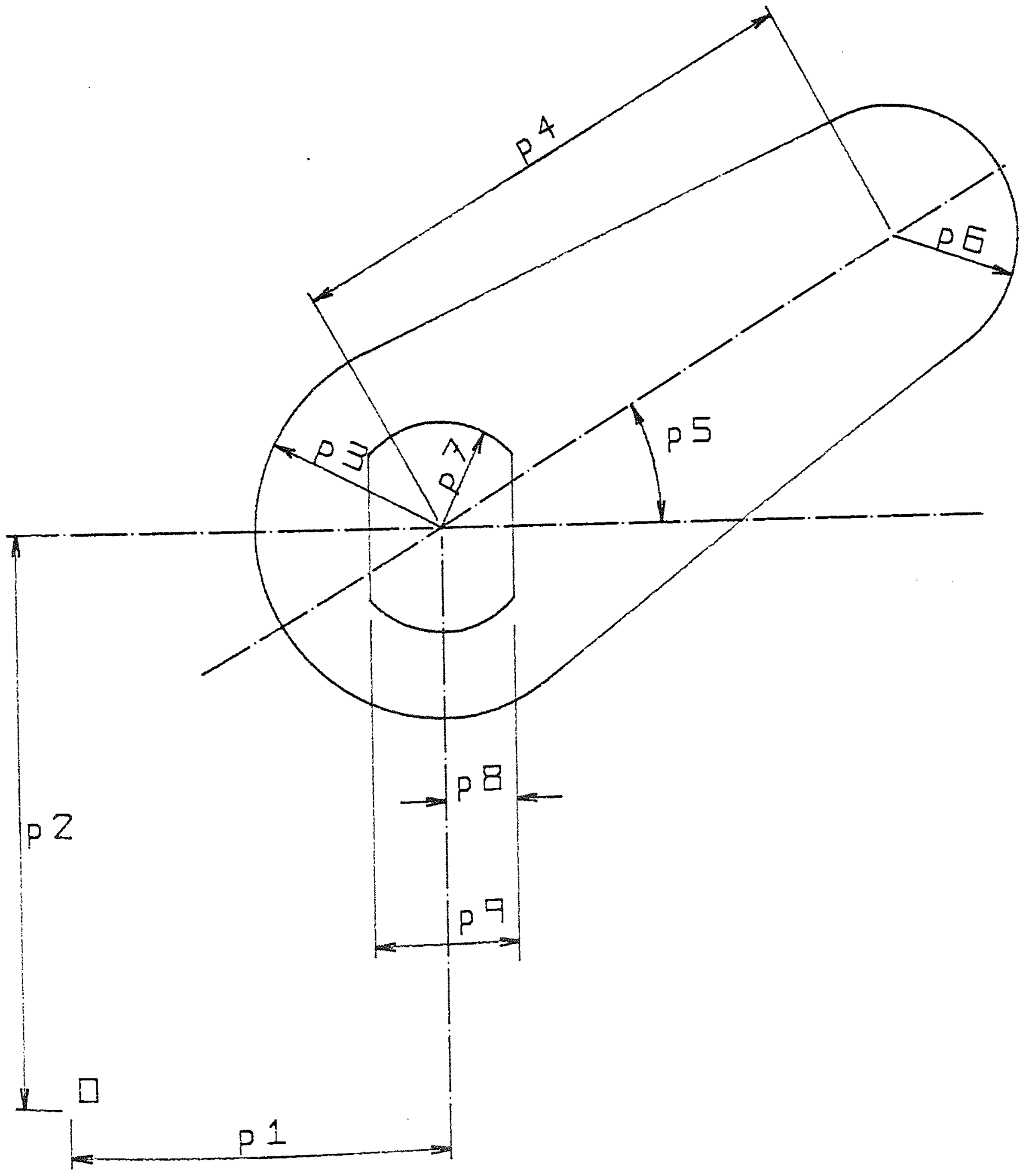


Fig. 10.2 Input parameters for surface

which have implications for dimensioning. For example, the rotation transformation implies a repeated feature on a pitch circle; the translation transformation implies a repeated feature on a pitch line, etc. User selection of the parallel line or perpendicular line routines implies, respectively, that the parallelism or squareness of the two lines is a significant feature of the geometry. The scaling transformation implies concentricity and the rectangle routine (Menu item 45) implies squareness.

In order to utilize the implied characteristics of the surface geometry, it is necessary to devise a method of recording the user input in some efficient format. Each menu item number describes a set of operations to be carried out on certain geometric variables which form the Graph that represents the surface. Therefore, it is convenient to use these numbers as a code for the routine that was selected. These routines process:

a) Nodes/Centres, of which each surface has a non-empty set N(G),

b) Edges (Lines/Arcs), each of which may be defined as a pair of Nodes and,

c) Sub-surfaces, each of which is a sub-set of Edge-families, E(G).

As each Node/Centre and each Edge has a unique address in the SUPERMODEL data structure, such addresses may be employed to generate an efficient code for the draughting operation. It is merely necessary to store the address of the Node/Centre where the cursor is located, or the address of the line/arc on which the cursor lies, or the movement of the cursor from its last position.

Additional integer codes are required for options that are available within some routines. The common tangent routine, for example, has up to four options. The output from each draughting operation is a new cursor position, a new line, arc or sub-surface, or the deletion of a line, arc, or sub-surface. The canonical form for each draughting operation is therefore:

1. Menu Item No.
2. Option No.
3. Address of Cursor Position
4. First real number input
5. Second real number input

A cursor address of zero signifies a new cursor position which was established by the draughtsman by inputting the appropriate cursor movements or coordinates. A positive cursor address signifies a Node/Centre address. A negative cursor address signifies a line/arc address. If the Option No. is zero then there are no options for the particular routine.

Using the above canonical form, the complete set of draughting routines for the shape shown in Fig. 10.1 may be stored as shown in Table 10.2. The advantage of this

Menu Item No.	Option No.	Cursor Code	Input Parameters	
40	0	0	0.	0.
41	0	0	20.	30.
3	0	0	0.	0.
44	1	0	10.	0.
42	0	2	30.	30.
3	0	0	0.	0.
44	1	0	15.	0.
54	1	3	0.	0.
4	0	0	0.	0.
54	2	6	0.	0.
4	0	6	0.	0.
35	0	8	0.	0.
1	0	8	0.	0.
2	0	6	0.	0.
7	0	6	0.	0.
1	0	5	0.	0.
2	0	7	0.	0.
3	0	1	0.	0.
35	0	-1	0.	0.
7	0	0	0.	0.
44	0	0	7.	0.
41	0	1	5.	10.
1	0	0	0.	0.
41	0	0	0.	-20.
2	0	0	0.	0.
4	0	11	0.	0.
50	1	0	0.	0.
1	0	0	0.	0.
50	2	0	0.	0.
2	0	0	0.	0.
34	0	0	0.	0.
4	0	13	0.	0.
55	0	15	8.	0.
1	0	15	0.	0.
2	0	13	0.	0.
35	0	13	0.	0.
7	0	13	0.	0.
1	0	12	0.	0.
2	0	14	0.	0.
7	0	14	0.	0.

Table 10.2 Canonical form of SUPERMODEL draughting input

data structure, apart from compactness, is that the shape has been parameterized in a form which is particularly suitable for both dimensioning and tolerancing. The effect of variations in the dimensions may, if required, be displayed or analysed by varying the input data and computing the influence that this has on the geometry of the surface.

10.2 Space Planning of Orthographic Views

As each surface is processed, it is first necessary to identify the view on which the surface appears in its true shape. In manual draughting, it is not uncommon for the views to be laid out and then dimensioned. This may be done with reasonable success because a draughtsman is able to make quite an accurate assessment of the space that will be required for dimensioning. The situation in which the human draughtsman operates is one of decision making under risk. By experience, he makes an assessment of the probability that a certain amount of space will be required for dimension and projection lines.

Such a tactic is not considered appropriate for automatic dimensioning. After studying various ways of estimating the probable space required for projection and dimensions lines, the author concluded that the risk could be eliminated by a correct ordering of the tasks that have to be undertaken. Hence, a view generated by the SUPERMODEL hidden-surface algorithm is not positioned within the total drawing until it has been dimensioned. Decisions about the placing of dimension lines and text are made first and then a suitable layout of the views may be accurately computed.

Unlike the interactive dimensioning algorithm, it is not possible to arrange for the longer dimensions to be further away from the outline of the view because the dimensions are not processed in order of magnitude. It is necessary to carry out a sorting process each time that a new dimension is to be placed adjacent to a view. This leads to a complex set of geometric decision-making routines; this is complicated by the need to avoid crowding and overlapping of both the dimensions and the text.

Fig. 10.3 illustrates how a poor decision taken at an early stage has a profound effect on the clarity of later dimensions. The dimension "110" is too close to the cross-hatched shape. Not only does it intersect the projection lines from the dimensions above it but it also intersects the actual dimension line for the figure "35.355".

Projection lines are used in order to take the dimension line away from the feature to which it relates. The projection line may remain within the boundaries of the orthographic view (internal) or extend outside these boundaries (external).

Let A be the angle at which a projection line heads away from the feature to which it relates. A projection line is designated positive if $0^\circ \leqq A < 180^\circ$.

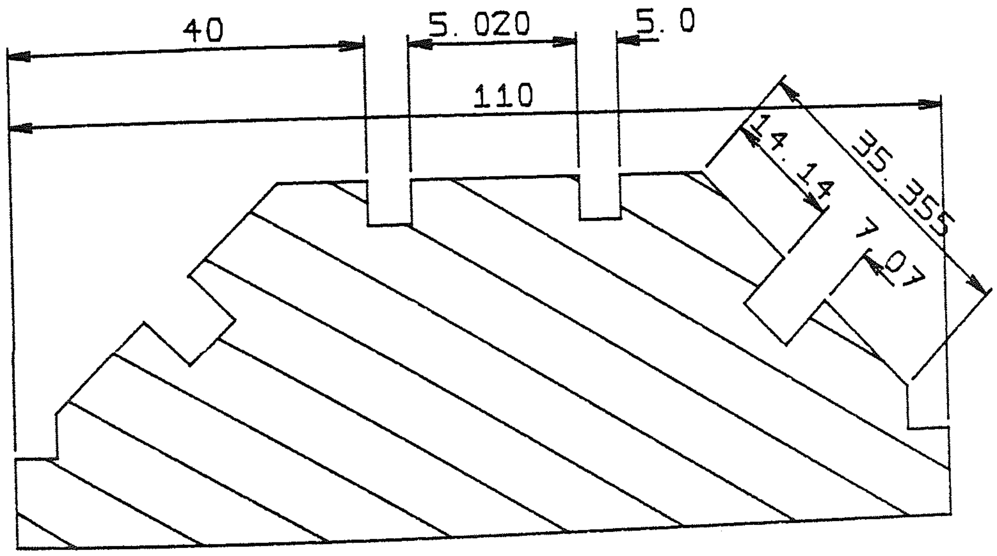


Fig. 10.3 Consequences of a poor dimensioning decision taken at an early stage

Fig. 10.4 shows the strategies that are available for drawing projection lines for linear dimensions :

S_1 No projection lines (internal)

S_2 One internal positive projection line

S_3 One internal negative projection line

S_4 Two internal positive projection lines

S_5 Two internal negative projection lines

S_6 Two external positive projection lines

S_7 Two external negative projection lines

S_8 Two internal projection lines with opposite signs

Only strategies S_6 and S_7 are provided by the implementation of the interative dimensioning algorithm described in Chapter 9. Reaction from users indicates strongly that all eight strategies should be available to a draughtsman faced with a dimensioning decision of this nature.
The disadvantages of accepting all eight strategies for a decision-making algorithm are twofold. Firstly, in order to made a decision in favour of an "internal" strategy, some evidence is required that the space is being put to good use. There may be other dimensions with a better claim to the space. Secondly, projection lines are frequently shared by two or more dimension

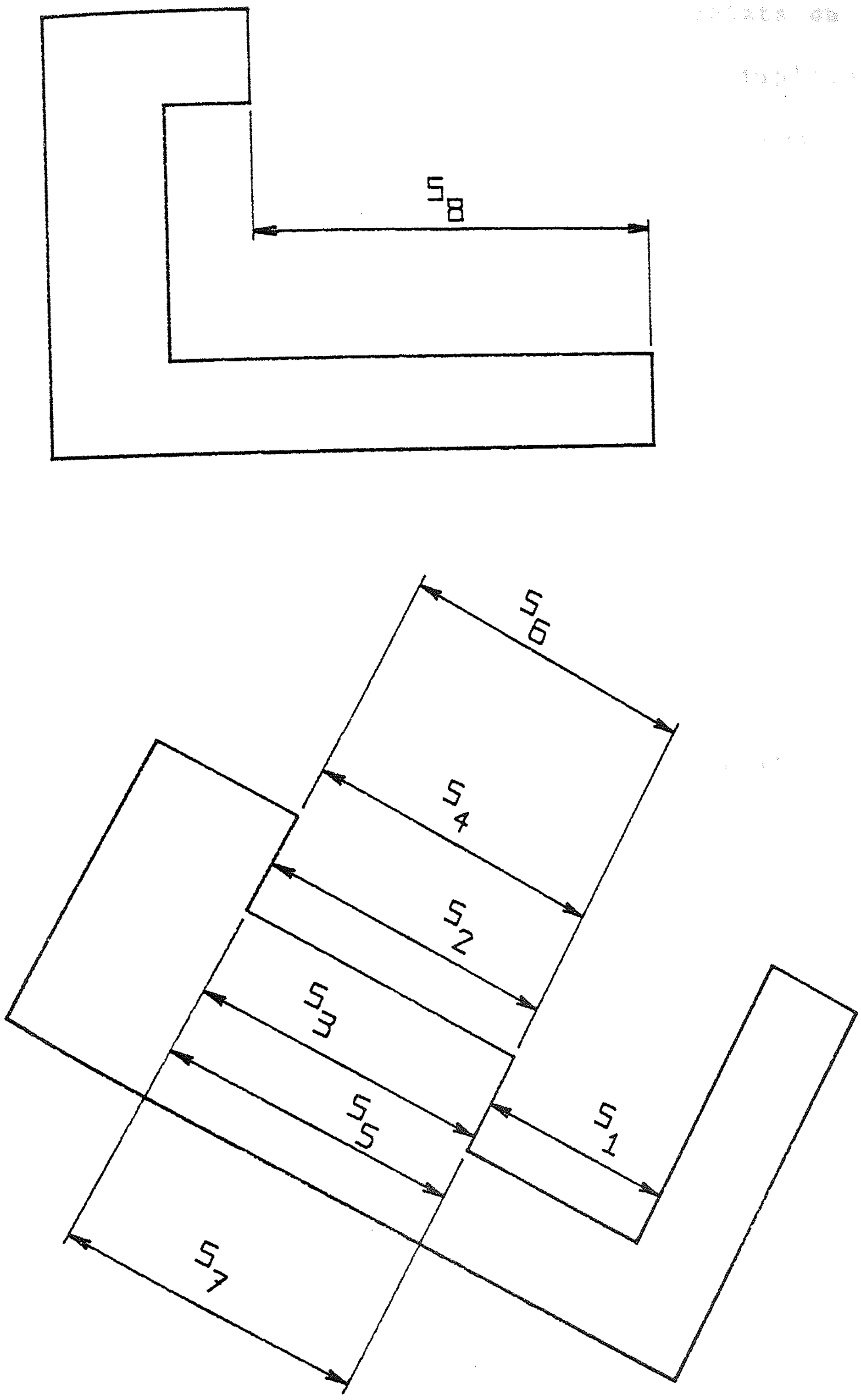


Fig. 10.4 Strategies for projection lines

lines (See Fig. 9.4). The sharing takes the form of two opposite pointing arrows meeting at the same point or two arrrows in the same sense at two points on the same projection line. In order to avoid duplication of projection lines, it is necessary to search through those in place to determine if a suitable one already exists, or should be extended. The cost of this checking is compounded by the availability of many projection line strategies.

Furthermore, there are three other drawing features to be considered. Fig. 4.4 shows two techniques for dimensioning angles: one using projection lines. Fig. 4.10 illustrates four distinct strategies for circles and Fig. 4.10 has five strategies for arcs.

It is not, however, considered necessary in earlier Chapters to reject a strategy on the grounds of suspected high computational cost. The uncertainty here arises from the lack of a description of the mechanism of dimensioning, whereas the mechanisms of, say, sectioning and hidden edge determination are well understood. The following model is proposed as a means whereby the complex relationships between the orthographic view and its linear dimensions may be investigated. If the behaviour of the model is in reasonable agreement with draughting practice then it may be used as the basis for a decision making algorithm.

10.3 Dynamic model for dimensioning

The initial concept is that of a projection line "attractor". The positions of the Nodes and Centres of the Graph which is an orthographic projection of the solid model may be specified only if a projection line is constructed through such a point. Therefore, such points are said to be "attractors" of projection lines. A "dormant" Node (defined in Section 9.2) does not attract projection lines. A Centre attracts projection lines if its position cannot be inferred from the radius and/or end Nodes of an arc around it. Fig. 4.9 indicates that two diametrically opposite points on the circumference of a circle may also be projection line "attractors".

Attractors generate projection lines from two opposite sources, each at an infinite distance from the orthographic view. A projection line starts at a source, heads in a direction that is normal to the dimension required, and stops either at the attractor or when superposition on another line is about to occur. The conventional gap, referred to in Fig. 4.1, is not relevant to the model under consideration. A projection line does not start if it would superpose immediately on another projection line.

According to the above model, the attractors contained within the orthographic view shown in Fig. 10.5 would generate the projection lines that are indicated.

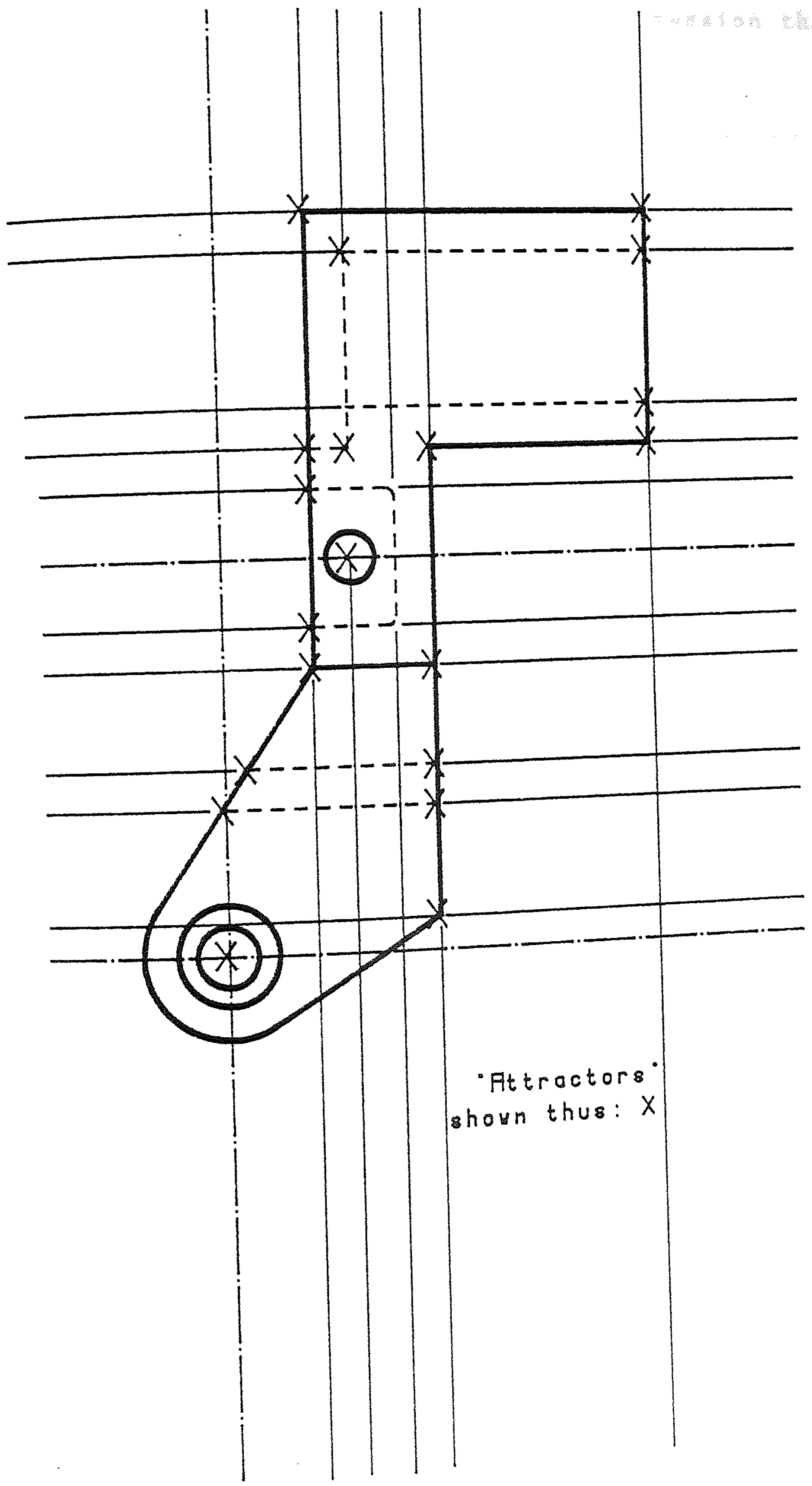


Fig. 10.5 'Attractors' and projection lines

The next concept built into the model is a "slider" with the following properties:

10.3.1 A length equal to the linear dimension that it is required to indicate.

10.3.2 An ability to slide along parallel projection lines.

10.3.3 A tendency to minimize the sum of the two distances to its attractors.

10.3.4 A mandate to stop at a fixed distance from the blocking feature if it cannot move closer to its attractors.

10.3.5 When moving towards its attractors, the ability to change places with or pass through a parallel blocking slider whose projection lines are not intersected.

10.3.6 Absolute rejection of a position that superposes a centre line or projection line.

10.3.7 An aversion to intersecting other sliders, projection lines and Edges.

The above properties are proposed in order to model the dynamics of adding new linear dimensions to an orthographic view. Fig. 10.6 shows how sliders might take up a new equilibrium position after the introduction of a new slider. At Stage 1, the slider labelled "220" is the first to be introduced. It settles on the right because that is the nearest it can get to its upper attractor. If it were on the left, the sum of

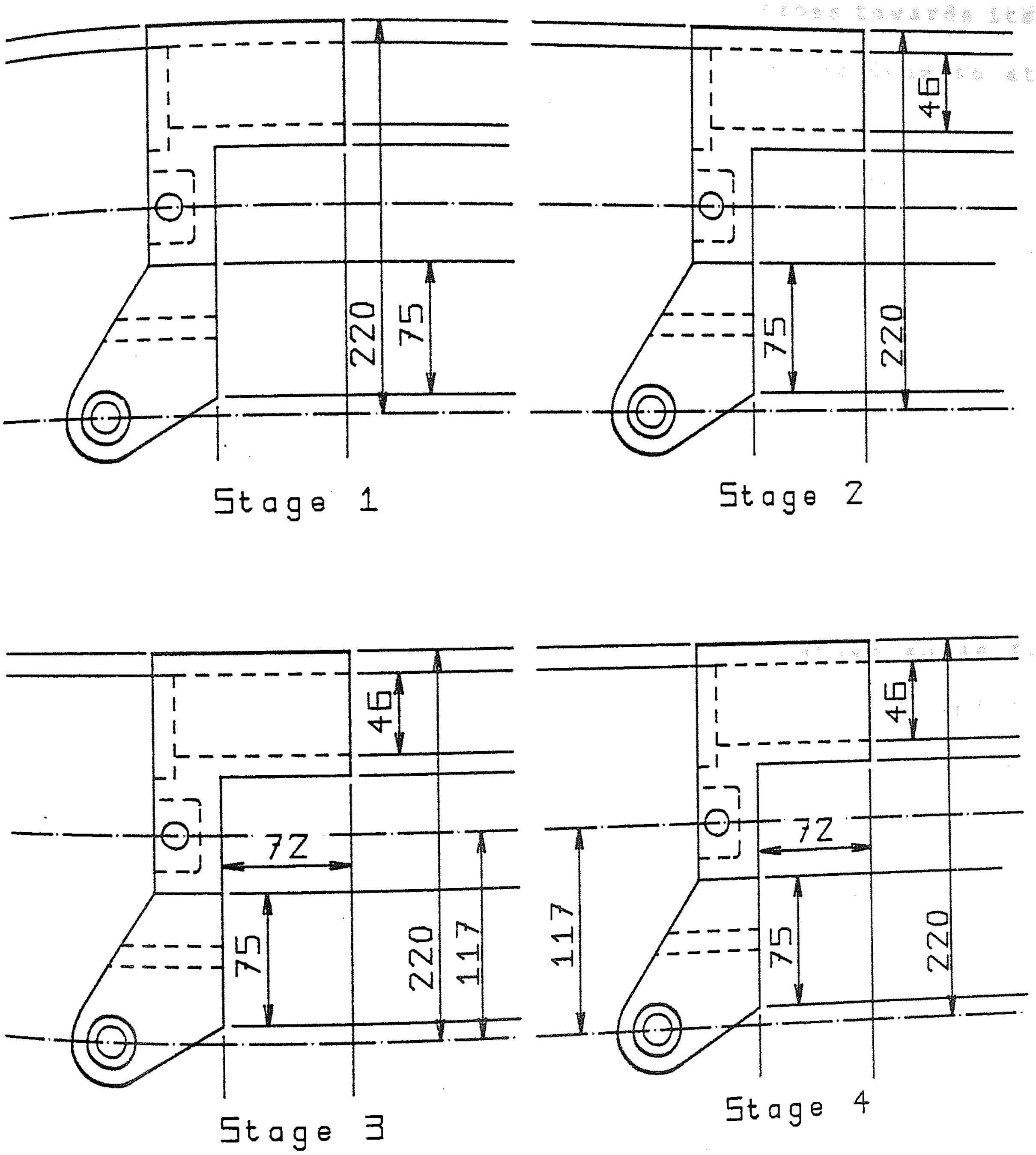


Fig. 10.6 Equilibrium of 'sliders' in dynamic model for dimensioning

its distances from its two attractors would be greater than on the right. The slider labelled "75" is then introduced. As it does not intersect the projection lines of the other slider it can move across towards its own attractors and it is shown as having done so at Stage 2.

The slider labelled "46" is then introduced. It does not intersect the projection lines of the slider labelled "220" and so it may move towards its own attractors by an interchange of positions. Had this slider been introduced from the left, it could not have moved so close to its attractors. The equilibrium positions of these two sliders is shown at Stage 3.

Also at Stage 3, the horizontal slider labelled "72" is introduced. It takes up the position shown so as to avoid intersecting the slider labelled "75" and to minimize the sum of the distances from its attractors.

The vertical slider, labelled "117", is now introduced in the position shown. Its equilibrium state is the most complex yet examined.

Stage 4 shows the equilibrium position of the slider labelled "117". It may pass through slider "220" and could remain colinear with slider "46" but the sum of distances to its attractors would not be minimized. As it moves towards its attractors it crosses the slider labelled "72" and then other Edges. At the position shown, it intersects no Edges and has minimized the sum of distances to its attractors.

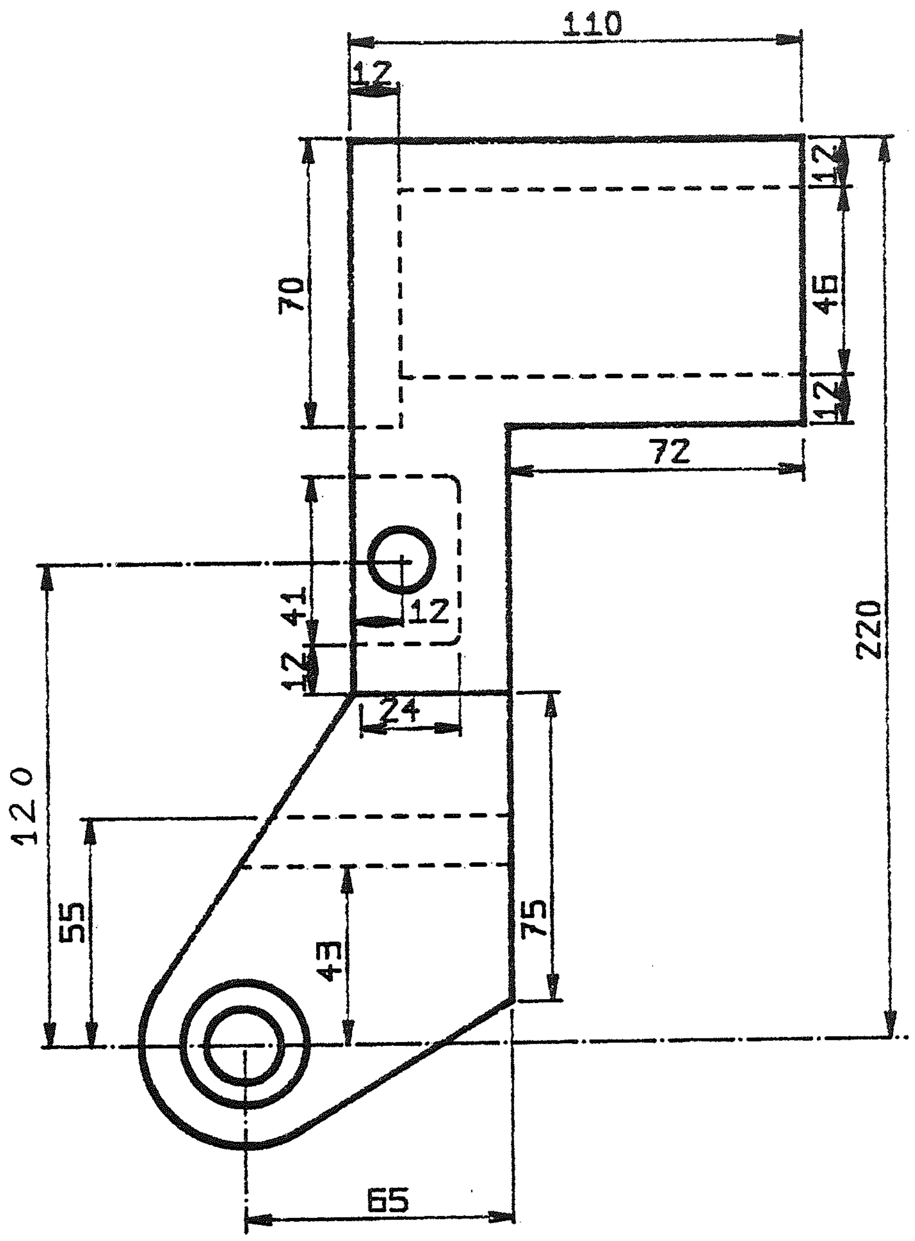


Fig. 10.7 Final scheme produced by dynamic model for dimensioning

The final dimensioning scheme that would be output by the proposed dynamic model is illustrated in Fig. 10.7. Tests on other orthographic views were undertaken and found to be reasonably encouraging. Properties 10.3.3 and 10.3.7 above need to be quantified in order to use the model in a decision-making algorithm. It is the relative strengths of the tendency to minimize distances, on the one hand, and the aversion to intersecting lines, on the other which has the most important influence on the output of the model.

Property 10.3.7 may be subdivided in order to take account of different types of lines. It may, for example, be considered more important that a slider avoids intersecting cross-hatching lines than any other type of line and a suitable weighting may be given to this aversion.

An algorithm based upon the above dynamic model cannot process angular dimensions. Any angles that were input by the user in the course of constructing a surface are stored in the canonical form that is shown in Table 10.2. and require a separate, but similar set of rules. The same is true for special features such as radii, diameters, regular polygons, etc. The algorithm described below is essentially a prototype for testing the basic dynamic model for linear dimensions.

10.4 Algorithm J

The basic procedure is to process the geometric model by working through its surfaces in order of their classification, i.e. all Class 1 surfaces (hard interfaces) are dealt with first, followed by Class 2 (soft interfaces), etc. A file is maintained for each of the orthographic views that were selected by Algorithm A. As each surface is processed the data for the relevant view is modified and extended. When all surfaces have been processed, it is possible to determine a suitable spacing for the views and their surrounding dimensions.

SPECIFICATION OF TOTAL MODEL DIMENSIONING ALGORITHM

10.4.1 Select first class of surface: Hard interface.

10.4.2 Select next example of current class of surface. Read file containing canonical form of draughting input for this surface.

10.4.3 Determine view on which surface appears and read data file for this view.

10.4.4 Select next parameter (linear dimension) of surface.

10.4.5 Determine the equilibrium position of the conceptual "slider" which is to obey these rules:

> 10.4.5.1 Start on side of view with fewest parallel "sliders" and move to fixed distance from another "slider" or Edges of view.

10.4.5.2 If there is a parallel blocking "slider" whose projection lines are intersected then go to 10.4.6

10.4.5.3 If there is a parallel blocking "slider" whose projection lines are not intersected then interchange with it. Go to 10.4.5.2

10.4.5.4 If there are no parallel blocking "sliders" then move to the position of minimum sum of distances from "attractors". Compute number of other "sliders", projection lines and Edges that are intersected. If number of intersections is greater than four then go to 10.4.6

10.4.5.5 Move "slider" to position that has been checked out. Go to 10.4.5.2

10.4.6 Update dimensioning scheme of view.

10.4.7 If there are more parameters to be processed then go to 10.4.4

10.4.8 Write updated view data to diskette. If there are more examples of the current class of surface to be processed then go to 10.4.2

10.4.9 Increment class of surface and go to 10.4.2 unless there are no more classes to be processed.

10.4.10 Determine bounding rectangle for each view with its dimensions.

10.4.11 Compute positions of bounding rectangles to give suitable clearances.

The final decision-making algorithm has been described. In Chapter 11 there is a discussion of some general principles in the design of such algorithms which may be observed in the ten algorithms detailed above. The areas where work remains to be done are indicated and some potential benefits are listed.

CHAPTER 11

CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

11.1 The nature of decision-making in Geometric Modelling

In order to view this project in context, it may be useful to consider first if the decision-making algorithms constitute what has come to be termed an "expert system". In lay terms, it may appear to be so because the routines are designed to emulate the work of an experienced draughtsman. There are still several interpretations of the phrase "expert system", despite efforts by the British Computer Society to lay down a standard definition. It is frequently difficult to distinguish between an "expert system" and a large database which can be accessed by advanced enquiry and reporting routines.

Legard [53] quotes the opinion of the technical director of the Framentec company: "An expert system is a database, with a set of rules held separately to make the search of the database more efficient". There appears to be a trend to refer to such a database as a "knowledge base" and to the set of rules which organizes this knowledge as an "inference engine".

From these definitions it is clear that the above algorithms should not be classified as an "expert system" because, although there is a small number of data to be referenced by many of the procedures, there is no need for an especially efficient search through these data. The element of expertise that is built into the decision-making algorithms is manifested in the rules for testing alternative strategies and establishing priorities.

It is stated in Chapter 1 that the essential difference between the type of application reported in this thesis and many of the other applications of Geometric Modelling is the large number of valid solutions which may be generated. In certain circumstances there is a small group of solutions which are arguably better than most of the others, in the sense that they are less confusing and more readily interpreted. In the course of the project it became clear that the principal difficulty when designing a decision-making algorithm is to devise a test which detects a marginal difference between the better solutions. Essentially, it is necessary to examine precisely the characteristics of a "good" decision but, in practice, it is more feasible to quantify certain properties of a "bad" solution.

For each of the algorithms described above, the first stage in their design is (as suggested in Section 5.1) to decide on the general aim that is to be achieved and then to specify it in detail. However, it was not always possible to use this specification directly. One might suppose that any intermediate decision that advances the purpose of the activity could be considered to be a "good" decision but there are, of course, many situations where a decision can only be judged with hindsight. This is especially true of game-playing algorithms, which have many similarities with the type of algorithms described above. There are other objectives which may only be attained by first making some neutral decision which does not appear to advance the cause, and yet other situations where it is actually necessary to give ground before the ultimate goal may be achieved.

The fundamental difficulty in developing software which is to emulate human judgment is that it is virtually impossible to quantify all the characteristics of the ultimate objective. For most of the objects that may be defined by using SUPERMODEL there can be thousands of significantly different dimensioned working drawings. All of these drawings must specify the geometry completely and unambiguously but some may be highly impracticable when the manufacturing process is considered , others would be acceptable, while some might stand out as manifestly "good" solutions. One technique which helps to diminish the effects of such difficulties is referred to in Chapter 6: a search for quantifiable undesirable characteristics. It appears that such negative evidence is a vital part of man´s subjective assessment of probabilities, a point which is taken up again below when some results from experimental psychology are examined.

If the results of an intermediate decision will not be reversed or nullified by later decisions and it proves impracticable to quantify the desirable qualities of the alternative outcomes then it would appear that a useful way forward is to minimise the undesirable characteristics. In other words, the concept of "minimax regret" pervades much of the work reported here, even when the mathematical formulation of this particular decision-making technique is not employed.

One unusual feature of this project is that the decision-making algorithms are ultimately required to generate an end-product which has, hitherto, been heavily reliant upon human experience and judgment. There are parallels with game-playing computer software but they need to be regarded with caution before attempting any general conclusions. The principal difficulty is that, in game-playing, there is one simply stated and clearly manifested objective: to win. The attainment of a win is defined by the rules of the game and most game-playing algorithms are designed to detect this situation and either claim victory or concede.

For this project (and any similar work) the final arbiter of the performance of the algorithms must be the human user of the working engineering drawing. The outputs of the various algorithms produce a complex end-result which cannot be easily assessed by computer. It

is possible to improve an algorithm by analysing the comments of users of the output, but eventually there is no concensus of criticism and further improvement becomes a question of individual preference. In game-playing algorithms there is always the ultimate test: can it beat the strongest available opposition?
The author has sought an analogous problem in science or technology so as to broaden the discussion of this project. The fundamental nature of the problem is to transform a compact numerical description into a standardised format that is suitable for human reading. There are one or two indirect references to associated problems in the literature of "expert systems" but nothing which merits detailed consideration. The closest parallel may be the automatic translation of human languages which, as Leech [54] suggests, is still at the stage of attempting to understand the structure of the input data and the output format. Clearly, there is still much to be learned about computer applications which try to emulate the human skills involved in communication.

11.2 Types of decision-making algorithms

The decision-making algorithms that are described above fall into two broad classes, according to the technique that is used for making the decision. These technques are well-established in the field of artificial intelligence. The earlier algorithms, such as those for selecting exterior views, employ a "Generate-and-Test" paradigm. Algorithms C, for testing auxiliary views, and J, for dimensioning the total geometric model, are examples of a purely "Rule-Based" paradigm. Some of the other algorithms (for example Algorithm F, for clarifying orthographic views) are hybrid.

This immediately raises the question of the circumstances which give rise to the adoption of a particular paradigm. They are well illustrated by the case of Algorithm J, in which the total geometric model is dimensioned by employing a complex set of rules. Given the views which have been selected for dimensioning, there is no apparent limit to the number of different ways in which the dimensioning may be done. If the continuously variable positions of the dimensions is ignored and one dimensioning scheme is considered to be equivalent to another with the same topological arrangement of dimensions, then there is still a very large number of possible schemes. It may often be of the order of 10^4 for a model with twenty surfaces, many of which are "hard interfaces".

Clearly, the computational cost of generating and testing such a large number of schemes would be prohibitive. If the number of acceptable dimensioning schemes were limited by applying some arbitary rules there is still the problem of how to test the acceptable schemes. Algorithm B is based upon the assumption that the better viewing direction is the one which reveals the greater number of surfaces, but there is no such simple test for a dimensioning scheme. It would appear that the "Generate-and-Test" paradigm is not suitable for situations where large numbers may be generated, or it proves impossible to formulate an acceptable test which yields satisfactory results, or both.

The main disadvantage of a "Rule-based" paradigm is exactly that which applies to a set of rules which are supposed to guide man-made decisions. Legislation is a typical example: sooner or later an Act of Parliament has to be interpeted by the Courts, because the Legislature can never hope to foresee all the special cases that will arise after enactment. An implementation of Algorithm C (for testing if an auxiliary view is necessary) would be likely to encounter some special cases and a rigid application of its rules would create anomalies. One feasible way out of this difficulty would be to encourage accurate reporting by users, so that these anomalies are quickly recognized and, if possible, eliminated.

The design of a "Rule-based" paradigm is especially difficult when it is required to make a "go or no go" decision, and yet it cannot be proved that the quantifiable characteristic on which the decision is based is vital. Experience with prototype implementations of some of the algorithms described above suggests that, if it is possible for a decision to be made simply on the basis on one computed inequality (which cannot be shown to be essential), then some of the decisions will be questionable.

In human terms, examples of this may be found in the administration of examination regulations. A rule which absolutely prohibits the award of a diploma to a candidate who fails one final examination paper is likely to produce some harsh decisions if it cannot be demonstrated that this requirement is essential for professional competence, maintenance of standards, etc. Fortunately, for such a candidate, there may be an element of discretion available to the human decision-maker which mitigates the effects of a rigid application of the rules.

A decision-making algorithm may have a type of quasi-discretion built into it by avoiding such single-inequality rejections. In other words, unless a particular characteristic is demonstrably essential, there should be at least two independent computable parameters, both of which provide evidence in favour of a certain course of action. These parameters are referred to below as "suggesting" and "confirming" indicators.

Thus, in Algorithm B, the existence of more normal surfaces orientated towards the viewing direction than away from it is a "suggesting" indicator in favour of the viewing direction. If one surface chosen at random is clear then this is a "confirming" indicator. The situation will arise when there is a "suggesting" indicator for a particular decision but no "confirming" indicator. In Algorithm B, the choice between the two options is made at random because, in that case, there is no third, independent indicator upon which to base the decision.

Results obtained from an implementation of Algorithm B tend to the conclusion that it is unsatisfactory to base a decision on viewing sense simply on the numbers of surfaces that are orientated in each of two opposite directions. Further information is required, but the results suggest that certainty about the numbers of "clear" surfaces is not necessary. The range of models tested was necessarily limited by the interests of persons with access to SUPERMODEL. The size of sample required for statistical accuracy is unlikely to be accessible until the software is commercially available. Until then, the only practicable method of avoiding an unrepresentative sample is to arrange for a Poisson distribution of the numbers of surfaces, as detailed in Table 5.2.

In order to improve the performance of decision-making algorithms it is clearly important to have reliable data on the ways in which human-beings reach certain decisions. Wright [55] gives details of some descriptive theories of decision-making and reports on process tracing techniques. Although this work is at an early stage, it is already clear that the more attributes and alternatives there are in a decision problem, the smaller is the percentage of these aspects considered. This suggests that a well-designed decision-making algorithm could be more consistent and closer to optimum choice than some individual human decisions.

Wright [op cit] refers to empirical evidence on the subjective assessment of probabilities. Lichtenstein and Fischhoff [56] found that most people are overconfident in probability assessment, especially with difficult questions. This finding has been duplicated many times and might usefully be employed to adjust probabilities that are to be the basis for a decision-making algorithm. Koriat et al [57] have shown that one reason for overconfidence is the failure to generate negative evidence. Several of the algorithms detailed above make extensive use of negative evidence (quantifiable undesirable characteristics) because it is technically easier to generate than evidence in favour of a particular strategy. If the work of Koriat is confirmed then such a technique is not merely expedient but soundly based.

The generation of negative evidence does, however, lead to considerations of computing cost. Discussion of Bayes´ theorm (Chapter 5) tends to the conclusion that, in certain circumstances, it is possible to evaluate objectively the value of additional evidence.

Unfortunately, this is rarely the case and the detailed design of a particular algorithm has to be based upon a subjective assessment of the relative importance of operating cost and quality of output. Alternatively, a commercial user might prefer to select from a range of algorithms, according to cost and performance.

11.3 Areas where work remains to be done

It is an unfortunate but inescapable fact of software development that, when the fundamental technical problems have been largely solved, one has merely reached the end of the beginning. The difference between a satisfactory research system and a commercially acceptable package is vast. For this reason, the suggestions for further work have been divided into two sections, the first of which lists the technical problems which are yet to be solved and the second discusses enhancements that may be required for commercial exploitation.

The SUPERMODEL modelling software is, in itself, an example of a usable research system which is not commercially acceptable because of the lack of a user-friendly input protocol. The following suggestions are concerned solely with the preparation of dimensioned engineering drawings from a stored shape description.

11.4 Unsolved technical problems

11.4.1 Algorithm C (for determining if an auxiliary view is necessary) is based upon a set of intuitive rules which have not been thoroughly tested. The design of a controlled test, the scientific analysis of the results and the modification of the algorithm in the light of these results are areas where substantial work is required.

11.4.2 The selection of sectional views (Algorithm D) is another area where extensive testing of a prototype algorithm is required. Indeed, it may call for some collaboration between Engineers and Applied Psychologists in order to answer the fundamental questions of under what circumstances a sectional view is more readily comprehended than an exterior view with hidden detail.

11.4.3 Further work is required in order to implement Algorithm E (selection of compound sections). This work would be largely statistical in nature as much of the evidence is contained in existing engineering drawings. The author has observed considerable variations of practice in different industries and between different branches of Mechanical Engineering. If these variations are found to be significant then the algorithm should provide specific options to particular users.

11.4.4 Algorithm F, for the clarification of output from the hidden-surface algorithm, is, as described, based upon the recommendations of British Standard 308. There are two types of work that remain to be done. Firstly, the detection of "ribs" (and other features that should not be cross-hatched) involves a process of shape-recognition which is not present in the algorithm that is described. Before such a sophisticated routine is appended to the algorithm, the question of whether the clarification process is really necessary should be addressed. The recommendations contained in this particular British Standard were published many years before geometric modelling techniques were developed and, for that reason alone, it may be wise to re-examine their intent.

Secondly, if some form of clarification and conventional representation is still thought to be desirable, it is necessary to consider how this aim may be achieved in relation to the user input to the geometric model. A surface may be designated for special treatment by the algorithm which prepares the orthographic view, or it may have to be detected by a suitable recognition procedure if the user is unable or unwilling to designate.

11.4.5 Algorithm G, for the selection of essential hidden detail, is based upon the assumption that there is a close correlation between the avoidance of ambiguity and the provision of detail for dimensioning. As this connection has not been proved theoretically, it would be valuable to investigate a suitably large sample of models in an attempt to show, with a specified level of confidence, that a particular orthographic view is unambiguous.

11.4.6 Research is required into the surface classification that is proposed in Section 4.5. The following questions might be posed. Are the six classes sufficient? When are they applicable? Does the relationship between surface classification and dimensioning scheme give acceptable results for a wide spectrum of models? Is the classification scheme suitable for objects that require significant geometrical tolerancing?

11.4.7 The dynamic model for dimensioning, described in Section 10.3, appears to be satisfactory for the limited range of models that have been tested. The precise formulation of a set of rules for the dimensioning algorithm is a more complex problem. The author is of the opinion that there is a case for treating certain dimensions as of secondary importance. Such dimensions would then be fitted into the scheme as and where space could be found for them. One significant disadvantage of the dynamic model is that any dimension that is added may need to displace all of those that have been inserted before it arrives at its equilibrium position.

11.4.8 Further work could be undertaken to investigate the relationship between input to the geometric model and the dimensioning scheme. One of the most frequently observed conventions on engineering drawings is implied symmetry. At present, a user of SUPERMODEL may define a symmetrical surface by using the "Mirror Active Surface" routine, but the only effect of this is to save time at the model input stage. Algorithm J will output a set of symmetrical dimensions for such a surface. This is both time-consuming and potentially confusing and should, therefore, be obviated by further development of the software.

Another convention disregarded by Algorithm J is that for repeated parts. These may be input to the model by using the "Move Active Surface" or the "Scale & Rotate Surface" routines. A rather complex algorithm is required in order to select one example of the feature and to generate the appropriate annotation.

A feature such as a blind, tapped hole would normally appear on an engineering drawing with all the relevant information (diameter, depth, thread type, pitch, etc.) shown together in a note. Algorithm J would display the diameter and depth as separate dimensions as there is, at present, no mechanism in the dynamic model for grouping this information together.

11.5 Requirements for commercial software

11.5.1 It is clear from user reactions to Algorithm I (Table 9.1) that what suits one organization is not necessarily welcomed by another, even one whose business appears to an outsider to be somewhat similar. In other words, some features of draughting policy are company-specific and their drawings, although comprehensible to other organizations, would not be acceptable to them. This implies that, as the actual software should not be tailored for each company, there must be various options available for selection by the user in order to satisfy his own requirements. For example, by default the software may be set to output an auxiliary view if one is required, whereas an option might be available which suppresses Algorithm C so as to ensure that no company drawings contain such views.

11.5.2 The preparation of orthographic views requires lengthy computation. This is a function of both the algorithm and the processing hardware. The latter factor may readily be improved by implementation on a more powerful machine, but there is evidence (e.g. Pafec "BOXER" software running on a VAX computer) that delays will still occur. It may be necessary to re-examine the hidden-surface algorithm and, perhaps, to exploit the superposition of edges in order to reduce computational time.

11.5.3 The process of dimensioning, using the dynamic model detailed in Chapter 10, is extremely time-consuming because of the number of consequential changes that may be required for each additional dimension. The main disadvantage, for the online user, is that for long periods he sees nothing more than reassuring messages. A commercially desirable facility would permit the user to view the progress that has been made with the dimensioning of a selected view, or the entire drawing. With refresh graphics, the consequential changes could be shown dynamically, provided the display proved neither irritating to watch nor detrimental to the progress of the main computation.

11.5.4 The automatic look-up and insertion of tolerances, as implemented for the interactive dimensioning software (Algorithm I), would need to be available as an option for Algorithm J. The author presumes that this is of major importance only for surfaces that have been classed as "hard interfaces". If that proves to be the case then it might be feasible to prompt a user to pre-tolerance such surfaces before the entire model is processed for the purpose of outputting an engineering drawing.

11.6 Potential benefits

A successful commercial application of this work would mean that engineering drawings could be produced from a stored shape description at a modest cost. As the traditional engineering drawing seems likely to remain an important means of communication within manufacturing industry, a net saving in this area would shorten the amortization period of the capital investment.

The average clarity of engineering drawings would be improved because of the amount of checking that is always carried out by the various algorithms. Both manual and computer-aided draughtsmanship are variable in quality, but an automatically produced engineering drawing would be of consistent quality. The information provided by the drawing could be further improved by appending a pictorial view of the object that is represented. As the hidden-surface algorithm functions equally well for viewing directions both normal and inclined to flat surfaces, the additional cost of a pictorial view is marginal.

The automatic production of engineering drawings could also be a helpful adjunct in the training of the type of draughtsman who will never have been called upon to use a drawing board. The software described above could readily be adapted to demonstrate both good and bad aspects of this form of communication.

There is still much to learn about the design of decision-making algorithms. A possible spin-off from this work could result in some improvements in the formulation of such procedures.

Lastly, the work reported here is fundamentally a process of synthesis: geometric model to working drawing. By studying the process it is hoped to be able to reverse it. In other words, what has been learnt from this work should assist with a new project (now under way) for generating a geometric model by analysing an engineering drawing.

REFERENCES

1 **Requicha, A.A.G.** "Representations for Rigid Solids: Theory, Methods and Systems", ACM Computing Surveys, Vol.12, No.4, pp.437-464, 1980

2 **Braid, I.C.** "Designing with Volumes", (2nd Edition), Cantab Press, Cambridge, 1974

3 **Baumgart, B.C.** "A Polyhedron Representation for Computer Vision", Proc.NCC., pp.589-596, 1975

4 **Barr, A.H.** "Superquadrics and Angle Preserving Transformations", IEEE Comput.Graphics & Appl., Vol.1, No.1, pp.11-23, 1981

5 **Requicha, A.A.G. & Voelcker, H.B.** "Solid Modelling: A Historical Summary and Contemporary Assessment", IEEE Comput.Graphics & Appl., Vol.2, No.2, pp.9-24, 1982

6 **Hillyard, R.C. & Braid, I.C.** "The analysis of dimensions and tolerances in computer-aided shape design", Comput. Aided Des., Vol.10, No.3, pp.161-166, 1978

7 **Baer, A, Eastman, C. and Henrion, M.** "Geometric modelling: a survey", Comput. Aided Des., Vol.11, No.5, pp.253-272, 1979

8 **Eastman, C.** "Representations for space planning", Commun. ACM, Vol.13, No.5, pp.253-272, 1979

9 **Braid, I.C., Hillyard, R.C. and Stroud, I.A.** "Stepwise Construction of Polyhedra in Geometric Modelling", Mathematical Methods in Computer Graphics and Design (Ed. Brodlie, K.W.), Academic Press, 1980

10 **Ekstrom, J.** "LYNX User's Manual", Redding Group, Ridgefield, Connecticut, 1980

11 **Anon** "2C L Part Programming Reference Manual", (2nd Revision), National Engineering Laboratory, Glasgow, 1973

12 **Anon** "GINO-F User Manual", Computer Aided Design Centre, Cambridge, 1976

13 **Newman, W.M. and Sproull, R.F.** "Principles of Interactive Computer Graphics", McGraw-Hill, New York (2nd Edition), 1979

14 **Bowyer, A. and Woodward, J.R.** "A Programmer's Geometry", Butterworths, London, 1983

15 **Angell, I.O.** "A Practical Introduction to Computer Graphics", MacMillan, 1981

16 **Walker, B.S., Gurd, J.R. and Drawneek, E.A.** "Interactive Computer Graphics", Edward Arnold, London, 1975

17 **Bresenham, J.** "A Linear Algorithm for Incremental Digital Display of Circular Arcs", Comm. of ACM, Vol.20, No.2, pp.100-106, 1977

18 **Faux, I.D. and Pratt, M.J.** "Computational Geometry for Design amd Manufacture", Ellis Horwood Ltd., Chichester, 1979

19 **Eastman, C.M. and Preiss, K.** "A review of solid shape modelling based on integrity verification", Comput. Aided Des., Vol.16, No.2, pp.66-80, 1984

20 **Klein, F.** "Geometry: elementary mathemaiatics from an advanced standpoint", Dover, 1939

21 **Barton, E.E. and Buchanan, I.** "The polygon package", Comput. Aided Des., Vol.12, No.1, pp.3-11, 1980

22 **Booker, P.J.** "A History of Engineering Drawing", Chatto and Windus, London, 1963

23 **Roe, J.W.** "Interchangeable Manufacture", Trans. Newcoman Society, Vol.17, pp.165-174, 1936

24 **Anon** "Dimensional analysis of Engineering Designs", Vol.1, "Components", H.M.S.O., London, 1948

25 **Foster, L.W.** "Geometric Dimensioning and Tolerancing: A working guide", Addison-Wesley, 1970

26 **Anon** "Engineering Drawing Practice", B.S.308, Parts 1,2 & 3, British Standards Institution, London, 1972

27 **Hillyard, R.C.** "Dimensions and Tolerances in Shape Design", Ph.D. thesis, University of Cambridge, England, 1978

28 **Light, R.A.** "Symbolic Dimensioning in Computer Aided Design", M.S. Mechanical Engineering, Massachusetts Institute of Technology, February 1980

29 **Adams, C., Gagg, S. and Tayer, G.** "Living Decisions", British Broadcasting Corporation, London, 1973

30 **Dixon, J.R.** "Design Engineering: Inventiveness, Analysis, and Decision Making", McGraw-Hill, 1966

31 **Kmietowicz, Z.W. and Pearman, A.D.** "Decision Theory and Incomplete Knowledge", Gower Publishing Company, Aldershot, 1981

32 **Mischke, C.R.** "An Introduction to Computer-Aided Design", Prentice-Hall, New Jersey, 1968

33 **Savage, L.J.** "A theory of statistical decision", Journal of the American Statistical Association, Vol.46, pp.55-67, 1951

34 **Colman, A.M.** "Game Theory and Experimental Games", Pergamon Press, Oxford, 1982

35 **von Neuman, J. and Morgenstern, D.** "Theory of games and economic behaviour", Princeton University Press, Princeton, New Jersey, 1947

36 **Lindley, D.V.** "Making Decisions", Wiley, London, 1971

37 **Fishburn, P.C.** "Mathematics of Decision Theory", Mouton, The Hague, 1972

38 **Cannon, C.M. and Kmietowicz, Z.W.** "Decision Theory and Incomplete Knowledge", Journal of Management Studies, Vol.11, No.3, 1974

39 **Barnes, A.W. and Tilbrook, A.W.** "The Theory and Practice of Drawing in SI Units", English Universities Press, London, 1970

40 **Sutherland, I.E., Sproull, R.F. and Schumaker, R.A.** "A Characterization of Ten Hidden Surface Algorithms", Comput. Surveys, Vol.6, No.1, pp.1-55, 1974

41 **Foley, J.D. and Van Dam, A.** "Fundamentals of Interactive Computer Graphics", Addison-Wesley, 1982

42 **Warnock, J.** "A Hidden Surface Algorithm for Computer Generated Half-Tone Pictures", University of Utah Computer Science Dept., TR 4-15, 1969

43 **Roberts, L.G.** "Machine Perception of Three-dimensional Solids", Massachusetts Institute of Technology Lincoln Lab., TR315, May 1963

44 **Appel, A.** "The Notion of Quantitative Invisibility and the Machine Rendering of Solids", Proc. ACM 1967 National Conference, pp.387-393, 1967

45 **Loutrel, P.P.** "A Solution to the Hidden Line Problem for Computer-Drawn Polyhedra", IEEE Trans. on Computers, Vol.19, No.3, pp.205-213, 1970

46 **Galimberti, R. and Montanari, U.** "An Algorithm for Hidden-Line Elimination", Comm. ACM, Vol.12, No.4, pp.206-211, 1969

47 **Braid, I.C.** "The Synthesis of Solids Bounded by Many Faces", Comm. ACM, Vol.18, No.4, pp.209-217, 1975

48 **Weiss, R.A.** "BeVision: A Package of Programs to Draw Orthographic Views of Quadric Surfaces", Journal of ACM, Vol.13, No.2, p.194, 1966

49 **Woon, P.Y. and Freeman, H.** "A Procedure for Generating Visible Line Projections of Quadric Surfaces", Proc. 1971 IFIP Congress, North-Holland, 1971

50 **Mahl, R.** "Visible Surface Algorithm for Quadric Patches", IEEE Trans. on Computers, C-21, pp.1-4, Jan. 1972

51 **Levin, J.** "A Parametric Algorithm for Drawing Solid Objects Composed of Quadric Surfaces", Comm. of ACM, Vol.19, No.10, pp.555-563, 1976

52 **Preiss, K.** "Algorithms for conversion of a 3-view drawing to the 3-D representation", Comput. in Industry, Vol.2, No.2, 1981

53 **Legard, D.** "Engineers go to work with some expertise", Computer News, No.43, p.36, 13 Sept 1984

54 **Leech, G.** "Semantics: The Study of Meaning", (2nd Edition), Penguin Books, 1981

55 **Wright, G.** "Behavioural Decision Theory: An Introduction", Penguin Books, 1984

56 **Lichtenstein, S., and Fischhoff, B.** "Do those who know more also know more about how much they know?", Organizational Behaviour and Human Performance, Vol.20, pp.159-183, 1977

57 **Koriat, A., Lichtenstein, S., and Fischhoff, B.** "Reasons for confidence", Journal of Experimental Psychology: Human Learning and Memory, Vol.6, pp.107-118, 1977

APPENDIX A

GRAPH THEORY

A.1 Introduction

Fig. A.1 and A.2 depict, respectively, part of a road map and a sectional view from an engineering drawing. Either of them can be represented diagramatically by means of points and lines as in Fig. A.3. The points A,B,C,D,E,F,G,H,I,J,K,L,M,N and P are referred to as Nodes and the connections between them are Edges; the whole diagram is a Graph.

The Degree of a Node is the number of Edges which have that Node as an end-point. A Node of degree two is referred to as a 2-Node. Nodes A,D,E,H,J,N and P are 2-Nodes. Nodes B,C,F,G,I,K,L and M are 3-Nodes. It is not possible for a Graph which accurately represents an orthographic view of a solid object to contain any 1-Nodes but Nodes of all higher Degrees are possible.

A 'Simple' graph is one in which there is never more than one Edge joining a given pair of Nodes. If there is more than one Edge joining a pair of Nodes then they are called Multiple Edges.

Multiple Edges can arise in an orthographic view in several ways. One example is a semi-circular feature where the same two Nodes are connected by both an line and an arc Edge. Another example occurs if the view is formed by processing a solid model of the object. If two

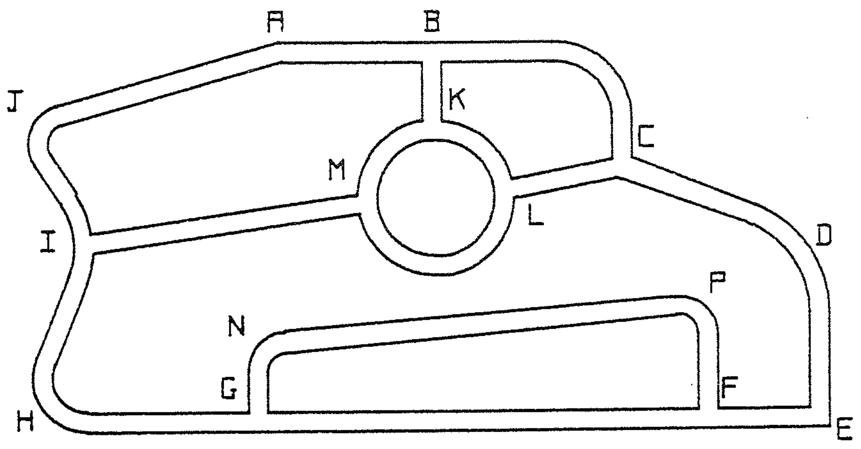


Fig. A.1 Road map

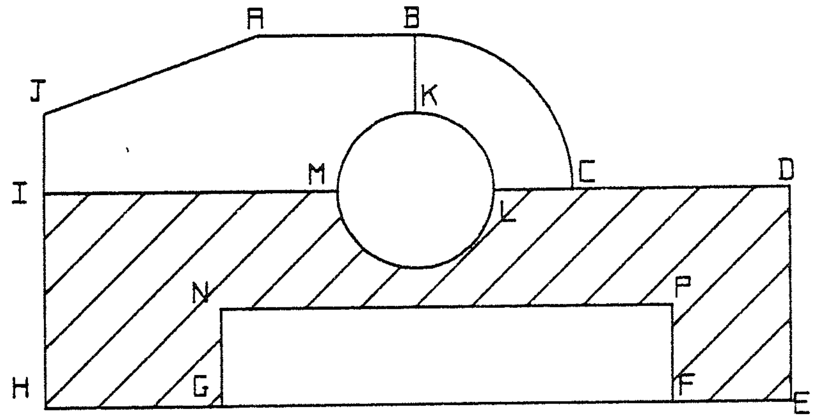


Fig. A.2 Sectional view from Engineering Drawing

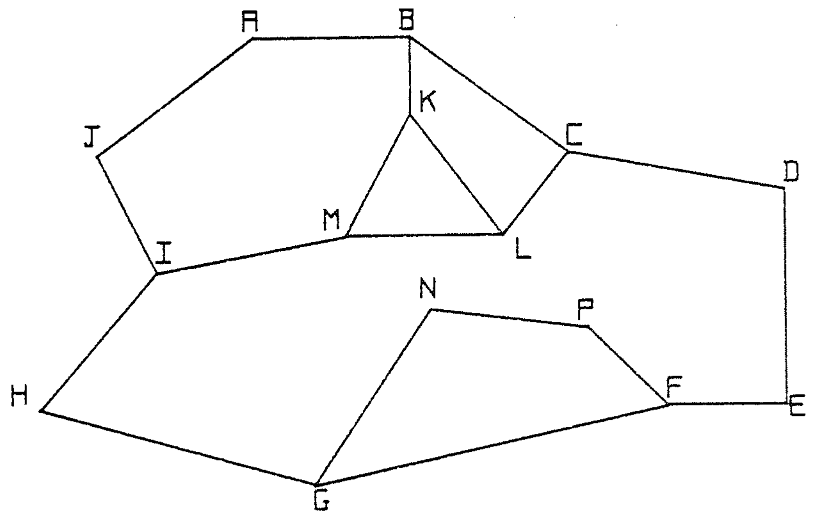


Fig. A.3 Graph equivalent to both Fig. A.1 & A.2

lines of equal projected length are superposed on a view then those lines may be regarded as Multiple Edges.

A Loop is an Edge which has both end-points at the same Node. This may occur on an Engineering Drawing whenever there is a complete circle. The geometric model may be structured so that the coordinates of the centre of the circle and the coordinates of some arbitrary point on its circumference are stored. Fig. A.4 illustrates that the circle is a Loop which has both end-points at this arbitrary Node.

A.2 Graphs, Digraphs, Paths and Circuits

Formally, a Graph G is defined to be a pair [N(G),E(G)], where N(G) is a non-empty finite set of elements called Nodes and E(G) is a finite family of unordered pairs of elements of N(G) called Edges. N(G) is called the Node-set and E(G) the Edge-family of G. Thus, in Fig. A.3 N(G) is the set {A,B,C,D,E,F,G,H,I,J,K,L,M,N,P} and E(G) is the family consisting of the Edges {A,B}, {B,C}, {C,D}, {D,E}, {E,F}, {F,G}, {G,H}, {H,I}, {I,J}, {J,A}, {B,K}, {K,L}, {L,M}, {M,K}, {C,L}, {F,P}, {P,N}, {N,G}, and {M,I}.

The following definitions are illustrated in Fig. A.5. A directed graph or Digraph I is defined to be a pair [N(I),D(I)], where N(I) is a non-empty finite set of elements called Nodes and D(I) is a finite family of

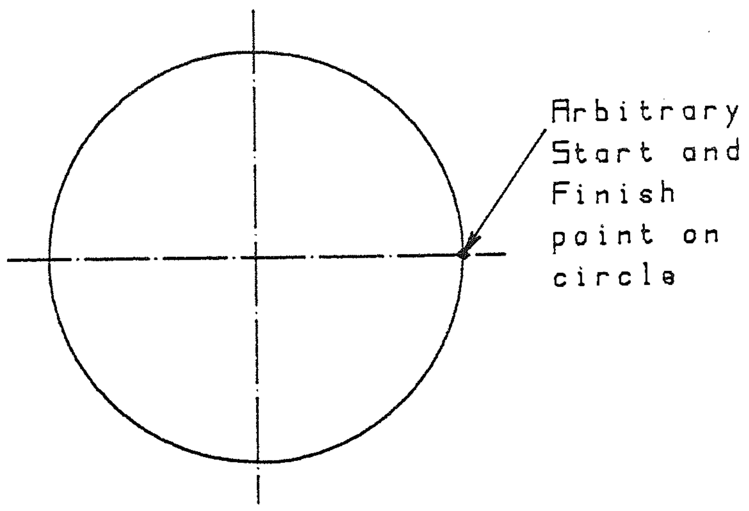


Fig. A.4 Occurrence of a loop on an engineering drawing

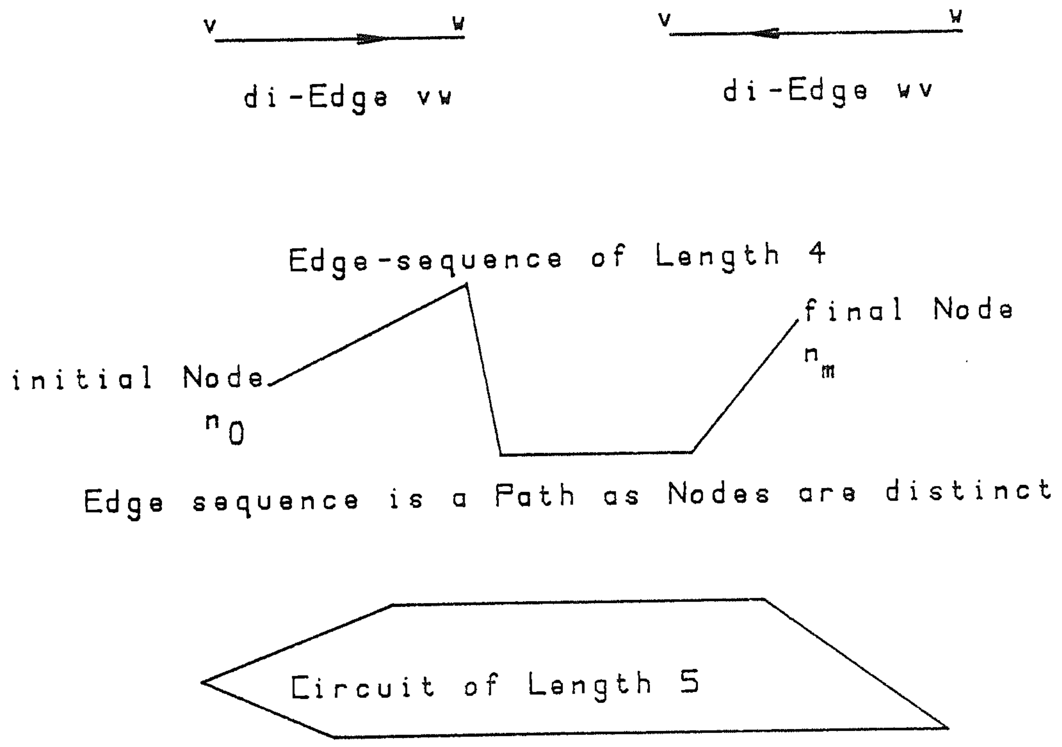


Fig. A.5 Definitions of Digraphs, Paths and Circuits

ordered pairs of elements of N(I) called di-Edges. A di-Edge whose first element is v and whose second element is w is called a di-Edge from v to w and is written vw. The di-Edges vw and wv are different. For the validation of a geometric model input to SUPERMODEL, a di-Edge has material on its left side. Thus, in Fig. A.2, the di-Edge ED has material to the left and, conversely, di-Edge DE has material to the right.

Given any graph G, an Edge-sequence in G is a finite sequence of edges of the form

$$n_0 n_1,\ n_1 n_2, \ldots,\ n_{m-1} n_m$$

It is clear than an Edge-sequence has the property that any two consecutive Edges are either adjacent or identical.

n_0 is called the initial Node and n_m the final Node of the edge sequence. The number of Edges in an Edge-sequence is called its Length. An Edge-sequence in which all the Edges are distinct is called a Trail. If, in addition, the Nodes n_0, $n_1, \ldots,$ n_m are distinct (except, possibly, $n_0 = n_m$), then the Trail is called a Path. A Path or Trail is Closed if $n_0 = n_m$, and a Closed Path containing at least one Edge is called a Circuit.

In Fig. A.2, the Circuit ED,DL,LM,MI,IH,HG,GN,NP,PF,FE represents a cross-section through a solid object. A valid cross-section consists of one or more Circuits.

APPENDIX B

HOMOGENEOUS COORDINATES

B.1 Introduction

Homogeneous coordinate representations of points, lines, and planes are particularly useful for describing and transforming geometric models. The representations are termed 'homogeneous' because each class of object is modelled by an equation which has no explicit parameter. The familiar explicit form of a two-dimensional line is $y = mx + k$ from whence its homogeneous (implicit) form is $mx - y + k = 0$

The homogeneous representation of the two-dimensional point (x,y) is written [wx wy w] where the symbols wx and wy are dipthongs (single numbers, not multiplications) and w is any nonzero scalar called the 'scale factor'. The mapping from a homogeneous point [wx wy w] back to its two-dimensional image is simply (wx/w, wy/w).

Three-dimensional objects are treated in an analogous fashion. The implicit form of the equation for a plane is $a_1x + a_2y + a_3z + a_4 = 0$

The homogeneous representation of the three-dimensional point (x,y,z) is written [wx wy wz w] for any nonzero value of w. The mapping from this homogeneous point back to its three-dimensional image is (wx/w, wy/w, wz/w).

B.2 Two-dimensional points and lines

B.2.1 A two-dimensional point (x,y) is represented by the row vector **p** = [wx wy w]. Any nonzero scalar multiple of this representation represents the same two-dimensional point.

B.2.2 A line in two dimensions is represented by a column vector:

$$\gamma = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

B.2.3 The condition that a point **p** is on a line γ is $\mathbf{p}\cdot\gamma = 0$. This is equivalent to the scalar equation: a(wx) + b(wy) + c(w) = 0. If the scalar product is not zero then **p** does not lie on the line but the product is proportional to the perpendicular distance of the point to the line, where:

$$\text{distance} = \frac{a(wx) + b(wy) + c(w)}{w\sqrt{(a^2 + b^2)}}$$

B.2.4 The line γ between two points $\mathbf{p} = [\, p_1 \ p_2 \ p_3 \,]$ and $\mathbf{q} = [q_1 \ q_2 \ q_3]$ is given by the vector product:

$$\gamma = \begin{bmatrix} q_2 p_3 - q_3 p_2 \\ q_3 p_1 - q_1 p_3 \\ q_1 p_2 - q_2 p_1 \end{bmatrix}$$

This is the result of solving the following implicit equations simultaneously:

$$ap_1 + bp_2 + cp_3 = 0 \quad \text{and}$$

$$aq_1 + bq_2 + cq_3 = 0$$

B.2.5 The point at the intersection two lines given by: $\gamma = \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix}$ and $\lambda = \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix}$

is given by:

$$\mathbf{p} = [\,(b_1c_2 - b_2c_1)\ \ (c_1a_2 - c_2a_1)\ \ (a_1b_2 - a_2b_1)\,]$$

B.3 Three-dimensional Points, Lines and Planes

B.3.1 A three-dimensional point (x,y,z) is represented by the row vector $\mathbf{p}$ = [wx wy wz w]. Any nonzero scalar multiple of this representation represents the same three-dimensional point.

B.3.2 A line is represented parametrically as a function of some parameter t which ranges from zero at one endpoint of the line to unity at the other endpoint. The formulation is:

$\mathbf{p} = [\ t\ \ 1]\ \mathbf{L}$ where $\mathbf{L}$ is a 2x4 matrix which may be found by requiring that:

$[0\ 1]\mathbf{L}$ = Row vector at one endpoint of line

$[1\ 1]\mathbf{L}$ = Row vector at other endpoint of line

B.3.3 A plane is represented by a column vector:

$$\gamma = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

B.3.4 The condition that a point **p** is on a plane is $\mathbf{p}\cdot\gamma = 0$. This is equivalent to the scalar equation: $a(wx) + b(wy) + c(wz) + d(w) = 0$. If the scalar product is not zero then **p** does not lie on the plane but the product is proportional to the perpendicular distance of the point to the plane, where:

$$\text{distance} = \frac{a(wx) + b(wy) + c(wz) + d(w)}{w\sqrt{(a^2 + b^2 + c^2)}}$$

B.3.5 Three points, (x_1, y_1, z_1), (x_2, y_2, z_2) and (x_3, y_3, z_3) determine a plane unless all are collinear. The simultaneous equations are:

$$awx_1 + bwy_1 + cwz_1 + dw = 0$$

$$awx_2 + bwy_2 + cwz_2 + dw = 0$$

$$awx_3 + bwy_3 + cwz_3 + dw = 0$$

In geometric modelling applications it is frequently necessary to determine the plane equation from a set of more than three points and inadvisible to select three at random. The following equations provide a computationally inexpensive solution for n points:

$$a = \sum_{i=1}^{n} (y_i - y_j)(z_i + z_j) \quad \text{where } j = i(\text{mod } n) + 1$$

$$b = \sum_{i=1}^{n} (z_i - z_j)(x_i + x_j)$$

$$c = \sum_{i=1}^{n} (x_i - x_j)(y_i + y_j)$$

The value of d is found by requiring any one of the n points to lie on the plane.

APPENDIX C

USER'S GUIDE TO "SUPERMODEL" MODELLING SOFTWARE

C.1 Introduction

SUPERMODEL is a geometric modelling program for the Intertec Superbrain QD microcomputer. It provides you with the ability to draw, edit, modify and annotate the surfaces for a boundary-representation of a solid object. The data that are generated are stored on diskette for possible processing by other applications programs. Your interaction with the computer screen is by means of arrow keys which drive a cursor around the screen. You may select drafting, editing and display functions from a menu of approximately 64 items. SUPERMODEL provides all the normal drafting functions, such as lines and arcs, redraw to different scales, searches for nearest points, lines and arcs, transformation of entire shapes, and the erasure of specified items. The main program calls overlaid subprograms for such operations as cross-hatching, curve-fitting, geometric properties, etc.

A surface may be incorporated into the geometric model by specifying its z-dimensions(s) and rotating it around a pre-defined axis. The total model may be rotated so as to give a view from any required direction.

C.2 Menu Map

CURSOR, LINE, ARC	DIMEN- SIONS	TEXT & SURFACES	SEARCH & ERASE	INPUT & OUTPUT	GEOMETRY	DISPLAY & MODEL
0 Cursor at Ref. Point	10 Cursor- Centre Screen	20 Input Text	30 Search for Point	40 Cursor to Ref. Point	50 Line and Arc Cross	60 Zoom In
1 Cursor at Start Point	11 Line - Arrow Finish	21 Text Dir´n & Size	31 Search for Line	41 Input Rect. Coords	51 Two Lines Cross	61 Redraw All
2 Cursor at Finsh Point	12 Line - Arrow Start	22 Change Active Surface	32 Search for Arc	42 Input Polar Coords	52 Two Arcs Cross	62 Scale for Redraw
3 Cursor at Centr Point	13 Line - Arrow Ends	23 Mask Surface	33 Search for Text	43 Output Arc Details	53 Tangent- Point to Arc	63 Set Model Axis
4 Full Line	14 Arc - Arrow Finish	24 Unmask Surface	34 Erase Line	44 Circle: Input Radius	54 Common Tangent- Arcs	64 Rotate Model
5 Dash Line	15 Arc - Arrow Start	25 Display one Surface	35 Erase Arc	45 Rect.: Input Sides	55 Line parallel to line	65 Store One Surface
6 Chain Line	16 Arc - Arrow Ends	26 Mirror Active Surface	36 Erase Text	46 Output Rect. Coords	56 Line perp. to line	66 Retrieve One Surface
7 Full Arc	17 Horiz./ Vert. Dimen.	27 Move Active Surface	37 Erase Surface	47 Output Polar Coords	57 Line radial to arc	67 Position Flat Surface
8 Dash Arc	18 Angle Dimen.	28 Rotate & Scale Surface	38 Cursor Step Size	48 Rect. Coords/ Datum	58 Cross- hatch Surface	68 Position Curved Surface
9 Chain Arc	19 Diametr Dimen.	29 Move Unmasked Surfaces	39 Arc- fitting	49 Polar Coords/ Datum	59 Arc- blending	69 Store and Exit

C.3 System initialization

Check that there is no diskette in either of the drives. Switch the Superbrain on at its mains switch. Set the Micronex selector switch on the left at the back of the computer to its central position. Following the instructions that have appeared on the screen, insert the diskette labelled 'SUPERMODEL' into Drive A (on the left). Now type: SM and hit the ENTER key. The screen will clear and the program version number and copyright message will be displayed, followed by this message:

```
Hit ENTER to initialise working diskette in Drive B.
```

A properly formatted diskette MUST be inserted into Drive B. When you hit the ENTER key some data will be written onto your working diskette in order to check it.

C.4 SUPERMODEL starting procedure

After system initialization, the next prompt that you will receive is:

```
Do you wish to retrieve a model file (ENTER=Yes) ?
```

The SUPERMODEL dialogue was designed so that you respond to a question that has a yes/no answer by simply hitting the ENTER key for 'Yes'. Hitting any other key indicates 'No'. In order to prepare a new model you might type 0. The program then responds with the following question:

```
New model. File name ?
```

File names may have up to eight characters: you should type a name and hit ENTER. SUPERMODEL creates two files on the diskette in Drive B. If the chosen name is TESTDRAW then file TESTDRAW.SMG is used to store the geometry and topology of the model and file TESTDRAW.SMT is used to store any text that is added to the drawing. The former file will occupy at least ten records of diskette storage and another five records are required for each new surface that is entered. The latter file requires one record for each text-string that is written on the drawing.

The next prompt invites you to input the model's initial scale (1.000 is the default value, i.e. full size):

```
Scale for redraw is 1.0  New scale (ENTER=No change ) ?
```

You should simply hit ENTER for full-size, 0.5 (ENTER) for half full-size, etc. If a scale of, say, 1 is selected, the following message is shown at the bottom of the screen:

```
SCALE: 1. ACTIVE - Graphics: 1 Text:A  USE KEYPAD
```

A blinking cursor appears at the centre of the screen.

C.5 Cursor Controls and Menu item selection

The four arrow keys at the right of the keypad move the cursor in the direction shown. Each arrow key moves the cursor one step. Initially the step size is 5 mm of screen movement (this would correspond to, say, 10 mm on the actual object if the scale was 0.5). If any of the arrow keys is held down for a second or so the cursor will appear to move continuously until the key is released.

An item from the SUPERMODEL menu is selected by typing the appropriate one or two digit code at the keypad. The digits are echoed at the bottom right of the screen. You should hit ENTER to select the menu item displayed or you may delete one or both digits (by using the DELETE key) or recommence the process by typing a third digit: the menu display returns to zero.

The cursor step size is changed by selecting menu item no. 38: the following prompt appears:

Step 5 mm at Scale=1. New cursor step (ENTER=No change)?

This menu item may also be selected in order to check the current step size.

Menu item 0 sets a datum point to which the cursor may be returned at any time.

It is quite possible for the cursor to 'get lost'. The you may, for example, have requested a search which results in the cursor jumping to some point, line, arc or text which is currently outside the area displayed on the screen. The cursor may be retrieved by selecting menu item 10 (Cursor to screen centre). The cursor will immediately reappear at the centre of the screen, whatever is currently on display.

C.6 Start, Finish and Centre Points

Menu item 1 specifies that the current cursor position is a 'Start' point. It may be used to define:

C.6.1 The start of a line

C.6.2 The start of an arc

C.6.3 One corner of a 'zoom' rectangle (see C.9)

C.6.4 A point from which measurements are to be made (see C.11)

C.6.5 The starting corner of a rectangle (see C.13)

Menu item 2 specifies that the current cursor position is a 'Finish' point. It may be used to define:

C.6.6 The finish of a line

C.6.7 The finish of an arc

C.6.8 One corner of a 'zoom' rectangle (see C.9)

C.6.9 The point to which measurements are to be made (see C.11)

Menu item 3 specifies that the current cursor position is a 'Centre' point. It may be used to define:

C.6.10 The centre of a circular arc or complete circle

C.6.11 The centre about which the 'active surface' is to be rotated and/or scaled (see C.15)

C.7 Surfaces

The primary subdivision of a SUPERMODEL file is referred to as a Surface. A set of Surfaces may be assembled into the boundary of a geometric model.
Data are stored in the currently 'active' surfaces. There is one active surface for lines and arcs (initially Graphics: 1) and another active surface for words and numbers (initially Text:A). Up to 38 surfaces are available for graphics; they are referred to by numbers 1-38. Up to 26 surfaces are available for text; they are referred to by letters A-Z.

Any of the graphics surfaces may contain any combination of the following types of lines and arcs:

C.7.1 Continuous (full) lines and circular arcs

C.7.2 Hidden detail (dash) lines and arcs

C.7.3 Centre or pitch (chain) lines and arcs

C.7.4 Dimension (arrow-head) lines and arcs

You may select a new surface at any time when the 'USE KEYPAD' prompt is visible on the bottom line of the screen. This selection is achieved by selecting menu item 22; the following prompt appears:

```
Active surfaces - Graphics: 1  Text:A.  Change to ?
```

At this point any other surface number or letter may be selected in order to change an active surface. The effect of this selection is that the search routines which operate upon graphics surfaces (menu items 30,31 & 32) will operate only on the surface that is selected. For example, if the graphics surface is 2 and line search (menu item 31) is chosen, only those lines which have been drawn into surface no. 2 will be examined in order to find the one that is nearest to the cursor. Hidden detail, chain and dimension lines and arcs may be drawn at any time and they will be placed in the active surface and searched when requested.

Text may be written at any time by using menu item 20. If the text surface is C then the next text written will be stored in surface C. When text search (menu item 33) is chosen, only those text-strings which have been written into surface C will be examined in order to find the one that is nearest to the cursor.

C.8 Text

Text is added to a SUPERMODEL drawing by selecting menu item 20. The following prompt appears:

Text ?

The input text is added to the drawing, starting at the current cursor position. The text is stored in the current active text surface.
Initially, the direction for writing text is left-to-right (angle 0 degrees) and the character height is set

to a screen size of 2.5 mm. This is also the pitch between characters (about the minimum for easy reading). If the initial scale of the display was 1.000 then all text-strings will have a full-size height of 2.5 mm. These settings may be changed by selecting menu item 21 which causes the following prompt to appear:

```
Text 0.0deg 2.5mm high. Newvalues (ENTER=No Change) ?
```

This menu item may be used simply to read the current settings. You may hit ENTER when you wish to continue, leaving the angle and height unchanged. In order to change the settings you should input the angle required, measured in degrees anti-clockwise from the left-to-right direction, and the character height (full-size mm). The next text that is written will be in the direction that was specified and the characters will be of the height specified. When the scale of the display is other than full-size the characters are scaled accordingly.

In order to search for a text the active text surface must be set correctly. If the active surface is other than that which is to be searched (say C), then you should select menu item 22 and the following prompt appears:

```
ACTIVE SURFACES - Graphics: 1  Text:A. Change to ?
```

You should input C and then the TEXT message at the bottom of the screen shows C. If you then select menu item 33, the cursor will move on to the start point of the nearest text and a message, such as this, is displayed:

```
Text at  90 deg., 4.0 mm high is indicated.   USE KEYPAD
```

The display of direction and size that are given are useful when adding text to a drawing, keeping it parallel to and/or of the same size as existing text. If required, the text thus found may be erased by selecting menu item 36.

C.9 Masking surfaces, Scale, Zoom and Redraw.

When the total drawing produced using SUPERMODEL gets complicated and makes use of many of the available surfaces, it is convenient to be able to clarify the display by masking some of the surfaces. This is done by selecting menu item 23. You are then prompted to input the number or letter of the surface to be masked. When you wish to unmask a currently masked surface you should select menu item 24.
Menu item 61 (Redraw All) generates the following prompt:

```
REDRAW - Unmask all surfaces (ENTER=Yes) ?
```

If you hit (ENTER) then the total drawing that you have made is redrawn.

Menu item 25 (Draw one surface only) functions by masking all surfaces, except that which is specified for display. SUPERMODEL automatically masks all the other surfaces.
Text surfaces can be masked or unmasked by selecting menu items 23 or 24 and typing in a letter (upper or lower case) rather than a number in response to the usual prompt.

Initially, the scale of the model produced using SUPERMODEL is as specified at the start of a session. If the initial scale was 1., a line that is, say, 60 mm long on the physical object that the model represents is displayed with a length of 60 mm on the screen. The scaling factor of the current display may be read at the bottom, left of the screen whenever you are prompted to use the keypad:

```
SCALE 1. ACTIVE SURFACES - Graphics:1 Text:A  USE KEYPAD
```

The scale for future redrawn displays may be set by selecting menu item 62. This will display the prompt:

```
Scale for redraw is 2.5   New scale (ENTER=No change) ?
```

If you have specified a text character height of (say) 5 mm, the characters will be drawn with this height only when the scale is 1. If the scale of the current display is (say) 0.50, then such characters will be 2.5 mm high on the screen. This tends to cause confusion especially when a very large or a very small scale model is to be drafted. A drawing of a car, for example, with a scale of 0.02 (1/50) requires a character height of at least 150 mm for legibility.

When text to be displayed on the screen is so small that it would not be legible, SUPERMODEL simply draws a line from the start to the finish of the text. This avoids time-consuming and pointless character generation.

SUPERMODEL includes a menu item which enables you to zoom into any part of the current display and have it enlarged so as to fill most of the screen. You should first define two diagonally opposite corners of a rectangle by using menu items 1 and 2 (start and finish points). The order in which these points are specified is of no significance. You should then select menu item 60 (zoom in) and that part of the picture that has been specified is drawn out to an enlarged scale. The precise scale may be read at the bottom-left of the screen. As the corners of the zoom rectangle may have been selected in an arbitrary fashion, the scaling factor is unlikely to be one that would normally be used for a drawing. The purpose of displaying its value is merely to give the viewer some indication of the true size of the object that is represented.

C.10 Retrieval of SUPERMODEL files

The data generated whilst the program is in use are stored each time that the active graphics or text surface is changed. When you select menu item 69 (Store & Exit) the data files are closed, ready for retrieval on a future occasion and a message is displayed, such as:

```
TESTDRAW model file closed. Use PIP to back up. etc.
```

PIP (Peripheral Interchange Program) is a CP/M program stored on the SUPERMODEL diskette.

In order to retrieve this file the SUPERMODEL program is re-run and when this prompt appears:

```
Do you wish to retrieve a model (ENTER=Yes) ?
```

you should hit (ENTER). The directory of all SUPERMODEL files on the working diskette will be displayed followed by the prompt:

```
Retrieve model. File name ?
```

A file named TESTDRAW is retrieved by inputting the name only. It is essential that you do NOT input TESTDRAW.SMG. SUPERMODEL checks the diskette directory and informs you if there is no file of that name on the diskette in drive B. In such a case you are invited to reinput the name or change diskettes. When the file has been identified the following prompt appears:

```
Scalefor redraw is 1.0    New scale (ENTER=No change) ?
```

After you have input the scale or accepted the default value, all the surfaces which contain parts of the model may be drawn on the screen. You may opt to display all surfaces or may select those that are to be displayed as each one is retrieved.

When this process is complete the usual prompt displaying the scale and active surfaces and inviting the you to use the keypad appears and drafting may proceed.

C.11 Input and Output of Coordinates

Up until now the descriptions of cursor movements have been in terms of pressing the arrow keys in order to move the cursor a pre-set number of millimetres at a time. Clearly this method is far from suitable for precision drawings. There are two menu items which enable you to type in the full-size dimensions that you require and have the cursor moved by the correct scaled amount. When you select menu item 41, the following prompt appears:

```
x, y movement from cursor position ?
```

If, for example, you wish to position the cursor a full-size distance of 50 mm to the right and 30 mm downwards, you would input:

```
          50, -30
```

in order to achieve this. Horizontal movements are positive to the right and vertical movements are positive upwards. SUPERMODEL adopts the conventions of coordinate geometry for all input and output of this type. The cursor will actually move by these amounts if the current display scale is full size. Otherwise the cursor movements are scaled up or down accordingly.

The alternative method of precise cursor movement is provided by menu item 42 (Input polar coords). The following prompt appears:

```
Radius, Angle movement from cursor position ?
```

The convention for the input of angles is the universal mathematical convention that angles are measured in an anti-clockwise direction from the positive x-axis. Angles are assumed to be in degrees.

The reverse process allows you to make measurements between any two points on the active graphics surface. You should find a point within the surface by using menu item 30. You should then select menu item 1 to denote a start point. Next, you find another point using menu item 30 and select menu item 2 to denote a finish point. You then select menu item 46 and the horizontal and vertical distances between the points is displayed.

Similarly, once you have specified a start and finish point, you may select menu item 47 and read the radial distance and angular position of the finish point relative to the start point.

C.12 Dimension Lines and Arcs

There are three types of dimension lines available:

C.11.1Line with arrow-head at finish	-	Menu item 11
C.11.2Line with arrow-head at start	-	Menu item 12
C.11.3Line with arrow-heads at eachend	-	Menu item 13

All lines must have their start and finish points specified before drawing. The dimension line with an arrow-head at each end is actually two separate, single-arrow lines. In order to erase such a line it is necessary to set the correct active surface, search for the line required with menu item 31 and then erase a line with menu item 34.

There are three types of dimension arcs available. All of them are drawn around the specified centre point in an anti-clockwise direction from the start point to the finish point:

C.11.4 Arc with arrow-head at finish	-	Menu item 14
C.11.5 Arc with arrow-head at start	-	Menu item 15
C.11.6 Arc with arrow-heads at each end	-	Menu item 16

As with dimension lines, the arc with arrow-heads at both ends is actually two separate arcs and two searches and erasures are required in order to remove the entire feature.

C.13 Parameterized Circle and Rectangle

The following menu items provide you with rapid methods for drawing a complete circle of any radius and a rectangle of any length and height with its sides parallel to the edges of the screen. Before selecting either of these items the active surface should be set to that required.

C.13.1 Circle: Input Radius - Menu item 44

A complete circle may be drawn by specifying the centre in the usual way and then selecting menu item 44 when the following prompt appears:

```
Radius of circle (mm) ?
```

When you have input the radius required SUPERMODEL determines the start and finish points for the circle before displaying the following prompt:

```
Full (1), Dash (2) or Chain (3) ?
```

When one of these is selected the circle is drawn starting at a point to the right of the centre and in an anti-clockwise direction.

C.13.2 Rectangle: Input Sides - Menu item 45

A rectangle may be drawn by specifying a start point for any one of the four corners and then selecting menu item 45 when the following prompt appears:

```
Length (x), Height (y) of rectangle (mm) ?
```

You are asked to specify the coordinates of the diagonally opposite corner relative to the start point. That implies that, if the opposite corner is to the left of and below the start point then both x and y would be negative. The following prompt then requests the type of line:

Full (1), Dash (2) or Chain (3) ?

When one of these is selected the rectangle is drawn in the active surface.

C.14 Geometry

SUPERMODEL provides eight geometrical routines:

Point at intersection of two lines	-	Menu item 51
Points at intersection of line and arc	-	Menu item 50
Points at intersection of two arcs	-	Menu item 52
Tangents from a point to an arc	-	Menu item 53
Common tangents to two arcs	-	Menu item 54
Line parallel to line	-	Menu item 55
Line perpendicular to line	-	Menu item 56
Line normal to arc	-	Menu item 57

The elements in question must be in the same surface. It is not possible, for example, to determine the intersection point for two lines if they are not in the same graphics surface. For most of the above routines it is necessary to use one or more of the search routines in order to identity the elements whose geometry is to be examined. The most simple routine to use is menu item 51 and ,therefore, this will be described first.

C.14.1 Intersection of two lines:

You should check that the active surface is set to that containing the lines and then search for each of the lines in turn, using menu item 31. When appropriate, you should confirm that the cursor is on the line required. You should then select menu item 51 and the cursor will jump to the precise intersection point of the two lines. If the lines are parallel, a message to this effect is displayed.

C.14.2 Intersection of line and arc:

You should check that the active surface is set to that containing the line and arc and then search for the line, using menu item 31, and for the arc, using menu item 32. When appropriate, you should confirm that the items found are those required. You should then select menu item 50 and one of two messages appears. If the line does not intersect the arc then a message to this effect is displayed. Otherwise, the cursor jumps to one of the two possible intersection and the following prompt appears:

Is cursor at intersection point required (ENTER=Yes) ?

If the cursor is at the point required, you should hit (ENTER). You may then specify that this point is whatever you require. If the cursor is not at the point required, you should hit any key other than (ENTER) and the cursor will jump to the other intersection point. The above prompt re-appears.

C.14.3 Intersection of two arcs:

You should check that the active surface is set to that containing the two arcs and then search for each arc in turn, using menu item 32. When appropriate, you should confirm that the arc found is the one required. You then select menu item 52 and one of two messages appears. If the arcs do not intersect then a message to this effect is displayed. If the arcs do intersect then the cursor jumps to one of the intersection points and the following prompt appears:

Is cursor at intersection point required (ENTER=Yes) ?

If the cursor is at the point required, you should hit (ENTER). You may then specify that the cursor is at whatever type of point you require. Otherwise, you should hit any key other than (ENTER) and the cursor jumps to the other intersection point. The above prompt is re-displayed.

C.14.4 Tangents from a point to an arc:

You should check that the active surface is set to that containing the arc. You then specify that the point from which the tangents are to be drawn is the start point. You should then select menu item 32 in order to find the arc required and, when appropriate, confirm in the usual way.

You should then select menu item 53 and one of two messages appears. If the start point is within the circumference of the arc then there are no tangents and a message to this effect is displayed. Otherwise, one of the two tangents is drawn and the following prompt appears:

Is the displayed tangent the one required (ENTER=Yes) ?

If the displayed tangent is not the one required, then you should hit any key other than (ENTER) and the alternative tangent is displayed. When the displayed tangent is the one that you require you should hit (ENTER). The displayed tangent will vanish but the finish point has been left set to the position that was accepted and you may then select any of menu items 4-6 in order to draw the type of tangent line that you require.

C.14.5 Common tangents to two arcs:

You should check that the active surface is set to that containing the arcs. You should then select menu item 32 in order to find both of the arcs required and, when appropriate, confirm in the usual way.

You should then select menu item 54 and one of two messages appears. If one arc lies entirely within the circumference of the other arc then there are no tangents and a message to this effect is displayed. Otherwise, there are either two or four tangents. One of them is drawn and the following prompt appears:

Is the displayed tangent the one required (ENTER=Yes) ?

If the displayed tangent is not the one required, then you should hit any key other than (ENTER) and one of the other tangents is displayed. When the displayed tangent is the one required you should hit (ENTER). The displayed tangent vanishes but the start and finish points have been set to the position that was accepted. You may now use any of menu items 4-6 in order to draw the type of tangent line required.

C.14.6 Line parallel to line:

You should check that the active surface is set to that containing the line before searching for the line, using menu item 31, and, when appropriate, confirming that the cursor is on the line required. You should then select menu item 55 and the following prompt will appear:

Offset for parallel line (mm) ?

The parallel line will be offset in a direction that is 90 degrees ahead of the start-finish direction of the original line. When you have input the offset required the cursor jumps to the end of a parallel line and the following message is displayed:

```
Startand Finish points stored.Select Line. USE KEYPAD
```

C.14.7 Line perpendicular to line

You should check that the active surface is set to that containing the line. You then search for the line, using menu item 31, and, when appropriate, confirm that the cursor is on the line required. You should then position the cursor at the point from which a perpendicular to the line is required and select menu item 56. The cursor jumps on to the line that was indicated and the following message is displayed:

```
Cursor is at the point requested.            USE KEYPAD
```

C.14.8 Line normal to arc:

You should check that the active surface is set to that containing the arc. You then search for the arc, using menu item 32, and, when appropriate, confirm that the cursor is on the arc required. You should then position the cursor at the point from which a normal (radial) to the arc is required and select menu item 57. The cursor jumps radially on to the arc that was indicated and the following message is displayed:

```
Cursor is at the point requested.            USE KEYPAD
```

C.15 Transformation of Surfaces

The active GRAPHICS surface may be transformed in three different ways and the transformation may be placed in the same surface or another surface specified by you:

Move/Copy active surface	-	Menu item 27
Scale and/or rotate active surface	-	Menu item 28
Mirror active surface	-	Menu item 26

C.15.1 Move/Copy active surface

This routine enables you to produce an exact copy of the active surface in the same position or any other position specified by the finish point. The first step is to use menu item 30 in order to find some suitable point on the active surface. This point is specified as the start point. You then move the cursor to the position where the point just indicated on the active surface should be moved to, and specify that this position is the finish point.
After selecting menu item 27, the transformation is drawn and the following prompt appears:

```
Number of surface for storing transformation (0=None) ?
```

Any graphics surface may be selected, including the active surface itself. If the active surface is selected then future transformations of it will include the features that are added at this stage.

C.15.2 Scale and/or Rotate active surface:

This routine enables you to produce a scaled version and/or a rotation of the active surface. The scaling factor operates upon radial distances measured from the current centre point. A positive rotation is anti-clockwise around the current centre point. The factor that you are asked to input is not an absolute scale but the number of times smaller or larger than the original that the transformation to appear. After selecting menu item 28, the following prompt appears:

```
Magnification, Rotation of active surface ?
```

When these figures have been input the transformation is drawn and the following prompt appears:

```
Number of surface for storing transformation (0=None) ?
```

Any graphics surface may be selected, including the active surface itself. If the active surface is selected then future transformations of it will include the features that are added at this stage.

C.15.3 Mirror active surface

This routine enables you to produce a mirror image of the active surface. The line about which the active surface is to be reflected must first be specified by defining a start point and a finish point which lie on this line. It is not necessary to draw an actual line

unless it is required for other purposes.
After selecting menu item 26, the transformation is drawn then then the following prompt appears:

```
Number of surface for storing transformation (0=None) ?
```

Any graphics surface may be selected, including the active surface itself. If the active surface is selected then future transformations of it will include the features that are added at this stage.

C.16 Storage and Retrieval of single surface

A graphics surface may be regarded as a pattern of lines and arcs which is a sub-division of the total model. Inevitably there will be some such patterns that occur so frequently that you wish to make use of the same pattern in other models. Similarly, there may be blocks of text which may be required on more than one drawing. The means to do this is provided by menu item 65 - Store Single surface. When this item is selected the following prompt appears:

```
Number or Letter of surface to be stored (0=Cancel) ?
```

When a number of letter is input the next prompt is:

```
Name by which surface is to be stored ?
```

The name selected may contain up to 8 characters. The specified surface is stored on the diskette in Drive B with the name that you have specified. This file is quite distinct and separate from the model name file that was specified at the start of the SUPERMODEL session.

A single surface that was saved by selecting menu item 65 may be retrieved by selecting menu item 66 - Retrieve surface. The following prompt appears:

Retrieve surface: File Name ?

When you input such a name it is checked to discover if it exists and, if so, the surface is drawn out to its original size and position. A retrieved surface is, at the moment, homeless. The next prompt invites the you to specify which surface of the current model it should be stored in. The retrieved surface is not automatically inserted into its original surface number: it can go anywhere. The following prompt appears after a graphics surface has been retrieved:

Surface for storing retrieved graphics (0=Don't Store) ?

When you have input a suitable number the retrieved graphics is merged with the contents of that surface.

C.17 Repositioning of surfaces

As a model grows or is modified it may happen that some features are below the bottom of the screen or off the left-hand side of the screen when a normal redraw (menu no. 61) is selected. This forces you to use the window routine (menu no. 60) in order to see these features. Menu item 29 provides the means to readjust the total picture so that the bottom left-hand corner of the zoom window becomes the global origin of coordinates.

The effect of this repositioning is that the redraw routine will now reveal the previously off-screen parts. If only certain surfaces are to be repositioned then they should be the only surfaces shown within the zoom window. For example, to reposition graphics surfaces 2,4 & 7, hide all other surfaces and then zoom into the rectangle which will reveal all the detail required. Then select menu item 29 and messages will appear indicating that surfaces 2,4 & 7 are being moved. If you then redraw the entire picture (including previously hidden surfaces) you will see that only surfaces 2,4, & 7 have been repositioned.

C.18 Cross-hatching

On an engineering drawing it is common practice to cross-hatch sectional views in order to indicate the area where material has been cut by the sectioning plane. In other types of graphical work cross-hatching may be used in order to emphasise a particular feature.

SUPERMODEL provides a routine for cross-hatching the active graphics surface, subject to the following restrictions:

C.18.1 The complete surface will be cross-hatched. If a section through a solid object is to be cross-hatched then the sectional view only must be drawn in a particular surface. Additional continuous lines must not be present.

C.18.2 The cross-hatching routine ignores dash, chain and arrow lines and arcs. Such features may be included in a surface that is to be cross-hatched but only the continuous lines and arcs are considered to define the area(s) that are to be processed.

C.18.3 The continuous lines and arcs must 'bound' one or more closed shapes. It is obvious that an infinite area cannot be cross-hatched but, in practice, it is all too easy to draw such a shape.

C.18.4 The cross-hatching routine can handle any complexity of shape, provided the bounding requirement is met. For example, five concentric circles will be cross-hatched so as to produce an inner cross-hatched circle surrounded by two cross-hatched rings.

The cross-hatching routine CANNOT handle sections which (as is often done) show visible lines behind the section plane. Such lines do not bound the area which is to be cross-hatched and must be added after the cross-hatching has been done.

When you select menu item 58 the geometry of the active graphics surface is checked to determine whether cross-hatching is possible. The check will be unsuccessful if the elements do not bound an area and the following message is displayed:

```
Surface cannot be cross-hatched. No area(s) enclosed.
```

If there is a small gap between two elements it may still be possible to cross-hatch the shape by closing the gap. The cursor appears at the gap and the following question is output:

```
Cursor indicates a small gap. Close it (ENTER=Yes) ?
```

You may make a positive response to this question if, for example, the gap has come about accidentally because of some lack of precision in geometrical computations.

If the check is successful immediately or when small gaps have been closed up, the following prompt appears:

```
Angle for cross-hatching (degrees, anti-clockwise) ?
```

When you have input a figure the next prompt is:

```
Pitch of cross-hatching (mm, full-size) ?
```

You should respond by entering the full-scale distance required between the parallel cross-hatching lines. The next prompt is:

```
Full (1), Dash (2) or Chain (3) ?
```

The program then computes the overall dimensions of the shape that is to be processed. While it is doing that the following message is displayed:

```
Computing bounds of cross-hatching lines...
```

The routine computes the positions for the first and last cross-hatching line and then draws a series of lines at the angle and pitch specified. If the lines would fill a complete surface when they are stored on diskette or when the surface has been completely cross-hatched you will be asked to input the number of the surface in which they are to be stored. You may opt not to store the lines by simply hitting (ENTER).

C.19 Arc-blending

Menu item 59 provides a routine for automatically drawing an arc which is:

C.19.1 Tangential to two lines which meet at a point or

C.19.2 Tangential to a line and an arc which meet at a point or

C.19.3 Tangential to two arcs which meet at a point.

The above elements must be in the same surface and no other elements may start or finish at the intersection point that is to be specified. You should move the cursor unambiguously near to the point at which the arc is to be inserted and then menu item 59 is selected. The cursor jumps on to the nearest point within the active graphics surface and the following message appears:

Radius of blending arc (mm) ?

Once you have input a number, the centre of the blending arc is computed and the cursor is repositioned there. Parts of the existing elements are erased and replaced by a blending arc. The following message appears:

Is blended arc acceptable (ENTER=Yes) ?

If you hit (ENTER) the modified elements and the new arc are stored. Otherwise, the blending arc is removed and the elements are restored to their original state. If, for various reasons, it is not possible to insert a blending arc of the specified radius, an appropriate message is displayed.

C.20 Arc-fitting

SUPERMODEL provides a routine for fitting a series of arcs through defined points within a surface. The arcs are computed so that there is no change of gradient at the points, the initial gradient being user-specified. The routine operates subject to the following restrictions:

C.20.1 The active graphics surface must only contain lines.

C.20.2 The lines must be contiguous. In other words, the finish point of the first line must be the start point of the second line; the finish point of the second line must be the start point of the third line, etc.

C.20.3 The fitted arcs will have the same line-style as the lines.

When you select menu item 39 the following message will appear:

```
Hit SPACE bar to move cursor. Hit ENTER for start point.
```

The arc-fitting program actually computes the centres for a set of circular arcs which blend at the ends of the lines. In order to begin the computation you must specify the tangential direction for the first arc:

```
Angle of tangent at start  (deg., anticlockwise) ?
```

The program will accept any value for this initial direction except one which is the same as the first line (since an arc of infinite radius would be required). A poor choice of direction will produce an unnecessarily serpentine curve. There is usually a small range of starting angles which produce a gentle curve throughout the series of end-points.

When the set of arcs has been drawn you will be invited to store them:

Number of surface for storing arcs (ENTER=Don't store) ?

You should select an empty surface or one that will not be required later because only the displayed arcs will be written into this surface. If you are experimenting with different starting angles and do not wish to store the result, you should simply hit the ENTER key.

C.21 Placement of flat surface into geometric model

You should first check that the active graphics surface contains exactly what you wish to place into the model. Next, define the start and finish points so as to indicate an axis around which the surface may be rotated. If no rotation is required then there is no need to define any particular axis.

When you select Menu item 67 SUPERMODEL checks that the activesurface does bound an area. If this check fails then you will be informed of the fact. Next, a check is carried out to find if the surface is selfintersecting. If an intersection point is found then the cursor will jump to that position and you will be informed that the surface cannot be placed into the model.

If both checks are successful then the following list will appear:

```
SURFACE CLASSIFICATION
1  Hard Interface
2  Soft Interface
3  Guide
```

4 Dam

5 Boundary

6 Signal

Class number ?

If you are able to identify the function of the surface that you have specified then input the appropriate number. Otherwise input 0 and the default classification (Boundary) will be set.

The next prompt is:

Open (1) or Solid (2) side of surface ?

The purpose of this input is to orientate the surface correctly in the geometric model. If the view that you have drawn is as the surface would be seen from outside the solid object then you should input 1. If the view is as seen through the object then you should input 2.

The next prompt is:

Z coordinate of surface ?

You should determine the correct value for your surface, assuming that it is to be rotated around a start-finish axis which is in the same z-plane.

You are then invited to specify the rotation of the surface:

Rotation around Start-Finish axis (degrees) ?

The direction of rotation is anti-clockwise when looking from the start point towards the finish point. Type in the value required and hit ENTER.

When you input the value required, the data for the entire surface are updated and written to diskette but the display will not be affected. You may check the placement of this surface by selecting a redraw after the entire model has been rotated around some global axis. This process is detailed in section C.23.

C.22 Placement of conical surface into geometric model

You should first check that the active graphics surface contains just two circles or circular arcs which represent the ends of the conical surface that is to be placed into the model. Next, define the start and finish points so as to indicate an axis around which the surface may be rotated. If no rotation is required then there is no need to define any particular axis.

When you select Menu item 68, SUPERMODEL checks that there are two circular items only. If this check fails then you will be informed of the fact, otherwise the following prompt will appear:

```
Z coordinates of first arc, second arc ?
```

Type in the values required and hit ENTER. You will then be prompted to specify the number of the Class of surface that you have input. This procedure is detailed in Section C.21 above. In addition, if the surface is cylindrical, you will be asked if it is threaded. This classification is used by certain application programs which produce engineering drawings from the geometric model but it will not be acted upon by the SUPERMODEL display routines.

The next prompt is:

Are arcs Concave (1) or Convex (2) ?

If the solid material is on the opposite side of the arc to its centre then input 1 for Concave, otherwise input 2 for Convex. The last prompt is:

Rotation around Start-Finish axis (degrees) ?

The direction of rotation is anti-clockwise when looking from the start point towards the finish point. Type in the value required and hit ENTER. The data for the conical surface are not modified to include two straight edges between the ends of each arc. In the case of two circles these edges are coincident and will appear as one line when the model is displayed by selecting Menu item 61.

C.23 Display of geometric model

Menu item 63 lets you specify the orientation of the global axis about which the model is to be rotated. It is convenient to think of this axis as a needle pushed through the model. Position the cursor on the screen at a point though which the axis is to pass. When you select Menu item 63, the following prompt appears:

Angle for axis of rotation (from cursor) ?

Type in the angle in degrees, using the usual anti-clockwise convention, and hit ENTER. A chain line will be drawn on the screen to indicate the orientation that you have specified.

The z-coordinate of this axis and the rotation of the model around it are specified by selecting Menu item 64. These values default to zero initially and the following prompt appears:

```
Axisofrotationis Z = 0.  Rotation0 deg.New values?
```

The model will be rotated around the axis that was input when you last selected Menu item 63. A positive rotation is anti-clockwise around the axis as viewed from the cursor in the direction that was specified. Type in the values required and hit ENTER.

The next time that you select Menu items 23,24,25,60, or 61 you may opt to have the surfaces(s) rotated around the global axis.

Verification of the geometric model and application programs are detailed in separate documents.