SPATIAL MODELS IN COMPUTER-BASED INFORMATION SYSTEMS

by

Adrian Lynn Thomas, B.Arch., M.C.D

Thesis presented for the Degree of Doctor of Philosophy

of

Edinburgh University

April 1976

I hereby declare that this Thesis

"Spatial Models in Computer-Based Information

Systems" has been entirely composed by myself.

_____

ABSTRACT

From a series of initial studies in the area of computer cartography a dual data structure was evolved based on matrix representation of graphs and the use of boolean expressions. This data structure was used principally to represent zones in space though, by using boundaries of zones, it was possible to create line networks.

The original idea was to use the boolean expressions as an input language for creating volume and area descriptions and to use the graph matrices for internal manipulation and creating graphic output. However, a way was found to interpret the boolean expression directly into the form of graphic output suitable for the raster scan displays given by television monitors.

The software implementation of this process was very slow but, with the current developments in integrated circuitry, it suggested a way of creating a new form of parallel display processor. This possibility was investigated initially as a general processor to carry out several related spatial operations and then, finally, merely to create displays.

The applications depend on (1) the general nature of the data structure used and the possible graphic languages it makes possible and (2) the real time manipulation of displays. In the case of three-dimensional scenes, this includes an automatic hidden line and hidden area removal capability.

The particular applications which have been considered include the fast access and display of maps and technical drawings from planning, architectural and engineering data bases; the real time generation of displays for training simulation; the preparation of animated films for teaching and entertainment; the control of numerically-controlled machine tools; and solving the placement problem in computer-aided design work and overlap problems in type setting and map annotation.

To

Elizabeth Susan Thomas

ACKNOWLEDGMENTS

The origin of this work was a fifth year study "Models in Design Procedure" carried out while a student of Architecture at Liverpool University.

The initial work on computer models was carried out in "the Laboratory for Computer Graphics and Spatial Analysis" at Harvard University where the first version of the OBLIX program was written and, in collaboration with T. C. Waugh, the early work on the GIMM system was undertaken.

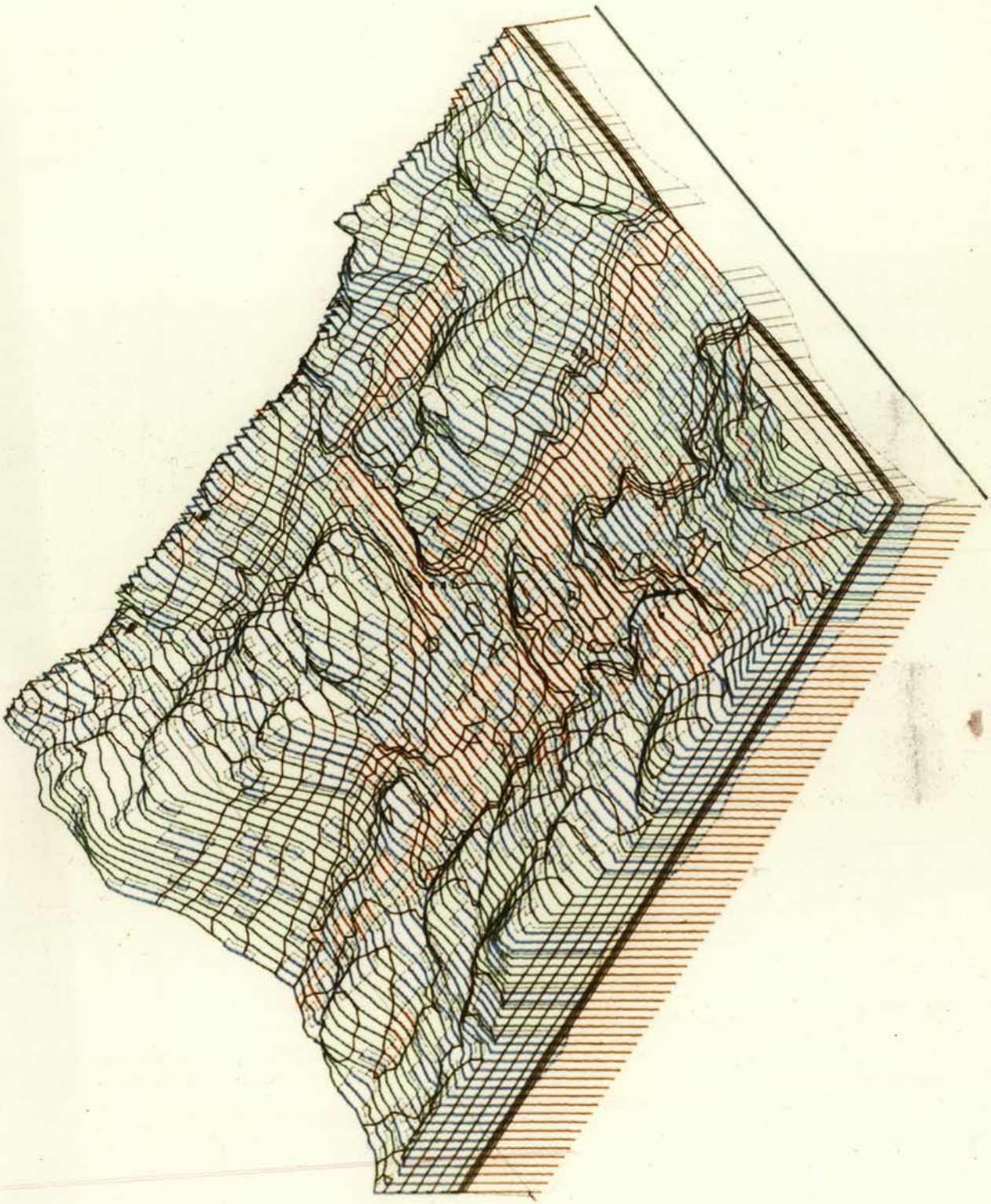
The period of study in Harvard University was carried out on a J. F. Kennedy Memorial Fellowship; it was the exploratory work conducted during this time which outlined the present area of study. In the Laboratory for Computer Graphics the author acknowledges with thanks help and guidance received from Professors A. Schmidt, C. Steinitz, P. Rogers, Professor H. Fisher and Professor W. Warntz.

In Edinburgh University the opportunity to undertake this work was provided by Professor J. T. Coppock. Dr. J. Oldfield, now Professor of Electrical Engineering in the University of Swansea, provided informal but valuable guidance in the area of Computer Science from 1970-1973. The author also wishes to thank Dr. J. Gray for his advice following this period. Dr. J. Downie contributed Appendix A describing a simulation he prepared of the display processor discussed in Section IV which clarified several of the issues which were being examined. This work is acknowledged with particular gratitude since he undertook it at a critical period in his own work.

In Heriot-Watt University, where research on a Hardware Display Processor is being continued, thanks are offered to Professors Heath and Nicol for providing facilities to continue the research described in Section IV.

Finally, for the principal load of supervision and support over six years, thanks are offered to Professor J. T. Coppock in the Geography Department of Edinburgh University.

CONTENTS



INTRODUCTION

INTRODUCTION

When the research project presented in this thesis was initially being formulated, it was not clear in what direction it would be possible to take the investigation, nor was it clear what were the most profitable topics on which to concentrate the main effort. Consequently, the scope of the project was left as wide open as possible and this is reflected in the title: "Spatial Models in Computer Based Information Systems". The background desire which has been retained throughout, was a wish to automate many simple activities which depend on the use of drawings. In these activities drawings are employed both as a way of storing and communicating information, and as a means of analysing information. Drafting is a labour intensive operation and consequently, because the activities appear simple, they are both tedious and time consuming. However, what appears simple to a person trained to use drawings turns out in practice to be far from simple to automate. It was hoped that if the appropriate approach to computer graphics and to the computer representation of spatial information could be found, then these 'simple' manual operations would also become simple to automate.

It was not possible to work in a vacuum. Consequently, it was necessary to consider the use of graphics in particular situations. The original areas of application were taken from the design and planning activities carried out by City Planners, Architects, Landscape Architects, and Civil and Structural Engineers. It was the predominance of spatial information processing in these areas of work which made the use of drawings both for storing and communicating information a necessity.

INTRODUCTION

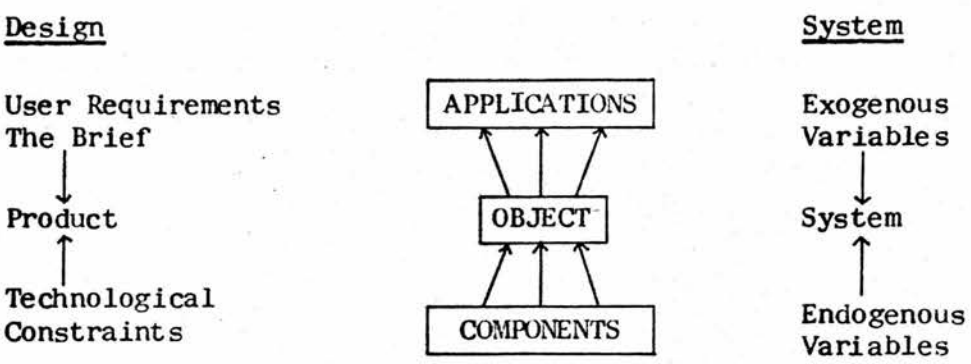
The study of automated cartography for planning purposes extended the area of application to also include many aspects of Geographical research where the map is an important form of communication and analysis.

The approach which was originally taken to the work presented in this thesis was that of a designer. In simple terms, the design process starts with a collection of components and assembles them into a working arrangement to give a product. The product, in this case an automated drawing system, will itself be a component which can be used to create other products and therefore must satisfy a set of external demands. Generally speaking, the process of design involves the use of known properties of a collection of components, to achieve some specified goal set up by analysing a particular area of needs.

Expressed in this way, it can be seen that there are very close parallels between the design approach, which is normally associated with physical components and physical products, and the systems approach which is more concerned with abstract components. In both cases a system and a design-product can be broken down into parts and in both cases the system and the design-product have to relate to a larger environment.

It is convenient to relate the design-product or system, its environment, and its components in the arrangement shown in Figure 1.

INTRODUCTION



Systems Analysis and the Design Approach

Figure 1

It can be seen that there are two sets of forces which restrict the freedom of the designer. Using the layout in Figure 1, there are the needs of the users acting from above. These are usually formulated as a design brief. Secondly, acting from below, there are the constraints which result from the properties of the components and the permissible ways in which they can be related. These provide the technological constraints within which the designer must work in his attempts to satisfy the external goal. The brief loosely corresponds to the exogenous variables, in the terminology of systems theory, and the technological constraints, in a similar way, correspond to the endogenous variables.

Descriptions of the design process tend to emphasise the progression down from the general statement of needs given in the brief to the solution of technical problems necessary to satisfying the brief. However, the analysis from the top, working downwards in this way, can only succeed if it leads to a specification, or classification which corresponds to known components. Where this does not occur, then the

process has to be reversed. Working upwards, new experimental combinations of basic components have to be tried, or the original definition of the product has to be modified to give a design goal which it is possible to implement.

Where design work is undertaken using components from a new technology, then there will be far less known about the potential combinations of these components. Consequently, a trial and error approach has to be employed to a much greater extent. In extreme cases, where very little is known, the product may cease to be defined in any effective way; possible products must be created first and their applications determined second. New technology requires a body of this type of research to be built up before designers can have the knowledge to undertake predefined design projects with any professional confidence of success. It is arguable whether an approach to design can exist without there being some form of preconceived objective. However, in the case of automatic data processing, new products or services have already emerged, which a top down approach based on narrow definitions of existing products would have failed to recognise.

It is only by taking a general view of a new technology such as automatic data processing that it becomes possible to provide a framework which is not structured by existing products and the technology on which they are based. Adopting such a strategy means a change to a systems analysis approach, since the aim becomes that of finding a system description which is capable of showing the consequences of a change in technology, but which is not structurally altered by the

change.

The title places this study firmly within the framework of systems analysis. This, perhaps, has to be done in an investigation of computer techniques and their applications. On one hand, the advantage of adopting a systems approach is often greatest when dealing with complex situations but this can only be made into a practical proposition if computers can be used. On the other hand, the nature of computer processing is based on the use of one kind of system to model the behaviour of another.

Presenting an aspect of the real world as a system is as much the creation of an artificial construction as designing a product. Its form depends on a preliminary choice or definition of components and the specification of the relationships they hold to each other. Though a variety of systems is defined for different purposes in the main body of the thesis, broadly speaking they can be divided into one of two groups. Firstly, there are the subsystems which are used in the representation and storage of spatial information. Principally, these correspond to components of a computer system. Secondly, there are the 'containing' systems, or the ways of modelling the context in which the spatial information is used.

The main area of study was the computer system itself and the aim in this context was to investigate ways in which spatial properties and their inter-relationships could be modelled for machine use. This part of the work was a design exercise and the result was a product or potentially a product. The 'containing' systems were considered because they provided

the brief for this design work. Two views were taken of these external systems. The first was short term where there was assumed to be little effect made on the external system by new techniques. This restricted the design work to providing alternative forms of existing products. The second took a longer term point of view where the effect of new techniques had had time to create changes in the context in which they were used. This approach raised general questions associated with the use of any new technology, such as new products, and opened up a much wider range of possibilities then the previous approach.

The complexity of the technical aspects of spatial information processing by computer made it necessary to define a few related projects of manageable size as the central theme of the research. In spite of this restriction, the aim was still to make the results of this design work applicable to as wide a range of uses as possible. Adaptability seemed to be a valuable asset for any product which did not already possess an established market.

The aim in the technical studies was to concentrate on the problems of representing zones in space. In three dimensions this corresponds to defining the spatial extent of objects and defining their relative locations and orientation. In two dimensional space the corresponding problem was to define regions on a surface such as the areal zones in a choropleth map. The formalism used for defining zones was chosen because it had already been successfully applied to line networks and had been used in a variety of spatial interaction models. This meant that if similar procedures could be developed for volumes and areal

zones as those used with line networks, then it might be possible eventually to merge all of them into a single, general purpose sub-system.

Following its initial definition, this project passed through several intermediate stages of development. These corresponded to modifications in the attitude taken to the various systems which affected the research. In the first stage, the aim was to implement a selection of spatial analysis techniques using computing, where the computer was merely regarded as a black box capable of carrying out the sequences of necessary numerical calculations. The second stage started when it was realised that even within the few programs developed for the special purpose of applications of stage one, many common processes were being employed. This changed the emphasis to developing a computer system which could provide the basic processes for a wider range of applications using spatial data. The third and final stage followed from the second stage when the requirements of the people likely to be using such a computer system were added to the list of considerations which had to be taken into account. This led to a variety of containing system structures being considered at different levels of generalisation and for different purposes and led to the approach which is currently being employed.

In presenting the final results of this work, this evolution of ideas has been masked to a considerable extent by the structure imposed from the analysis of the final stage which has been used to organise the material in the thesis. In spite of this, this sequence can be

detected in the nature of some of the examples still included in the text.

Thesis Structure

The Thesis is divided into five Sections:

I The Context

II Spatial Models

III Zones in Space

IV Hardware Possibilities

V Applications and Conclusions

The Context

In the first Section the broad issues which affect the use of computers within the existing structure of an organisation are considered. In Chapter 1: 'The Information System', the general concepts which make such an analysis possible are outlined. These centre round the idea of information and information flow. It is possible to construct many hypothetical arrangements which, from any practical point of view, would be impossible to implement. In Chapter 2: 'The Choice of Information System', the idea of cost, particularly where it affects new products and new production processes, is briefly examined. The general forces which these two Chapters review have presumably applied to the development of existing computer facilities and in 'Product Definition in Computer Based Systems', the growth of these facilities is presented from this point of view. This Chapter incidentally provides a brief description of the components available for the design work presented

in later sections. In the final Chapter in this Section: 'Computer Based Information Systems and Future Planning', the global implications of developing computer technology are considered. This approach was necessary because it appeared to be the only way to assess the ultimate purpose which future advances in computing could serve and therefore estimate profitable lines of development which present costs and other 'practical' considerations might not support.

Spatial Models

In the second Section a series of studies is presented which examines the geometrical operations necessary to create line drawings. In Chapter 5: 'Spatial Models as Data Structures', the general nature of this process is discussed and the relationship between the external representation of information and that used in the computer is described. Chapter 6 describes a program written to produce a specific form of line drawing. Chapter 7 shows how this work led to the development of a program system, the first stages of which acted as a research tool to test out the ideas presented in the following Section.

Zones in Space

In the third Section the representation of zones in space is taken as the central theme for study. The investigation centres round two data structures: the first based on the matrix representation of graphs and the second on the use of boolean expressions. The first topic considered was the formal properties of the matrices and the way they could be used to carry out a sequence of simple operations on spatial descriptions.

INTRODUCTION

The use of boolean expressions was introduced when a way of entering spatial descriptions into the machine system was required. The final Chapter in this Section discusses the general problem of converting these internal data structures into acceptable forms of graphic output. In particular, a way is outlined by which boolean expressions - used to enter data - can be interpreted directly into a form of display. This process initially appeared as a useful data checking device. On further consideration, it appeared to provide the basis for a very powerful display procedure if it could be implemented at the hardware level of a system.

Hardware Possibilities

In the fourth Section the idea of implementing the display of boolean expression forms of spatial description at a hardware level of a computer system is considered more carefully. In Chapter 13, the idea that the more processes which a particular piece of hardware can be used to carry out, the more reasonable it becomes to build it, is taken as a starting point to the investigation. As many of the operations as possible already considered as software processes are restructured to be carried out by the schematic machine. In Chapter 14, the alternative approach is adopted and an attempt is made to simplify the display operation to as simple a form as possible. In the following Chapter, this line of development is taken several steps further by simulating the basic procedure on an Hybrid computer and this leads, in the final Chapter, to a relatively detailed specification of several digital implementations of such a device.

Applications and Conclusions

The applications for the various aspects of the work presented in this thesis divide themselves into two groups. The first can be found in the construction and use of data-bases designed to hold spatial information. These include the data-bases which would be used in any of the areas outlined at the beginning of this project, ranging from Planning Department data in Local Government, to constructional details of a building represented in machine-readable form. The second areas of application arise from the real time display of moving objects which the hardware developments make possible, such as Pilot Training Simulators for new aircraft. When this is linked with particular properties of boolean expressions it is possible to set up control systems for Numerically-Controlled Machine Tools so that it becomes impossible for the machine to accidentally cut away parts of the object which it is being used to create.

SECTION I

THE CONTEXT

CHAPTER I

THE INFORMATION SYSTEM CONCEPT

THE INFORMATION SYSTEM CONCEPT

In this section some of the general issues are reviewed which govern the form in which computer-based facilities can be and may be implemented. When considering the replacement of an existing technique by an automated technique it is not possible to evaluate the merit of the change without first of all considering the environment in which the change is to take place. Since not all tasks can be automated, and for that matter it may not be desirable that they should, one aspect of this analysis must be to relate the computer-based methods with existing procedures. Various points of view are adopted, on one hand the issues which face an individual organisation wishing to use automation are examined, on the other the long term global implications of computer use are estimated. Each angle provides some insight into the possibilities and the limitations of machine-based techniques, the aim being to provide a context in which the work of later sections can be fitted.

The nature of the problem depends very much on the scale of computer operation envisaged, and how the machine is to be used. Computing is a capitally intensive process. Where an agency is setting up its own machine system, many of the economic advantages of automation can often only be achieved by the continuous use of the computer. Where the needs of an organisation can be met by a small machine then this may well be a feasible proposition. There are, however, many applications which do not require much time but still demand the facilities only found in large computing centres. In such cases computer time may be best purchased from a service bureau (William F. Sharpe, 1969).

THE INFORMATION SYSTEM CONCEPT

Once automation is accepted in one aspect of an organisation's activities, a chain reaction may result. The initial automation of one task may have unexpected effects which demand a major reorganisation of other tasks. Alternatively, in setting up a system to achieve one objective it may be realised that for every little extra expense or effort in modifications, the system can be made to carry out many more tasks. Allowing one change to lead to another without some form of planning cannot guarantee a result which is either the most effective or the most efficient from a cost point of view.

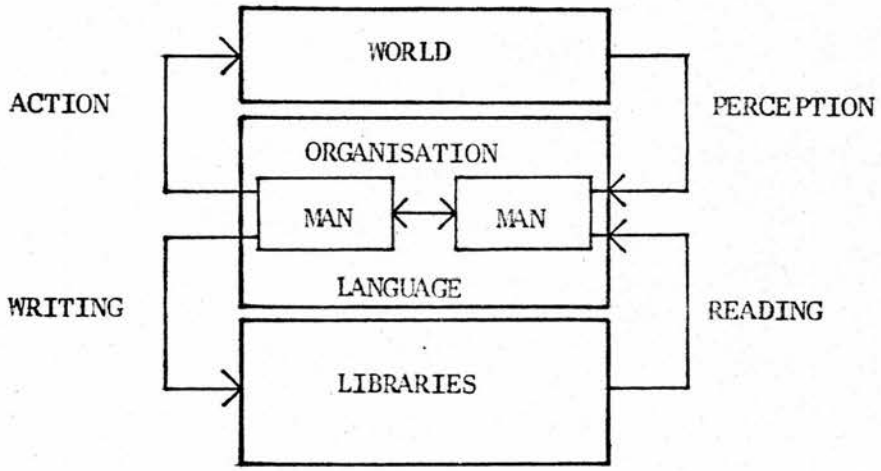
The Physical Components

The problems encountered when setting up large scale data processing systems for business firms, government agencies, and defence and military establishments, led the operations research workers to define the information system. This term was used to cover the collection of personnel, machinery, procedures, collections of data in libraries and filing systems, and the organisational structure, which had to be analysed before such a major undertaking could be attempted (P. E. Rosove, 1968).

Defined in this way, however, the information system is not a very useful concept. Giving a name to such a diverse collection of parts does not help with the problem of relating them into a working whole. The important unifying idea which makes it possible to bring these different elements together is the concept of information flow. If the flows of information between components of an organisation are represented as a network diagram, then it becomes possible to analyse the impact of using new techniques. Machines, individual people, and

organisations can form the nodes in such an information flow network.

The interpretation of information flow in such a scheme is however more complicated than it would first appear. Since its function is to provide a way in which the actions of people and machines can be expressed in a single system, it is perhaps not surprising that its natural meaning has to be stretched a little. Figure 1 provides the basic classification of the components of an information system. The arrows represent the linkages which are being regarded as flows of information.



Basic Information System

Figure 1

Information is normally taken to mean the knowledge which a person obtains either through perception, from conversation or from reading books or drawings. As a result, a flow of information, strictly speaking, can only occur directly when two people talk to each other, and indirectly when someone reads a book. This appears to make the two arrows labelled perception and action, though related to the creation of information, not an information flow in a conventional sense.

The difference between these two arrows and the others is the level of coupling they represent. Perception provides a person with the basic knowledge about his environment. Perception and physical action are the simplest feedback and control links between an individual and his surroundings. At its basic level perception depends on sensitivity to light, heat, sound, physical contact and various chemical effects - the same form of linkage which occurs between the components of a physical machine. Perception itself, involves more than sensitivity. It includes the ability to discern the coherent patterns in what is sensed. It is this composition of sensory inputs into something of a more unified nature which creates a person's knowledge of his environment (J. Piaget, 1971).

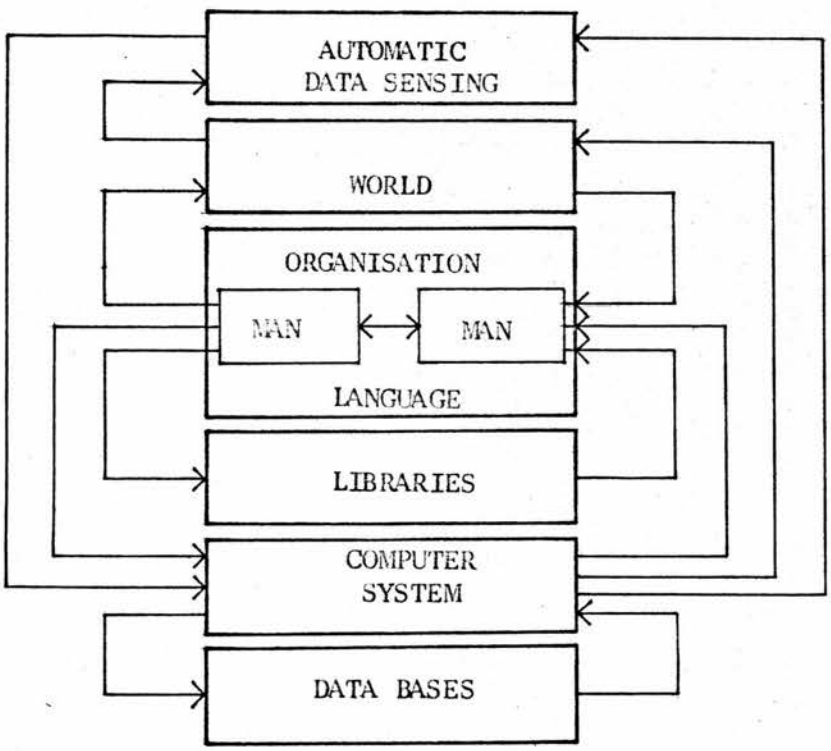
Information and Data

To overcome this problem, information flow is interpreted as a movement or change in data which can potentially be interpreted by a human observer in a meaningful way. This makes data any physical signs or signals which man can interpret as information. 'Data' as a term is much easier to work with than 'information', since it avoids the subjective component of understanding, and meaning. In this context 'action' is regarded as the way that information is created by re-structuring data.

In Information theory the problem of meaning is side-stepped by defining information in terms of the ability of codes, or data-structures to represent or distinguish different messages (C. E. Shannon and W. Weaver, 1949). A particular message contains more information if it rules out twenty alternatives, than if it only rules out ten. If a particular

data-structure can only represent ten different messages then this is the limit to the information which it can carry. The ability of a data-structure to represent twenty different states or distinguishable patterns, does not mean that they are all going to be used to represent different messages. Consequently, the way that meaning is attributed to data cannot be completely avoided by this approach. In the present context, where people are important components of an information system, it is useful to retain the word information in its natural sense to include the property of meaning, and to use the term information content to cover the concept of information in information theory.

Given this extended concept of information flow, the basic classificational diagram can be restructured to include automatic data processing as shown in Figure 2.



Basic Computer Based Information System

Figure 2

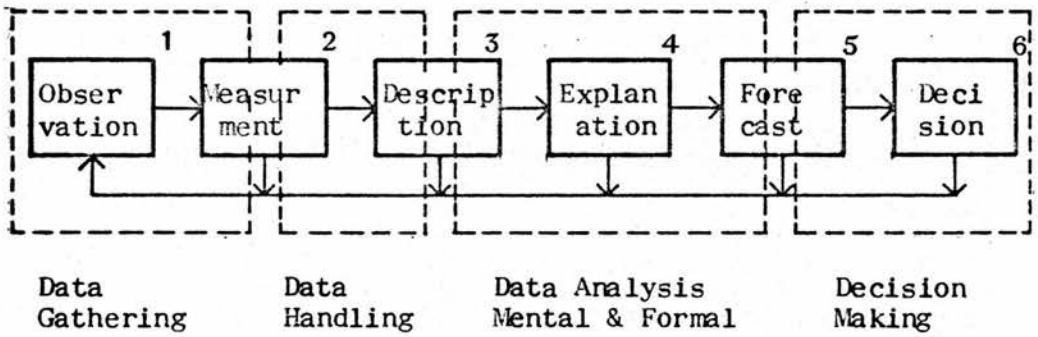
All the arrows in Figure 2 represent the movement of data or the actions which modify a data structure. The only restrictions on the form which these data can take occur where they enter the box labelled 'organisations'. The data which flow from this box to the computer, and the data which return, must be meaningful for people within the organisation. This limits these data to the language or graphic forms which already exist in man-to-man communication. In other sections of the diagram the data can exist in a variety of forms from the different states of a mechanical device to series of electrical impulses. It is only when the data are presented to man that they must be transformed into an acceptable form for one of his senses so that they can, subsequently, be given meaning.

Properties of the Total System

In many ways, Figure 2 represents the obvious. It serves its purpose as a starting point, which is to separate the principal forms of data flows discussed in later sections. As a flow diagram, it expresses one important assumption, which is that data flowing into one of the boxes are intimately related to the data flowing out. This makes it possible to define the boxes as processes of data-transformation. From this point of view the arrow labelled 'perception' in Figure 1, is strictly speaking, incorrect, since the flow is of raw sensory data, and perception is the transformation to these data which occurs within the box labelled 'man'. The diagram also serves an extra function as a dynamic model, in suggesting that a balance must exist between the flows of data, depending on the speed of the various processes, because a bottleneck would bring the whole system to a halt.

Except where ideas such as the equilibrium between data flows, or the laws of conservation such as the conservation of energy and matter are

under discussion, it is convenient to miss out the box labelled 'world', since for most purposes, though it formally closes the system, it contributes little else. By modifying the classification of components, in other words adopting different data transformation processes, it is possible to construct information flow systems suitable for describing a wide variety of tasks. R. F. Tomlinson (1974) adopted the relationships shown in Figure 3 as a system context to 'Data Handling' in his analysis of the applications of automatic data processing to map making.



System Context to 'Data Handling'

Figure 3

Activities as Network Components

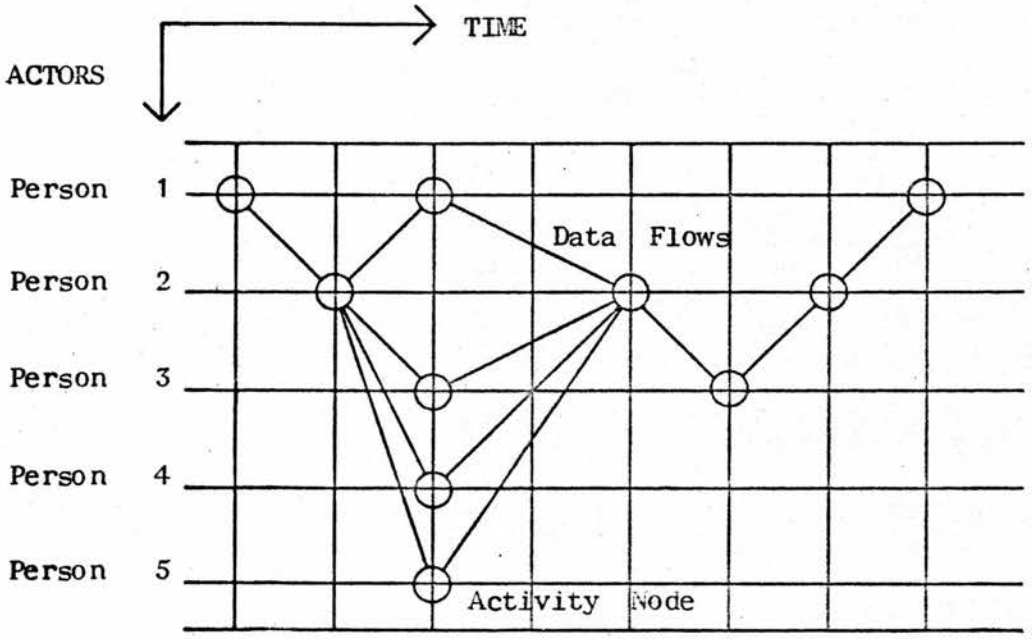
The classification of components in Figure 3 are compatible with those in Figure 2 in the sense that both networks can be used to describe the same overall operation. The 'observations' in box 1 of Figure 3 can, however, correspond to the observation or perception of a single man, the collective observations of many people, or the collection of data by remote sensing or measuring devices. In other words, the processes defined in Figure 3 can be implemented in a variety of ways by the components defined in Figure 2. Though the design brief for an information system can be given in the form of general components, the designer has to realise the system in terms of physical components.

This means that when the system designer is faced with an existing organisation, he has to describe the behaviour of its physical components in general terms before he can then return to consider new physical components and new arrangements of these components which will perform the same overall operations. The problem with the first stage of this analysis is that, in the words of W. R. Ashby (1964) "every material object contains no less than an infinity of variables, and therefore possible systems". Only a few of these system descriptions will be useful.

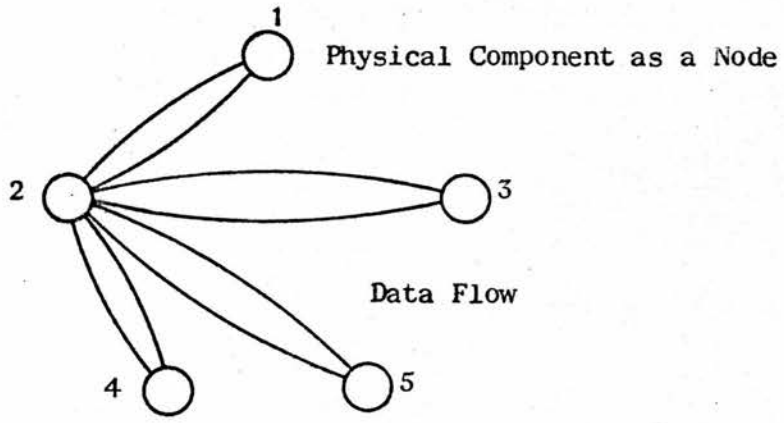
Just as Figure 2 is the collapsed version of the flows of data between the physical components of an organisation, so Figure 3 is a collapsed version of a precedence network of the activities which make up a more complex procedure. This form of activity network makes it possible to select the important data or information transfers in an organisation. It is clear that many more forms of communication between the people in an office take place than those essential to the office's primary activities. If a precedence network for each of an office's major activities is set up as shown in Figure 4, then by plotting a graph with the office personnel as nodes and using only the communications between activities taken from the precedence network, as links, the unnecessary forms of communication will be removed.

Both forms of network are useful. The precedence network can be restructured to show the result of using new techniques. The network showing the communication between the physical components of the organisation can be used to schedule the timing of all the office projects in a way that makes the optimal use of all the available facilities. There are a wide variety of network optimising procedures which can be

used once the appropriate network has been formulated but the original choice of components is important. In the case in hand, for example, it is no use defining a process which depends on manual techniques in a way that requires the human ability to read and interpret drawings if the resulting network is to be used as the basis for developing a computer-based system.



Precedence Network



Components Network

Figure 4

The process represented by a node in the precedence network is defined by the relationship between the input data and the output data. Where this transformation is suitably defined it can be carried out by machine.

A process which has been analysed in this way into a collection of transformations coupled together in a network can often be redesigned by starting with a different collection of basic transformations linked together by a different network. As long as the relation between the original input data and output data is maintained the overall process is regarded as the same. Though the abstract process is the same, however, and the component as a box in the precedence network will have the same name, the component which it relates to in the network of physical components may well be different. In such a redesign exercise, it is often the change in the collection of physical components which imposes the change in the available abstract transformations, which the designer is free to use.

Conclusions

Though the first objective for the designer is to produce a working system of transformations, the properties of the final physical system are what determine a system's viability. Each abstract operation is carried out by a physical component which has size, cost, and uses energy. The whole arrangement is organised in space, and the movement of data from one node to another takes energy and costs money. Given a basic mechanism, the physical design-implementation must produce the best result from these other characteristics. A variety of operation research techniques has been developed for this purpose, ranging from rigorous mathematical optimising techniques to less rigorous simulation procedures (H. M. Wagner, 1972). In this chapter the basic approach to the technical solution has been briefly discussed, in the next chapter the practical choice between alternative technical solutions is considered in a little more detail.

CHAPTER 2

CHOICE OF INFORMATION SYSTEM

CHOICE OF INFORMATION SYSTEM

In this chapter some of the issues are examined which will affect the choice made among alternative 'technical' solutions. The natural starting point for this investigation appeared to be given by considering the relative costs of the different possible alternatives. Assuming that the cheapest solution should be considered the best, other things being equal, the problem becomes that of deciding what is meant by 'other things being equal', and when they are not, finding some criterion other than cost on which to base a choice.

It seems a reasonable claim, to say that the way in which computer-based activities are introduced into an organisation, depends on some direct economic advantage, or an indirect advantage such as being able to perform tasks which could not previously be undertaken. Where a change means the substitution of one cheaper process by another, then the choice is simple. Where there are side effects resulting from a new technique, then the size of the system which has to be analysed to determine the real costs must be greater. It can be seen that this kind of evaluation raises issues which are similar to those associated with overspill diseconomies such as pollution. The selection of the processes which are included within the cost analysis becomes important and may well affect the final outcome. Similarly, the time intervals used in costing may also affect the result, for example where a process has a feedback effect which reduces the costs of its inputs. The simplified economic model which is used as a framework for this discussion can be constructed in the following way (W. F. Sharpe, 1969).

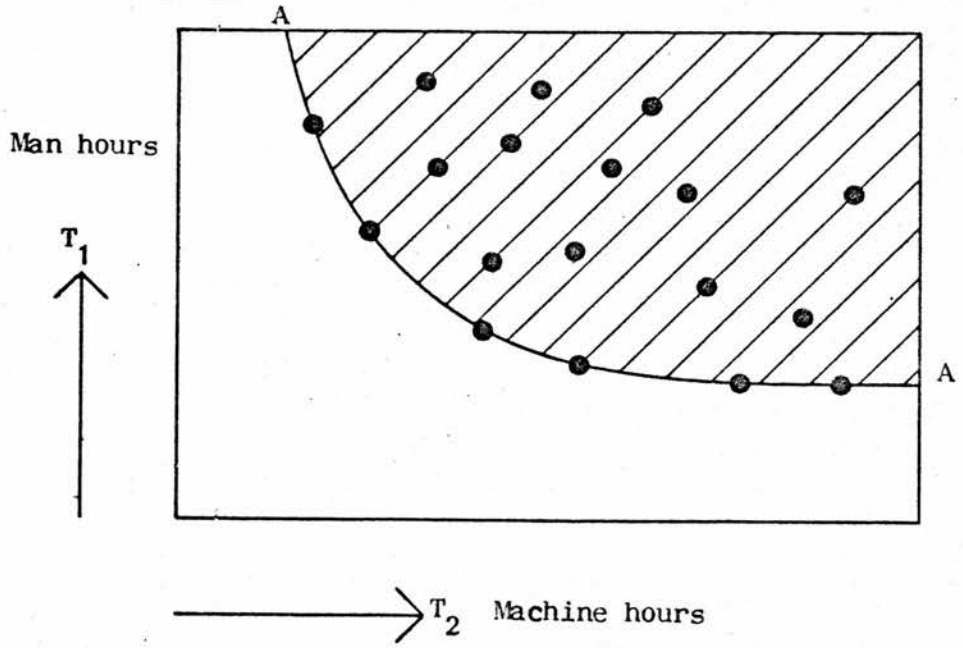
CHOICE OF INFORMATION SYSTEM

Cost of Alternative Schemes

Assume for a particular operation that there are many alternative ways of carrying it out, giving different levels of production and using different combinations of machine and manual labour. If the cost of the units of manual labour is P_1 per hour, and the cost of machine time is P_2 per hour: then if T_1 is the number of man hours and T_2 is the number of machine hours necessary to complete the operation for a level of output O, the cost of output C will be given by:

$$C = P_1 \cdot T_1 + P_2 \cdot T_2$$

This reduction of the different processes to machine hours and man hours, is an extension of the way in which the precedence or activity network was collapsed down to give the physical components network. By plotting the results for each of the alternative ways of carrying out the process in a graph with T_1 and T_2 as coordinates, the different alternatives become directly comparable.



The Technologically Efficient Combinations of Man and Machine Labour for Different Methods of Production but Producing the Same Level of Output Oa.

Figure 1

CHOICE OF INFORMATION SYSTEM

The alternative ways of carrying out this process represented by the points along the line AA in Figure 1, are referred to as the technologically efficient ways of combining man and machine labour for this level of output Oa. The shape of this curve is idealised in this diagram, based on the principle of diminishing marginal returns from continuously increasing one factor of production. This is reflected in the extreme cases at the two ends of the line AA by small changes in the lesser factor considerably reducing the need for the predominant factor. In practice, there is no reason to suppose that this curve would be so well behaved.

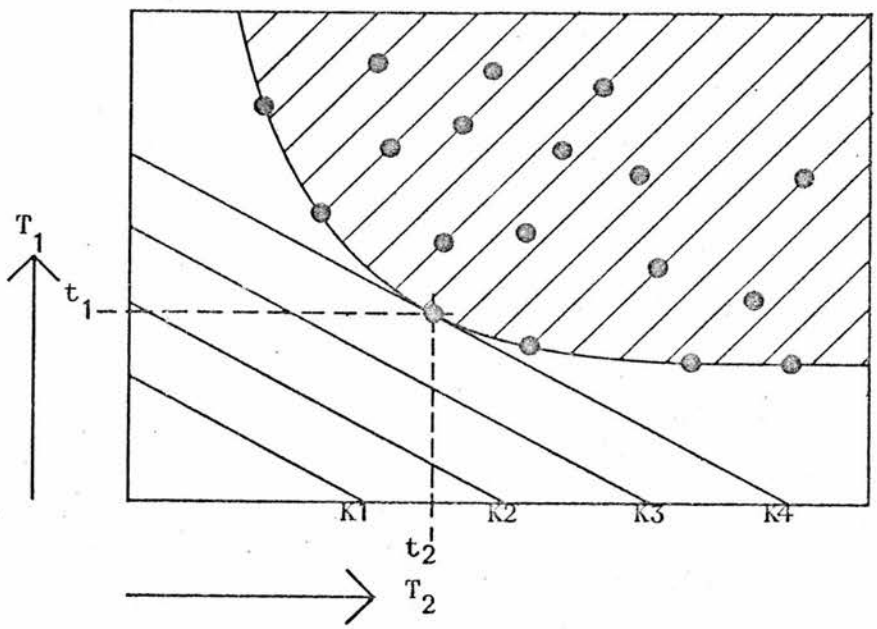
The points above this curve in the shaded zone of Figure 1 represent less efficient ways of carrying out the process, since there is always an alternative combination to the left of them or below them which has less of one of the two inputs.

From the collection of technologically efficient operations the one with the minimum cost of production for output Oa will depend on the relative costs of man hours to machine time. Where these costs are P_1 and P_2 respectively, the minimum cost of production can be found by the following geometrical construction. In the diagram shown in Figure 1 construct the series of parallel isocost line given by the relationship:

$$K = P_1 \cdot T_1 + P_2 \cdot T_2$$

where K is given constant values starting with 0 and is increased until one of the isocost lines passes through a point on the line AA. This point will represent the combination of man hours and machine hours which are needed to produce the output Oa for minimum overall costs. It will

thereby indicate the economically efficient form for the operation when the prices of man and machine time are P_1 and P_2.

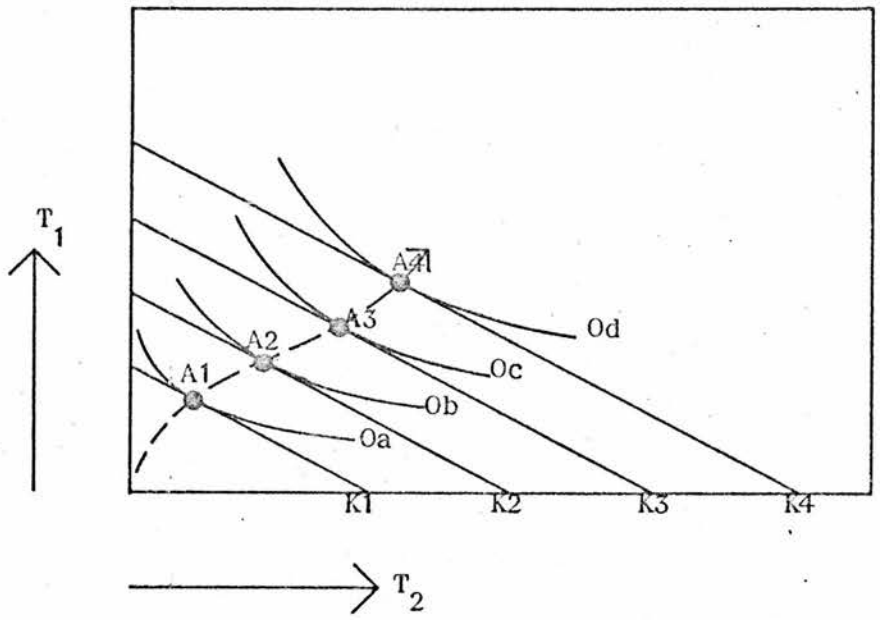


Output: Oa; Prices: P_1, P_2; Minimum Cost: K3; Times: t_1, t_2.
The Economically Efficient Operation for Given Prices.
Figure 2

It can be seen from Figure 2 that if the cost of machine time should go down in relation to the cost of man hours then the gradient of the isocost lines will change and favour the selection of a technologically efficient operation which uses more machine time. If the cost of machine time should rise relative to the cost of man hours then the reverse would take place.

By extending this form of analysis it is possible to express the changes in the form of operation which gives the most economic results at different levels of output. If the technological efficiency curves are constructed for different levels of output, with their tangent isocost lines for particular price levels, the result will be a diagram of the form shown in Figure 3. If the different levels of output are Oa, Ob,

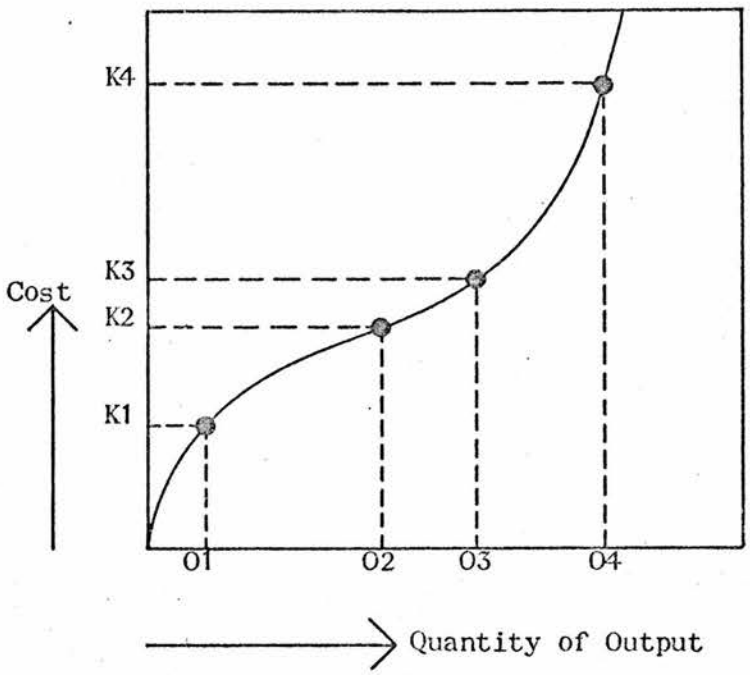
Oc, Od then the corresponding overall costs are K1, K2, K3, K4, and the structure of the different sized organisations needed to carry out the operation will be represented by the points A1, A2, A3, A4.



Economically Efficient Organisations for Different Outputs

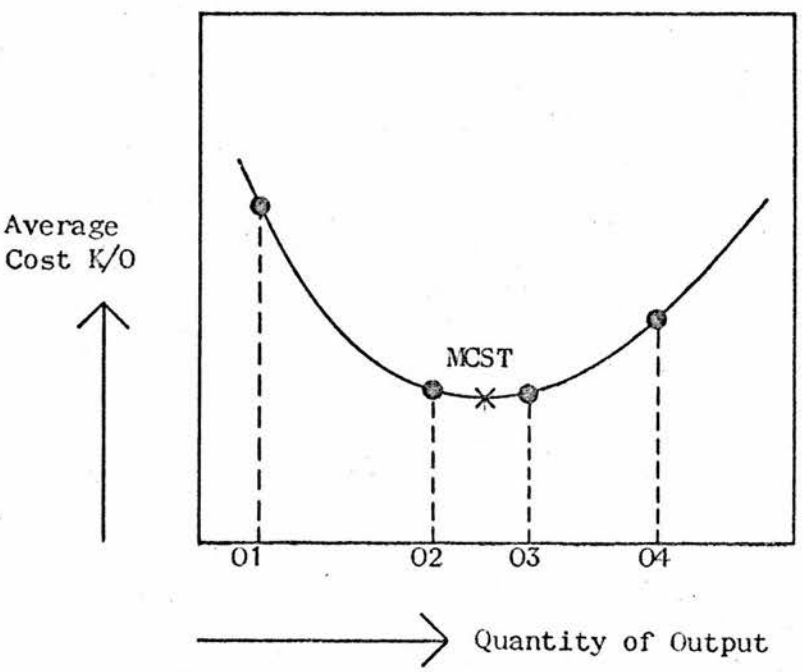
Figure 3

If the values of K and O in Figure 3 are plotted against each other in a graph showing Cost against Quantity, the result will generally be of the form shown in Figure 4. This relationship between a measure of the output against a measure of the inputs, assuming only technologically efficient processes are employed, is called an organisation's production function. If the information in the graph of Figure 4 is expressed in terms of average costs per unit of output plotted against the total quantity of output, the result will be that shown in Figure 5. At some level of production given certain technological possibilities there will be a minimum cost per unit of output, in Figure 5 shown by MCST.



The Production Function

Figure 4



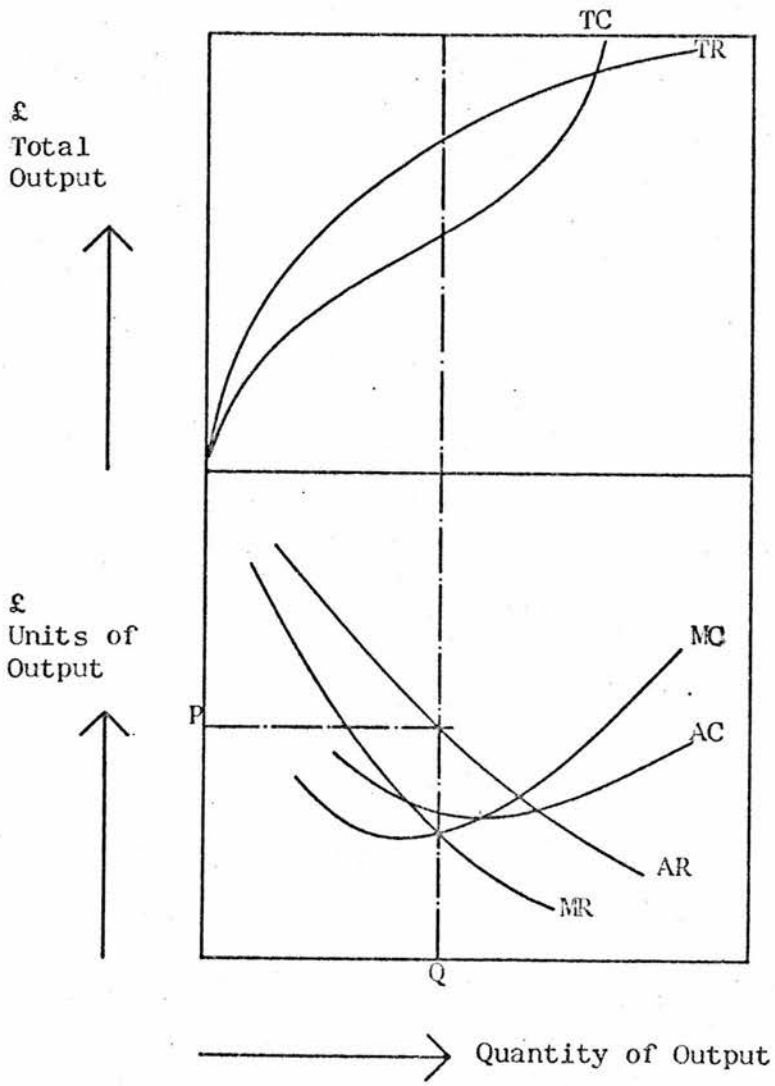
Average Cost Curve for Different Levels of Output

Figure 5

CHOICE OF INFORMATION SYSTEM

Demand and the Effect of Choice

Though at first sight the size of organisation which can produce a unit of output at a minimum average cost would appear to be an optimal size, if this level of output is greater than the total demand for the product there is little point in expanding to produce it. Depending on demand conditions, an organisation can maximise its profits whether average costs are increasing at a minimum, or decreasing. Where costs are decreasing, if the relationship between demand and supply is in the form shown in Figure 6, then an increase in output would reduce profits.

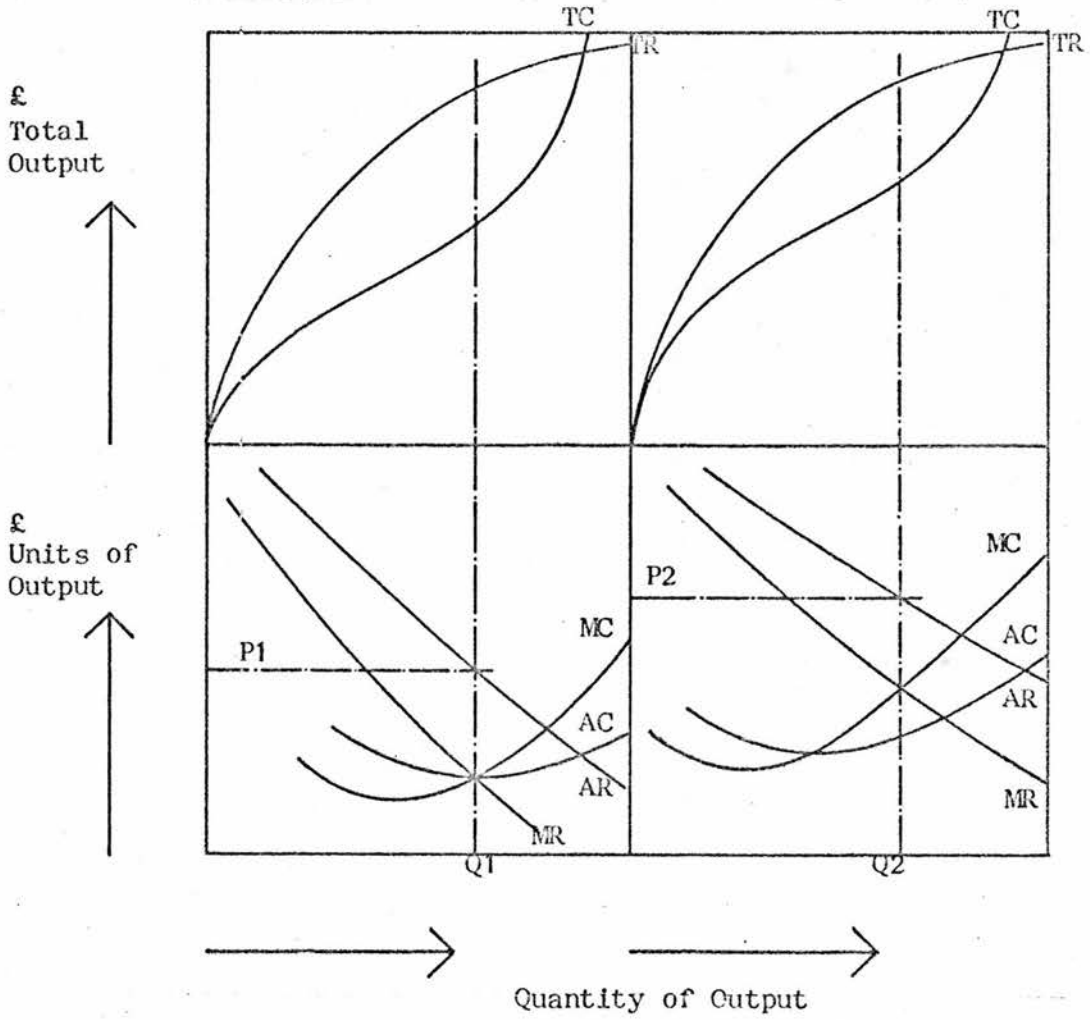


Maximum Profit with Decreasing Average Costs

Figure 6

CHOICE OF INFORMATION SYSTEM

In Figure 6, the relationship between supply and demand at different prices for the product, is indicated in the relationship between the TC, or total cost curve and TR, or the total revenue curve. The maximum profit will occur where the TC curve stops moving away from the TR curve; in the diagram this occurs at a level of output Q. At this point the gradient of these two curves will be the same. This is shown where the MC, or marginal cost curve cuts the MR, or marginal revenue curve. The relation between TC and TR in Figure 6 establishes this point below the level of production needed to give minimum average costs. In Figure 7 the two alternative relationships between TC and TR are illustrated.



Optimum Sizes of Organisations

Figure 7

CHOICE OF INFORMATION SYSTEM

The Problems of Optimal Choice

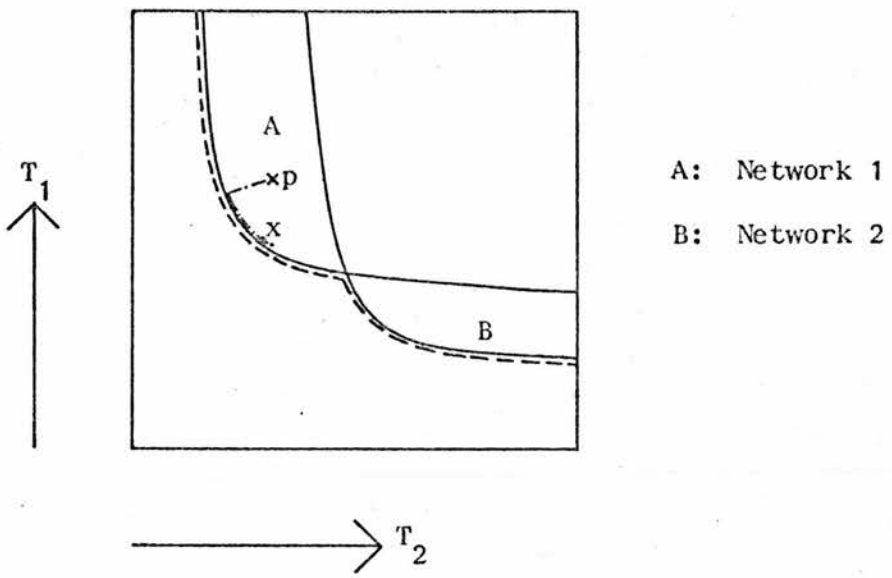
In the case of well established products, where the alternative ways of producing them have been thoroughly explored, and where consumers are well adjusted to the availability of the product, and demand behaviour is well known, then possibly enough information will exist to structure a model along these lines. Where new products are concerned this will not be the case, and even with established products a new factor such as automation may outdate existing information.

If established products can be produced using new techniques, the problem becomes that of finding the cheapest way to organise a new production process. It is in this area that operational research techniques provide powerful analytical tools. The advantage of starting with a network model is that it can be used in a variety of theoretical optimisation procedures related to linear programming. H. M. Wagner (1972) states:

> "The key justification (for the use of networks) is that the mathematical characteristics of network models are so special that by exploiting these structural properties, you can obtain major efficiencies in finding optimal solutions".

The mathematics of these processes lie outside the scope of the present study, but there seem to be several reservations which must be made about the use of these forms of optimisation procedure in their practical application. To start with, where a particular process is described by one network, an alternative process may require a different network description. Assuming that each process can be optimised using linear programming techniques, then each process will give a convex space in

which possible solutions may be found. The technologically efficient organisations, as defined in Figure 1 could then lie on the surface of the union of two zones as shown in Figure 8. It could be argued that if network 1 and network 2 are links between the same physical components, then the problem could be redefined so that the original network is a complete graph of all the possible links between the components. Though this makes it conceivable that some continuous change can be expressed moving from the technical solution of network 1 to the solution of network 2, it does not rule out the possibility that the eventual surface of the technologically efficient organisations will be multi-peaked.



Two Networks Used to Define Optimal Organisation Structures
Figure 8

If an organisation has chosen method of production A then internal changes to this organisation will be more likely than an attempt to change to the method of production B. In other words, in practice there will be a commitment to one or other forms of production, based on the cost of changing method. Within this framework an organisation will

attempt to improve its performance by trial and error. If the original structure of the organisation is represented by the point 'p' in the zone A of Figure 8 , then by improving efficiency and moving towards the boundary of A the costs of production will be reduced. Once on the boundary it will then be possible by incremental adjustments to follow the boundary to the point 'x' which represents the most efficient balance between the costs of the various inputs.

Where the surface of the solution space is not convex it is not possible to use the standard linear programming techniques. In his Ph.D thesis, 'Random Methods for Non Convex Programming', Peter Rogers (1966) starts from the idea that an existing design solution is a sample from such a solution space, and investigates the statistical problems of determining the shape of its surface. The problem with a single sample solution is that though it can be incrementally improved, which is equivalent to climbing a hill in the 'response surface', the top of this hill may only represent a local optimal configuration, there being higher peaks elsewhere on the surface. By taking sufficient design solutions as starting points the probability of finding the true optimal solution improves. One of the problems which Peter Rogers investigated was that of finding a sampling strategy which could give a reasonable indication of the surface's shape when dealing with many variables.

Where the problem is one of physical planning or design, such as the layout of reservoirs, and the demand for the product - water - is general, and its dimensions well known, and the demand is inelastic, then optimum results may well result from minimising costs. Generally speaking, reservations must be made when taking this approach, and even

in this case there are complications. It is rare that a physical product only serves one purpose. Reservoirs in water resource studies such as those carried out by Peter Rogers are primarily concerned with the supply of water. There are alternative functions which can be carried out by reservoirs. In certain situations they can provide hydro-electric power, in others they can make a valuable contribution to recreational facilities, and even more difficult to value they can contribute to the scenic quality of tourist areas. So when different design schemes are proposed it is not always a matter of choosing the cheapest solution. More expensive results may in reality be different or modified articles, which to the design-client or consumer represent better value for money.

Information about Optimal Products

The aim to minimise costs is only one part of the overall problem. It is unfortunately that aspect of the problem for which information exists prior to a design exercise being undertaken, and therefore tends to provide an easy rationalisation to the 'non-involved' decision maker. The choice of the consumers, reflected in the demand for a product, can be expressed in the context of the information system concept as the matching of products to their potential uses in other processes from a larger system. It is thus possible to argue that if the larger system were to be investigated the consumers' choice would belong within the technical decisions required to set up a working network, rather than a purely optimising decision which choice based on minimum cost would be. However, to be able to take a design approach to the larger system, it would be necessary to specify all the uses which would be made of the

product in question, since in the larger context it must be regarded as an intermediate product. It is clear that there is a practical limit to the scale of operation to which a design approach can be taken, primarily because the relevant information cannot be obtained.

R. A. Jenner (1966), in discussing 'optimal product determination' argues that the principal way in which business firms obtain information about the use of their product is by its success or failure in the market. A firm developing new processes and new products, is on one hand giving consumers the choice to define the type of product they want, and by the success or failure of the product providing themselves with otherwise non-existent information about such choices. Since success is measured by profit, the profit motive sustains a self-regulating, or self-designing aspect of the economic system which is beyond the scope of detailed administrative or design control.

With the use of a Markov model to formally represent the transitions of firms from one production process to another and from one product to another, Jenner demonstrates that if a superior product exists, then a steady state will eventually be reached where all firms move into its production. The assumption on which this model is based, is that a firm which is producing a successful product, is less likely to look for alternative products than a less successful firm. As a steady state is reached, the profits which are in effect what define the successful product, will be absorbed by new firms moving into its production. This results in a flattening of the demand curves for this product for firms already producing it and a consequent reduction of profits to a minimum. This sets into motion the process of searching for new superior products

all over again.

Figures 6 and 7 suggest that once a product has become well defined, in other words it can be assumed that each firm is producing the same article, that an optimal size exists for the production unit. For the three different market conditions these are Q, Q1 and Q2. It would appear, however, from Jenner's analysis that transition should be considered to be a more common state than that of stable products. Even when a basic product is well established pseudo-states of transition can be created by style and fashion changes. Because this dynamic model expresses as the motive force behind the behaviour which it describes the primary aims in developing and applying computer techniques - to improve the efficiency and productivity of existing processes - it provides a reasonable approach to the development and introduction of computer techniques to existing organisations.

Conclusions

The use of automatic data processing can be considered at two levels. The first where the scale of its application is amenable to design control, the second where the aggregate effect of its general use creates changes which are beyond this level of control. The speed at which the overall changes occur will determine to a great extent whether these two levels can be considered independently. In simple terms, the goal must be for a system to pay back the investment required to implement it before it is outdated. Where overall changes are occurring relatively quickly, there appear to be two related strategies by which this can be achieved. The system can be small, so that the time periods necessary for financing

it can be short, or not very different in practice, the system can be modular. A modular system consists of separate units acting within a more general purpose framework which by its nature will have a longer lifespan. It is then possible for single modules to become outdated and be replaced, while the probability that the whole system is rendered obsolete is considerably reduced.

It is clear that defining modules, defining products and choosing a classification of design components, can all be regarded as similar processes. New technology may affect the optimal grouping of components within existing organisations and systems. The way that the necessary readjustments are achieved, to provide a new optimal arrangement, has been discussed where they are under direct design control. It has also been suggested that such design solutions are subject to natural selection within a larger context, by the action of self-regulating forces within the economic system.

Automatic data processing already provides many profitable applications at the level of single organisations, but it is evident that further major changes can be expected at the level of the whole community. Based on equilibrium concepts it is reasonable to expect these changes to result in some form of overall advantage. Jenner's analysis shows that at the aggregate level the route which developments will take are based on the interaction of local decisions of choice at the level of the individual organisation or person. The behaviour of such a dynamic system gives no <u>a priori</u> reason to believe that potential, overall benefits will be achieved. It seems quite possible that the overall

behaviour of such a system of incremental changes could be cyclic or even regressive.

In the next chapter, the processes of product definition as they have taken place in the computer industry so far are discussed. This development started with simple computing machinery and, from this initial product, many other different products which depend on machine use have evolved.

CHAPTER 3

PRODUCT DEFINITION IN COMPUTER-BASED SYSTEMS

PRODUCT DEFINITION IN COMPUTER-BASED SYSTEMS

In Chapter 1 the general problem of merging computer-based facilities with the existing processes in an organisation was discussed. Since there are likely to be more ways than one in which such an operation can be carried out, the second chapter examined the factors which a system designer has to consider when making a choice between the alternatives. The general ideas presented in these two chapters have been taken as the backcloth to the work presented in the following chapters. They provide the context within which it is possible to discuss the efficient design or the evolution of effective computer-based information systems.

Given the initial intention of this study to transfer information held in drawings into machine data which was outlined in the Introduction, the first stage of analysis was to consider the components which were available for implementing such objectives. This posed a problem. Since the first computing machines emerged towards the end of the 1940's a complex hierarchy of computing facilities has developed, and a decision had to be made on which level in this hierarchy would form the basis for this study.

The difficulty is that basic computer technology is still changing rapidly. Structuring principles which would have been applicable to system design even five years ago have been superseded. Advances have provided greater freedom in design and future advances promise to do the same. For this reason, in this chapter the broad classification of components which is employed in later sections, is discussed from an historical point of view. This approach gives some basis for judging the forms of component which can be regarded as stable, and those which

are more likely to be transient. The particular economic and technical bootstrapping process from which the forms of existing facilities arose, probably no longer apply to the kind of facilities which it is now possible to create, and the kind of facilities which it would be desirable to have. However, future stages in development will be governed by the same general rules which have controlled previous advances: developments will have to be economically viable to their current situation to allow them to emerge from research into practical applications.

The idea of automatic calculation can be traced back to the invention of mechanical adding and multiplying machines. The first developments in what is now regarded as automatic data processing: in other words machines which can be programmed were carried out by Babbage at the beginning of the nineteenth century. It is the basic idea which underlies automatic calculation which can be regarded as the most permanent aspect of computer technology, and therefore provides a natural starting point for the discussion in this chapter.

The word computer is applied to a variety of machines. These machines can be divided into several classes depending on their principle of operation. The principal division is between those machines which use digital data and those machines which operate on analogue data. A second classification cuts across this grouping, and divides those machines which process data sequentially, in other words where the operations are ordered in time, from those machines which process data in parallel, for example, as independent operations in space. The main stream of development has been in the production and use of

sequential digital machines, and it is these machines which are of primary interest in this chapter.

The Schematic Machine

Adopting the approach outlined in Chapter 1 means starting with an abstract description of a computer as a network of transformations which can be performed on data. There is a variety of different forms for these abstract machines or 'automata' which have been shown to be equivalent to the general purpose computer (Hopcroft and Ullman, 1969). The one described below is the one which most closely resembles the structure of present machines.

The essential components of this machine are given in the following list:

a) A store, or memory for holding the representation of numbers constituting original data and created data which results from previous calculations.

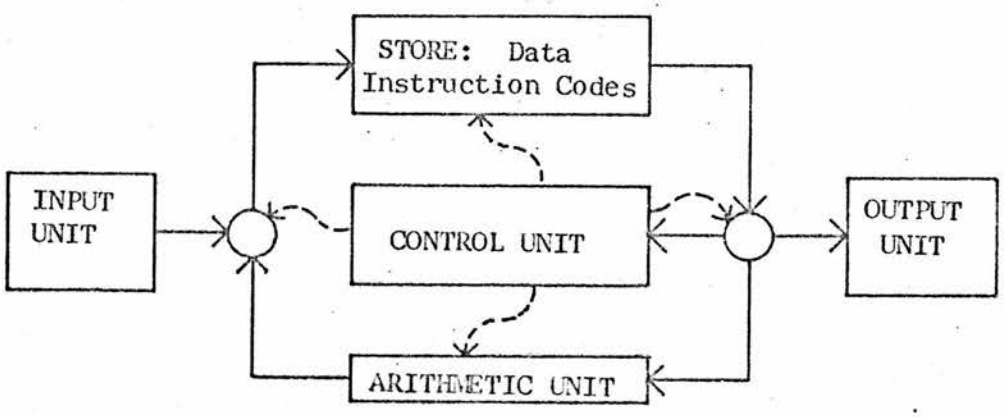
b) An arithmetic unit: a device for operating on the representation of numbers held in the store in order to carry out arithmetic operations and other basic operations.

c) A control unit: a device for controlling the sequence in which the machine performs its operation on the data in store.

d) Input devices: units which enter numbers and operation codes into the machine and create the appropriate forms of internal representation for them.

e) Output devices, similar units to the input devices except that they convert internal forms of data into external forms, such as typed numbers or alphabetic characters.

The network of linkages between these components is given in Figure 1

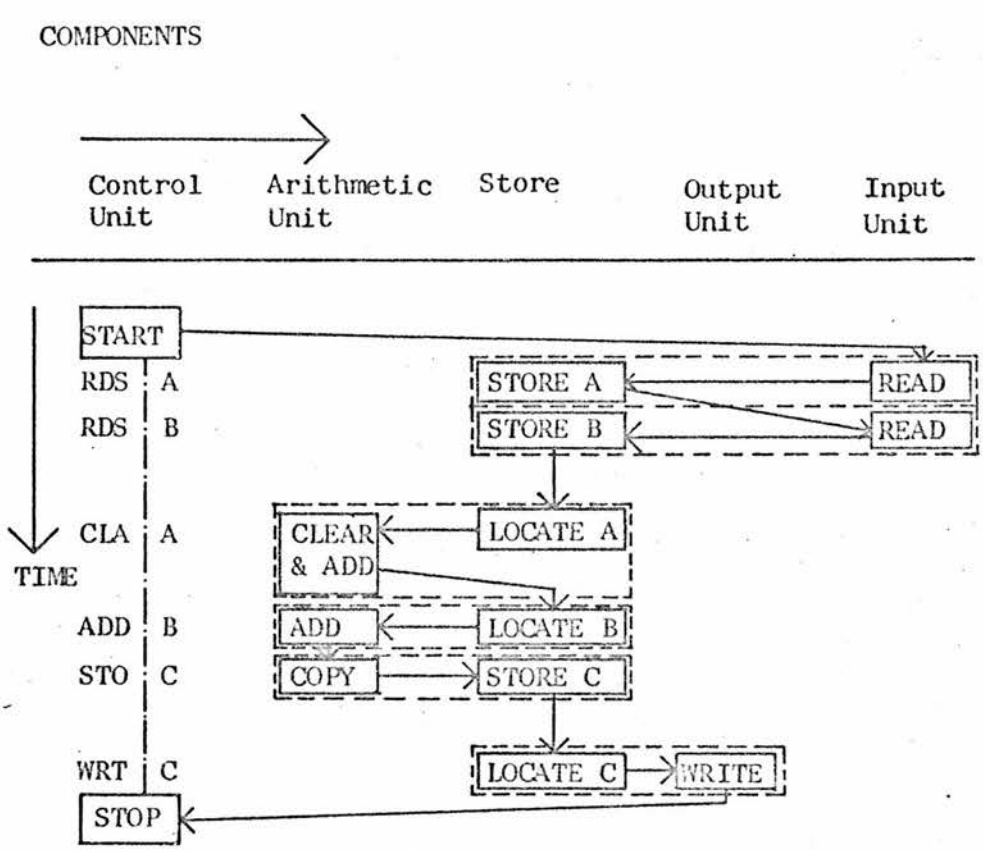


Computer, The Schematic Machine Abstract Components Network

Figure 1

The description of the way in which these components can interact to carry out calculating procedures can be expressed in an activity or precedence network. The operations involved in reading two numbers, adding them together and then printing out the result are shown in Figure 2.

The important property of this machine is that this sequence of activities can be represented by a sequence of coded instructions which can themselves be read into machine store. The principal operation of the control unit is to run through such a sequence of



Computer Program as an Activity Network

Figure 2

instructions and set into motion the actions each instruction represents in the other components of the machine. This operation converts a description of a procedure into its active implementation. In figure 2 the activities of different components, which are shown enclosed in dashed lines, correspond to five of the operation codes shown in Table 1 This collection of operation codes is sufficient for a small machine to carry out many simple arithmetic calculations. In practice, different machines have different instruction sets and it is on the basis of the available basic instructions that the form of primitive computer programs depend.

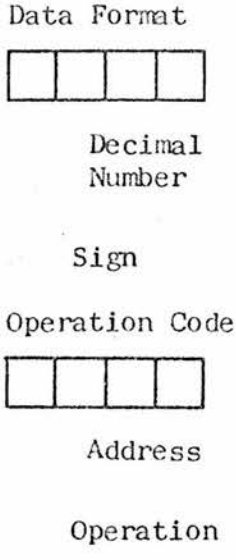
TABLE 1 Schematic Machine Instruction Set

Decimal Code	Activity Required	Components Used	Nmemonic Code
0	Clear and Add	Arithmetic Unit	CLA
1	Add	Arithmetic Unit	ADD
2	Subtract	Arithmetic Unit	SUB
3	Store	Arithmetic Unit	STO
4	Multiply	Arithmetic Unit	MLT
5	Divide	Arithmetic Unit	DIV
6	Read	Input Unit	RDS
7	Write	Output Unit	WRT
8	Jump	Control Unit	JMP
9	Jump if Negative	Control Unit	NJP

The simplest way of presenting the use of this instruction set is to describe an example using the simple schematic machine. Assume that the storage unit or memory contains 100 cells shown in Figure 3 It is necessary to identify each cell so the index number in the corner of each cell is taken as the cell's names. For this limited size of store only a two digit decimal number is needed to locate any cell. The first digit gives the cell's row and the second gives the cell's column.

Each storage cell is assumed to contain four storage elements. The first of these is the sign digit which can represent the value 0 or 1 as a code for + or -, the following three elements can each hold one of

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99



Memory Unit for the Schematic Machine

Figure 3

the decimal digits 0,1,...9. Consequently, +99 would be represented by 0099 and -99 by 1099. In storage each of these digits must be regarded as independent characters. It is only when they are transferred to the control unit, or the arithmetic unit that a relationship between them is created. It is the assumption of this relationship which is equivalent to giving the digit string a meaning. In this example, there are only two forms of interpretation which are applied to the numbers held in storage. These two interpretations are shown in Figure 3 as the Data Format, and the Operation Code.

The operation code is the 'meaning' which is given to a number when it is transferred from the store to the control unit. In the control

unit, the first digit is ignored; the second is taken as a definition of the type of operation: each digit from 0-9 corresponds to an instruction in Table 1. The third and fourth digits contain the row and column indexes of the storage cell which is associated with the operation, see Figure 3. The number 0151, when transferred to the control unit, will be interpreted into the operation: transfer the contents of cell 51 and add it to the contents of the arithmetic unit. It can be seen that if the same number in storage were transferred to the arithmetic unit it would be interpreted as +151.

The operation of the machine depends on the behaviour of the control unit, and the way it obtains instructions from storage. It is clear that since any number in store could be interpreted as an instruction, some way has to exist for the correct numbers to be selected. This selection is carried out by a counter in the control unit which automatically increments by 1 every time an instruction is carried out. Assuming the correct starting position, if a sequence of instructions is placed in storage with consecutive instructions in neighbouring cells, then the control unit will automatically run through them one at a time.

The real power and versatility of this device arise from the two jump instructions: "Jump", and "Jump if Negative". These two operations change the next instruction address in the counter of the control unit and permit the machine to return to operation codes it has already executed and repeat them, so allowing cyclic procedures to be implemented. The "Jump if Negative" command is a conditional change of sequence: if the arithmetic unit contains a negative number, then the jump

instruction address will be substituted into the instruction counter, if it is positive then the jump is ignored. The advantage of holding the instruction codes in storage is that they can be modified under program control, by treating them as arithmetic data. This makes it possible, for example, to change the address of an instruction which has to be repeated for a sequence of adjacent data cells in cyclic operations.

Once a correct sequence of operations is being carried out, particularly if there is a jump instruction at the end of it passing control back to the first statement, it is easy enough to see how the machine maintains its operation. The problem is how the machine is set going in the first place. A machine with an empty store cannot be made to do anything. The initialisation problem is of great importance since it is the process which permits the machine to be used at all.

The simplest way in which this schematic machine can be primed is to externally set up two values: 0601 and 0800 in the cells 00 and 02 respectively. If the instruction counter is set to 00, then when the machine is started it will immediately carry out a read instruction 0601, placing what it reads into cell 01. It will then automatically pass to this cell and interpret it as an instruction, after which it will move to the jump instruction 0800 which passes control back to the original read instruction, and so on. If the sequence of data statements and command statements fed to the machine primed in this way is correctly ordered, then a continuous stream of programs can be processed. To execute the program shown in Figure 2, would require the sequence of statements given in Table 2.

TABLE 2 Program to Add Two Numbers

Instruction Sequence	Nmemonic Code	Decimal Code
01	RDS 50	0650
02	+ 54	0054
03	RDS 51	0651
04	+ 55	0055
05	RDS 52	0652
06	CLA 50	0050
07	RDS 53	0653
08	ADD 51	0151
09	RDS 54	0654
10	STO 70	0370
11	RDS 55	0655
12	WRT 70	0770
13	RDS 56	0656
14	JMP 00	0800
15	JMP 52	0852

In Table 2 all the odd numbered instructions are read statements designed to enter the even number statements into a section of storage away from the space containing the simple input program, and as far as the required operation of adding two numbers together is concerned, can be ignored. The actual program which will be used for this task will be arranged in storage in the way shown in Table 2. By the time that instruction 15 is read by the input program, this program will be

in storage. Instruction 15 is a control instruction which passes control from the input sequence to the beginning of the new program. Similarly, once this program has been executed, the instruction counter will be pointing to cell 56, and the machine will pass control back to the input program, when it executes the command JMP 00. These two jump instructions correspond to the undefined operations START and STOP in Figure 2.

TABLE 3 Program in Storage

Storage Address	Contents Nmemonic	Contents Decimal
50	+ 54	0054
51	+ 55	0055
52	CLA 50	0050
53	ADD 51	0151
54	STO 70	0370
55	WRT 70	0770
56	JMP 00	0800

Implementing the Schematic Machine: Components

This example shows the nature of the basic computer. The operations specified in this description can be carried out by hand, but there is no real indication of the way that a physical machine can be constructed to perform the same processes. The earliest attempts to create machinery which could automatically carry out predefined sequences of calculations, were made by Babbage (1791-1871). His Analytic Engine, however, was never completed because of the construction problems which

arose from using purely mechanical components. The first successful machine which could automatically carry out sequences of digital calculations was developed by Howard Aitkin in collaboration with IBM at Harvard University (Rosen, 1969). This machine made use of a combination of electromagnetic and mechanical components.

Aitkin's machine, the Automatic Sequence Controlled Calculator is interesting because it implemented arithmetic operations using the decimal representation of numbers shown in the previous example. In the schematic machine decimal numbers were used because they were more intelligible to the reader than binary or other representations of numbers. In the case of Aitkin's machine, his initial design solution would be in terms of the classification of components which existed at his time. Even though binary numbers were known to Francis Bacon as codes, it was only when attempts were made to optimise the mechanisms used to carry out the various computing operations that new products based on the use of binary numbers came into existence.

In the Automatic Sequence Controlled Calculator, numbers were represented by the angular rotation of wheels in banks or registers. Each wheel could take up one of ten different positions in a full rotation, giving the sequence of digits from 0 to 9. Each register was an accumulator. This meant that if the setting of one wheel was turned through the position 9 back to 0, it incremented the wheel to the left by one place. This interpreted the process of 'carry one', and meant that the registers could be used for the arithmetic operations of addition and subtraction. If two registers were linked together so that each turn of one accumulator's right hand wheel - subtracting one -

caused a corresponding turn in the other accumulator's right hand wheel - adding one - then, when the first accumulator reached zero, the second would contain a number which was the sum of the two original numbers. In the original calculator these registers were used to store numbers and carry out the operations of additions and subtractions but multiplication and division required a separate unit to carry them out.

Electronic Components

Though the basic ideas required for the construction of computers can be traced back as far as 1605, it was the post World War II development of electronic components which made the present machines possible. The development of computers can be divided into three stages or machine generations with perhaps the fourth on its way, almost entirely on the basis of the types of electronic components used in their construction. The detailed aspects of these electronic developments lie outside the scope of this study, except in so far as they have affected the construction and limitations of the machines they were used to build.

The first generation of machines were based on vacuum-tube electronics, which had been developed for use in radio and radar applications. The Williams storage tube was used for early memory circuits, but was developed for holding radar signals. The machines constructed using vacuum tubes were many orders of magnitude faster than the electro-mechanical computers, but were limited in size by the rate of failure of these components. Since the probability of break-down increased with the number of vacuum tubes, the acceptable length of running time between

break-downs set an effective upper limit to the overall size of the machine.

The second generation of computers corresponds to the application of transistor technology. Although the transistor was invented in 1948, it took several years of further research and development before it was of practical use to the computing industry. The early transistors did not give adequate switching speeds and it was difficult to get consistent behaviour from one transistor to another. Because of this lack of uniformity replacement of a faulty transistor was a problem.

Either careful selection and individual testing was necessary, or circuits which did not depend on accurate response from transistors had to be developed. In fact, the advances in computing machinery from the first machines such as ENIAC, had already seen major changes in circuit design because of the need to reduce the number of vacuum tubes to a minimum, to give better overall performance. These new circuits made use of germanium diodes, and succeeded in reducing the number of vacuum tubes to a level that greatly improved the reliability of the whole machine.

The break-through in transistor design came in 1954, with the development of the surface barrier transistor created by the Philco Corporation. Within four years this and associated developments made the vacuum tube obsolete for computer circuits. As soon as the transistor could replace the vacuum tube, because the basic elements were smaller and less heat was generated, it meant that the size of viable computers was much greater. Several very large machines were started in this period:

IBM's Stretch computer needed 150,000 transistors, and a little later the CDC 6600 contained over 500,000. The size of these machines meant that a new approach could be taken to the problem of mass data processing.

Most of the computers which remained on the market after 1965 were classified as third generation computers. The transition from the first generation to the second generation of machines is fairly clear-cut, corresponding to the change from vacuum tubes to transistors. The change between the second and third is less clear-cut. To some the advent of the third generation meant the use of integrated circuits, to others it reflected more the improved level of performance reached by the overall computing system. The improvement in the performance of machines in this period can be attributed to both causes. On one hand, there was an increase in speed resulting from faster components such as integrated circuits and the use of thin film memory. On the other, there was also a better organisation of the machine-software relationship.

Development of Storage Machinery

As soon as electronic devices were used to operate on data it became necessary to find a storage medium which was reliable enough, and could generate the appropriate data signals from storage fast enough to keep up with them. Linked to this optimising design-problem there was a growing demand-pressure for more storage as the potential applications for these machines became apparent. Early proposals included the use of magnetic drums and discs, both of which had been successfully implemented at Manchester University and Harvard University by 1948. This extended the volume of data which could be stored, which had been severely

limited by the use of vacuum tube 'flip-flops' and cathode ray tube storage. 'Volume of data stored' was a complementary requirement to 'the speed of data handling', so that each of these devices remained in use. This must be contrasted with the replacement of mercury delay-line storage forms by the faster electronic devices when they became available.

The major breakthrough came with the development of coincident current magnetic core memory. This was a spin off from the Whirlwind project in M.I.T.: the first attempt to obtain the speeds and design structure necessary for computers to provide the 'real time' service required for monitoring and controlling automated machine systems.

When this 'core storage' was used in conjunction with magnetic disks or drums, employing the hierarchical storage strategies which had been developed in Manchester University, a composite product emerged which satisfied the demands made on data storage facilities by a wide range of users. By using high speed core storage for immediate data access and having secondary storage space on magnetic drums, it was possible to create the impression that the high speed storage space was larger than it was in fact.

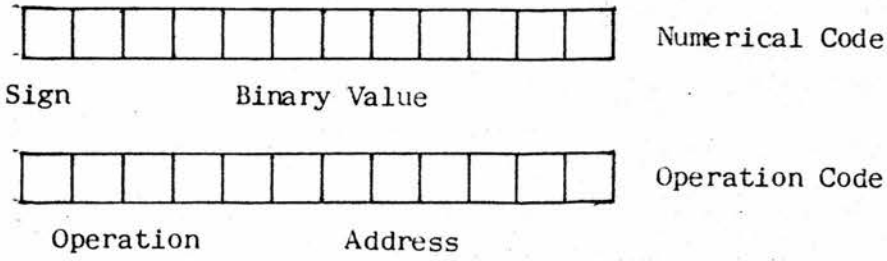
Work on the automatic transfer of data between different levels of storage started in Manchester in 1947. The magnetic drums were divided into fixed areas or blocks called pages, which matched with similar sized areas in the fast store called page frames. It was possible to keep in fast storage merely those parts of the data which were currently being used in processing, and then when data which was not in store was required, to automatically swap in the page on which they were located.

This system gave good results because most work was carried out on data located in contiguous areas of storage. As a general approach, it provided a system framework into which new devices could be introduced. As faster units came into existence they could be placed at the top of the storage hierarchy; existing components merely being used at a lower level in the overall system. Consequently, it is less likely that future development in the storage forms based on integrated circuits, or even eventually the apparently enormous potential of holographic techniques for holding large volumes of data will affect the general structure of this kind of system, though its overall behaviour will slowly be upgraded.

Developments in the Control and Processing Units

In order to describe the developments in the design of control units and data processing units in the corresponding formative period, it is necessary to extend the structure of the schematic machine to reflect the use of binary data codes. Technically, this change resulted from the use of electronic switches to give the two states 'on' and 'off' for 0 and 1, which match with the high and low values of data signals. If the previous decimal codes are to be replaced by binary codes, then a storage cell containing 12 places is necessary to contain the same range of information as the previous four place cells. For example, the number 0178 becomes 000001011010. For the operation codes, these twelve bits or places are divided into two fields of four and eight bits each. The operation code is converted into a sequence of operations by using the setting of the switches which are being used to represent the instruction code in the control register, to redirect an activating

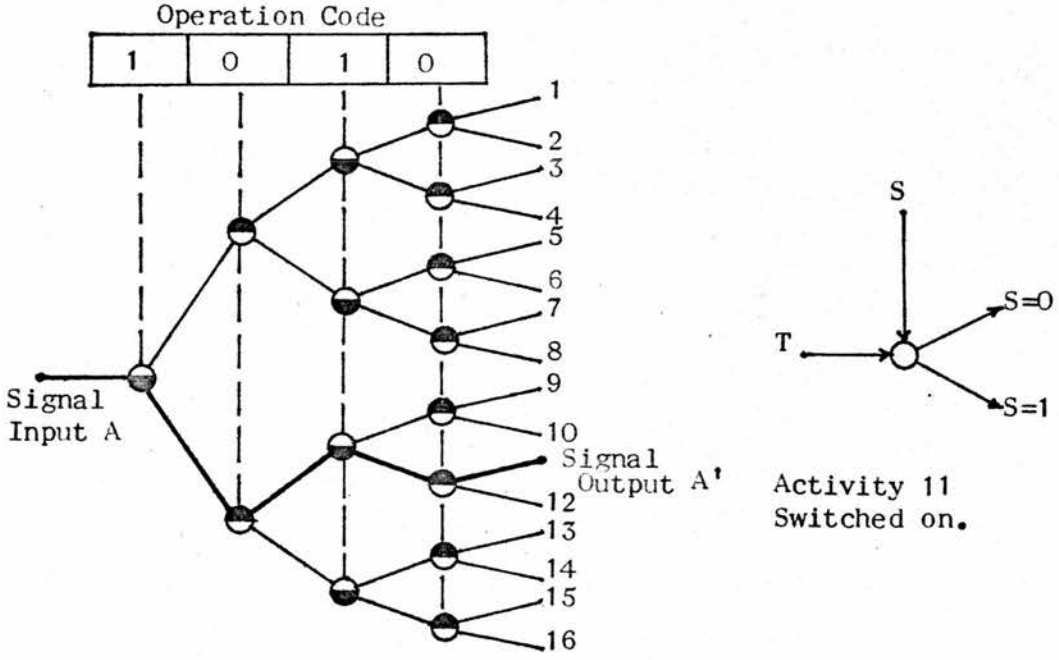
signal to the appropriate components needed to implement the instruction code's meaning. In the case of the schematic machine the first four bits selects the operation to be performed, the following eight bits select the value in core on which the operation is to be performed. In Figure 5 the use of a decoding tree shows the way in



Binary Formats for the Schematic Machine

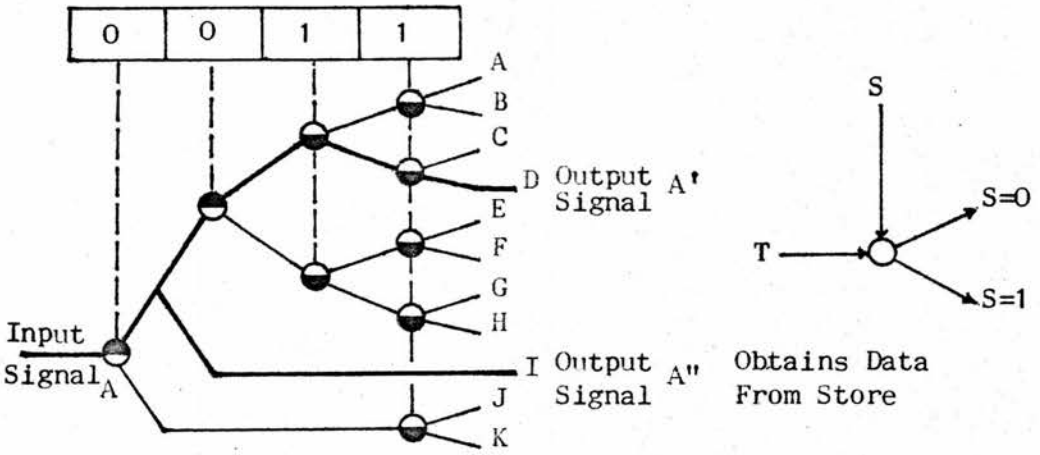
Figure 4

which four bits can be used to switch on one of 16 alternative devices. Using the ten operations given previously in Table 1, as examples, only two of these do not use the address field for accessing the main



Decoding Tree

Figure 5



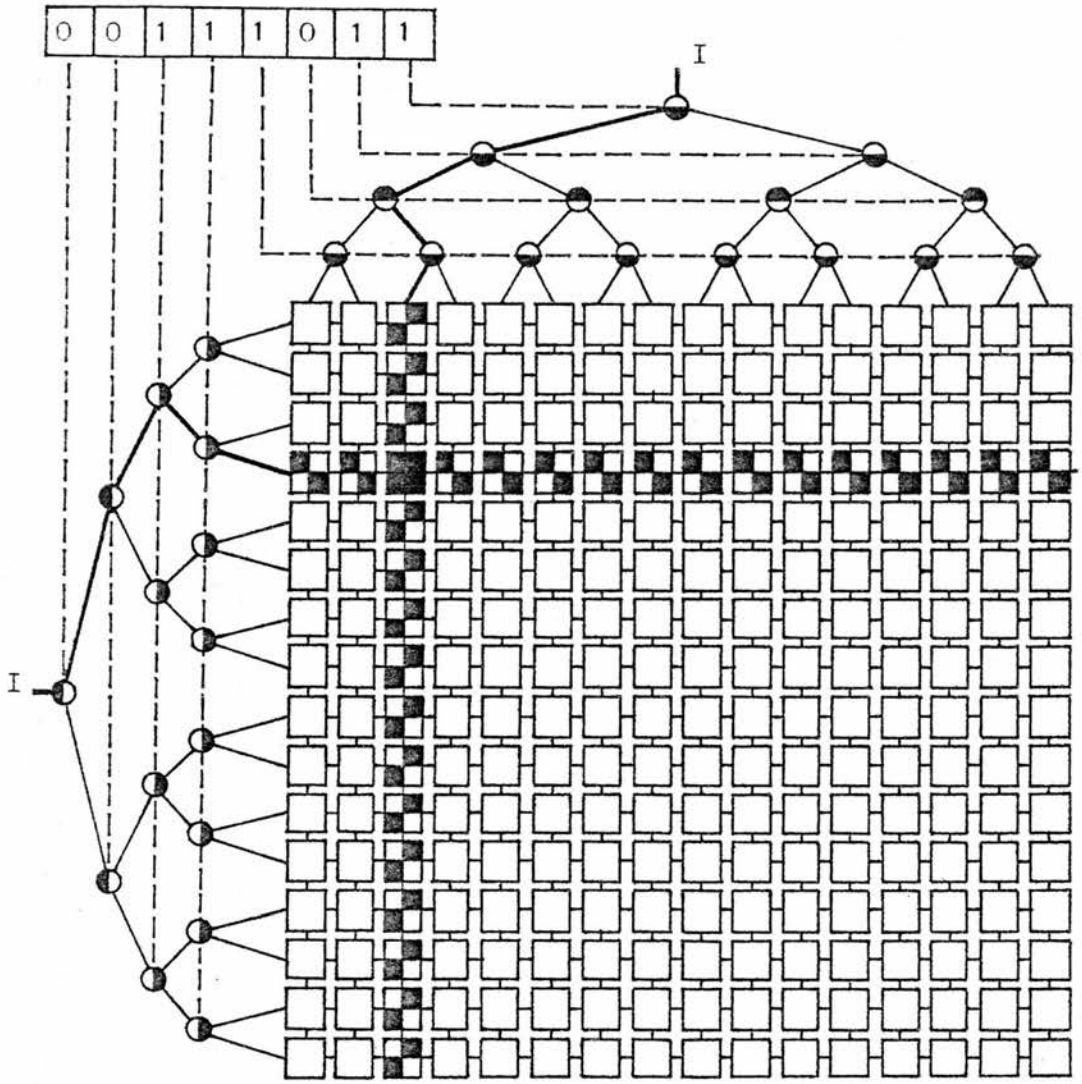
Decoding Tree for Schematic Machine

Figure 6

store. It is thus possible to divide the operation code into two sub-fields so that the first digit defines what action is to be taken in the storage unit, as show in Figure

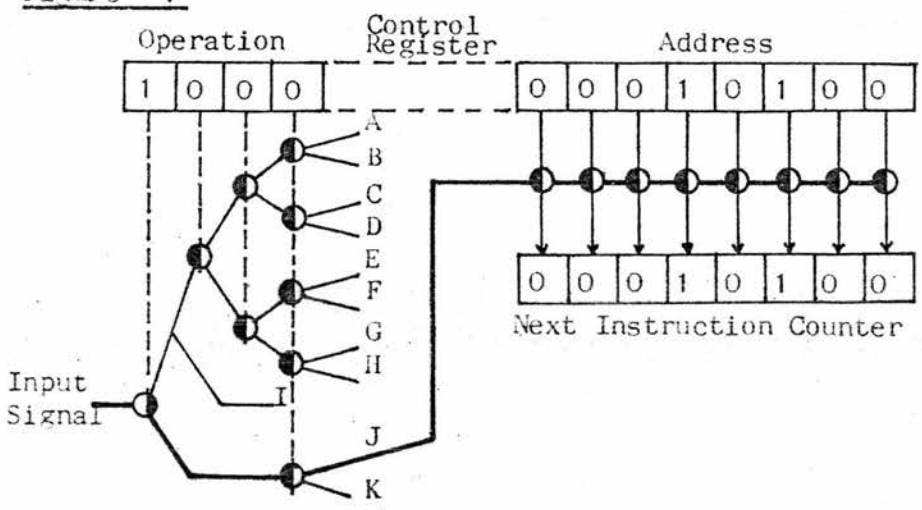
In Figure 6 the sequence of output signals A-H corresponds to the sequence of operations given by the codes 0-7 in Table 1. Each of these is performed in the arithmetic unit and also requires some operation to be performed in the storage unit. Outputs J and K correspond to the jump operations which act on the number held in the control unit counter. The way that the output I selects a value from store is shown diagrammatically in figure 7, and the basic operation in executing a jump instruction is shown in Figure 8.

Even in this very diagrammatic description of the control processes there exists a wide variety of ways that a designer could choose to implement them as machinery. Generally speaking, the execution of an instruction code involves a sequence of transfers of data from one register to another. M. V. Wilkes (1951) coined the term micro-



Data Access from Store

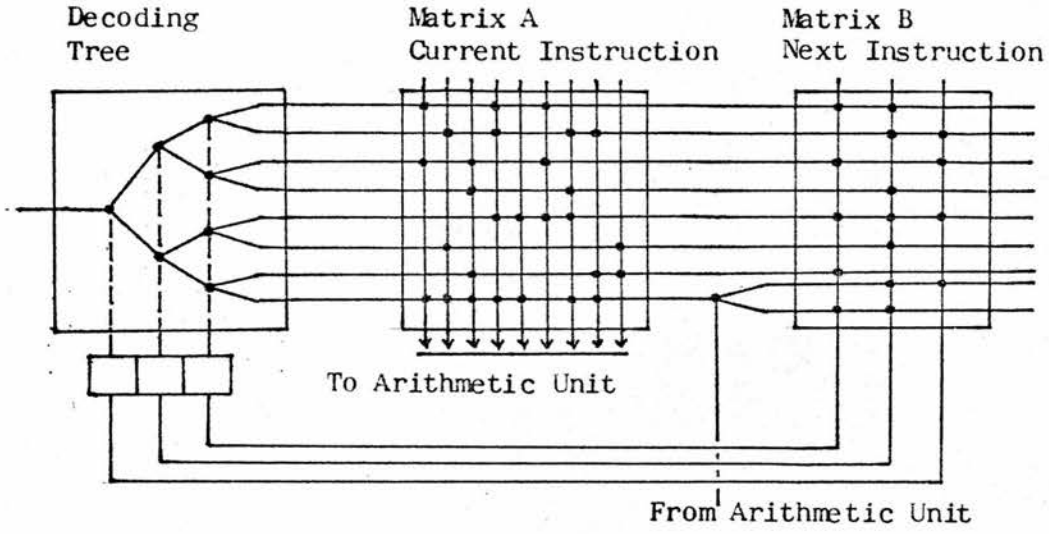
Figure 7



Jump Instruction

Figure 8

programming for a design approach which provided a systematic way of ordering these actions, to replace the relatively ad hoc approach taken up to that point.



Microprogramming Circuitry

Figure 9

The microprogram is held in a read only memory, in the scheme shown in Figure 9 consisting of two diode matrices labelled A and B. The output from matrix A is connected to gates in the arithmetic unit and other control points in the computer. The accessing circuits for this memory consist of a decoding tree with an associated address register. A timing pulse T enters the decoding tree and the resulting output from matrix A brings about the first micro-operation. The output from matrix B is fed by way of a delay circuit to the address register of the decoding tree, so automatically setting up the next micro-operation.

One of the lines from matrix A to matrix B is shown branching. The

direction taken by the timing pulse in this input to matrix B will be controlled by the input from the arithmetic unit. This permits conditional choices of next instructions to be made corresponding to 'jump if negative'. In a similar way the sequence of control can be made to depend on other states in the machine. Not only did this provide a standard approach which could be adopted for this level of machine architecture, it meant that the machine code formats could be standardised into a uniform product. It can be seen from the previous discussion that the form of a machine instruction will reflect the structure of the circuits used to execute it. This approach made it possible to simulate the code of one machine by microprogramming another machine which would normally have a different instruction set. Since it would appear that the most efficient form for instruction codes would be the one which most directly matched a machine's internal structure, this property was used by manufacturers to create a compatible language for a range of different sized machines. The code reflected the structure of the larger machine but could still be used on the smaller machines in the same series. As an approach, however, it allows the machine code instruction to be more versatile in that a single instruction could take a variable time in execution, depending on found properties in the data.

Developments of Input-Output Units

The improvements to the internal machinery of a computer accentuated the problem of getting information into and out of the machine. Most of the early systems depended on the use of punched cards or paper tape for data input. It can be seen that if a read command is bound by the

mechanical speed of a card reader, then most of the machine time will be spent in waiting for the data to arrive.

Not surprisingly, the development of input-output facilities probably reflects the demand pressure of machine users more directly than developments in other aspects of the machine system. Input output units were not considered of very great importance by the early scientific users. On one hand, they were people trained to use mathematical notations, to whom machine codes would not represent a great obstacle, on the other their problems tended to be computation-intensive, requiring relatively small amounts of input data for given amounts of calculation.

However, as other users appeared, where their goal was to carry out relatively simple operations on large quantities of data, this situation changed. When the United States Bureau of Census ordered a Univac I, data input and output must have been a major consideration. The Univac I, the creation of Eckert and Mauchley, was regarded as several years ahead ot its competitors when it was delivered in 1951 (Rosen, 1969). It had a magnetic tape input unit, which could read either backwards or forwards, and data was entered in blocks through buffers. The speed of the tape reader was high, even when compared with machines of a later date. The concept of reading data directly onto magnetic tape from a key-board, though it posed problems in editing and verification, took into account the growing difference in manual and machine speeds of operating. The magnetic tapes could be prepared off-line, in parallel and when they were ready, their contents read into the computer sequentially at a reasonable speed.

In a similar way, the output to line printers and other peripherals was improved by providing a system structure where data output from the main processor could be stored and then fed in parallel to a collection of units acting at a slower speed. The general system design problem being the same as the organisation of a factory production line using machines of different capacity and speed.

Communication between the Units

The consequence of these developments was that communication between all the units which were being included in a single machine system became a major problem in its own right. The introduction of the program interrupt mechanism made it possible to efficiently organise a system of units which were effectively running in parallel at their own speeds, but needed to communicate with each other for certain operations. This facility resulted from a request by Richard Turner (Rosen) working with NACA (later NASA) and was initially incorporated into the Remington Rand 1103. The interrupt principle, once it was established, became the basic starting point for the design of future computer systems. The IBM 709 system delivered in 1958, by time sharing the central processing unit between computational requirements and as many as six data channels, was able to offer many improvements in the level of service provided by the input output units. Data could be read from magnetic tape and card readers, and written back onto magnetic tape or directly to line printers. In essence, the interrupt led to the control unit operating on more than one level. Normal data processing was carried out at a lower level, and the occurrence of an interrupt automatically switched the control unit into a higher state,

where the cause of the interrupt could be investigated and dealt with before returning control to the lower state and the interrupted process.

As the size of machine systems increased with the advent of transistors, the problems of coordinating all the separate units which could be linked together became a major problem. The LARC (Livermore Atomic Research Corporation) computer incorporated an input-output processor, several computing units acting in parallel, all interacting with high speed core storage. The input-output unit was itself a stored program computer. Though this arrangement provided a great deal of versatility, it proved difficult for systems programmers to provide a control system which could obtain an optimal performance from the complex hardware. Rosen states that virtually every program run could be degraded in performance if the system programs were inadequate.

The LARC computer and the Stretch computer, another early large machine with advanced hardware features, can both be viewed as intermediate developments in the evolution of a new level of computer product. In the Stretch computer the processing units were fed by 'look ahead' units which collected and decoded several instructions ahead of the one currently being processed, providing a stream of instructions and operands to one or more processors at a greater rate than in a strictly sequential system. The problem with this machine was again at the programming level. Because it was difficult to implement a multi-programming system it meant that except for a relatively few large programs, the full power of the machine was rarely in use.

PRODUCT DEFINITION IN COMPUTER-BASED SYSTEMS

Standardisation of Products

With the development of the machine interrupt and a standardisation of interfaces between different machine units, it became much easier to construct machine systems on a modular basis. Other moves towards standardisation began to appear in many areas. In 1961 IBM started the design of their 360 series of computers. The goal was to standardise within IBM such computer characteristics as instruction codes, character codes, units of information and modes of arithmetic, so that in theory at least, the same program could be run on the smaller machines in the series as the larger machines. The success achieved by these machines can be shown by the fact that many of the standards established in this range were adopted by other manufacturers for their own machines. Whether these standards can be regarded as optimal products in any overall sense is made difficult to decide because of the very powerful position of IBM. Compatibility of software was becoming an important consideration for the consumer, and consequently it could be argued that any system which worked and was adopted by the mass of consumers, must be adopted by the rest of the industry where it affected the form in which programs were written.

Development of Operating Systems and Languages

However, by the time that the 360 range of computers was on the market, many aspects of computer languages and computer programming which could affect a consumer's choice had become independent of an individual machine's structure. This was not the case for the earliest machines. The simple example given for the schematic machine at the beginning of this chapter, shows the programming problem faced by early machine users.

Even in this example the operation codes were referred to by mnemonic names, but to permit data to be input in this character form itself requires special provisions to be made in the design of the hardware. The Manchester Mark I read its input in a way which favoured the direct use of binary codes in writing programs. The EDSAC 1 computer in Cambridge, however, was designed so that an assembler program could be used to convert mnemonic names into the appropriate internal codes.

As machine use built up, it is not surprising that programmer convenience should set the pattern for development, since demand pressure would select the form of machine which was easiest to use. The same pressure led to the development of composite products in the form of subroutines and subroutine libraries. This meant that procedures which had already been programmed could be used in new programs and consequently save a considerable amount of effort in duplication. Primitive forms of subroutine were available, even on Aitkin's Automatic Calculator, but it was work on the EDSAC, with its automatic relocation facilities which made the subroutine libraries easy to use, and established them as the basis for future programming techniques.

The work on computer-orientated languages was extended by many groups of research workers. A. Holt and W. Turnaski (1958) developed a compiler and a concept called General Programming. Their system assumed the existence of a very general purpose subroutine library. All new programs would be structured as if they were library programs. A program was assembled at compile time by the selection and modification of particular library routines. The input program provided the information necessary for this selection and modification process.

Subroutines in the library were organised in hierarchies so that one level of routines would call routines at a lower level. In this system the important advance is the fact that the machine which the programmer uses is not merely the hardware machine, but the 'extended machine' - consisting of the whole system of hardware and software in the library. This definition of a system includes a wide variety of facilities and services not available at the primitive machine level. More important perhaps is that the facilities are routines or modules in a general system which can be added to and have parts replaced, so that over time it could accumulate a wider range of procedures and could also be updated to keep the system competitive.

The development of 'problem orientated' languages was an attempt to make the task of the programmer simpler. Early work in this field was carried out by a research team in Univac working under Dr. G. Hopper. This group investigated a series of languages which were related to the form in which problems would be expressed by the user, rather than the form in which the same problem would have to be expressed to make use of a particular machine. Among these languages were an Algebraic translator which later contributed to ALGOL, BO which influenced COBOL, and the first large scale symbol manipulation program which differentiated algebraic formulae as operations on an algebraic notation. Following these experiments various languages which are now accepted as established and relatively stable products were slowly evolved. J. Backus, working for IBM, brought FORTRAN into existence. In 1958 a meeting of GAMM and ACM led to the preliminary definition of a new international language for describing computer procedures in the ALGOL

58 report. In 1960 the final form of this definition was published in the ALGOL 60 report. In 1959 the early developments of COBOL were initiated by CODASYL - the Committee on Data System Languages. Each of these major languages had many teething problems to contend with before it became fully operational. In the case of COBOL the language was only really accepted after the United States government refused to purchase computing equipment from any manufacturer unless a COBOL compiler was available with it, which was the form of demand pressure which no manufacturer could ignore (Rosen).

After this period programming systems or operating systems could only get more complicated. If an installation was to provide a variety of user languages, then the bookkeeping and internal administration of the overall control system had to increase. The aim became that of providing the user with simpler procedures, and, where possible, standardising them from one installation to another. The program operating system SOS - Share Operating System - developed by IBM aimed to provide a system in which the programmer would only need to know one user language to be able to run programs successfully. To provide a simpler set of procedures for the user, with a wider range of overall facilities meant that the system had to automatically handle many of the operations previously handled by the machines operator.

The evolution of the operating system led to the machine system being divided conceptually into two levels. This provided many practical advantages. On one hand, it prevented the average user from being able to interfere with the efficient way in which jobs were run. It provided automatic traps for illegal operations, and allowed diagnostic messages

to be provided when programs failed. On the other hand, the system was able to organise its work load to optimise the use of the available hardware. The task of coordinating peripheral devices could be handled by the operating system so that the administrative tasks involved were invisible to the user. Input-output units could consequently be made to run independently, except for interrupt messages, and this greatly improved efficiency where different operating speeds were involved. Multiprogramming became a possibility, so that when one program was being held up waiting for input or output operations to be completed, the central processor could be allocated to another program.

The earliest forms of operating system provided a batch processing service. All user programs were entered into a queue, and depending on their size, priority and other information provided with the program, they were selected by the operating system when the appropriate facilities were ready. Other possibilities were also developed. By 1959 the computer designers in Manchester in cooperation with Ferranti Ltd. had completed the design of the Atlas system. By using automatic paging, the system gave the user a large one level storage space. Though this was in fact 'virtual memory', the user could write programs as if he had sole access to a very large machine. Automatic paging not only made storage problems easier, but also meant that program overlays could be handled by the system.

The extension of the use of program interrupts from coordinating the parallel operation of many peripheral devices to handling different users working in parallel gave the time sharing system. Project MAC in MIT pioneered some of these developments. Each user program was

allocated time on the central processor unit in time slices, by the executive system. Bearing in mind that a user, when directly connected to a machine, will on average only employ one minute of central processor time for every hour of connect time, this facility makes the direct interaction of man and machine into a feasible proposition.

Time sharing provided a fundamentally new form of computer service and led to new structures being developed in hardware layout to accommodate it. General Electric proposed a modular structure which would permit multiple processors to communicate with multiple memory units and peripheral devices. They extended the paging system of Atlas to include another level of grouping called segments. The use of multiple processing units accompanied by the time slicing used in time sharing, changes the nature of the operation from being sequential to a complex mixture of parallel and sequential processing. The development of parallel processing offered major increases in speed in certain situations, and experimental machines such as the ILLIAC IV were built to investigate its potential. Parallel processing makes use of spatial ordering as opposed to temporal ordering of activities but has proved difficult to employ in a general way

Computer System Hierarchy

The development of the operating system can be viewed as a part of the product differentiation process. It made the average user into a consumer of the system programmers work, who in turn made use of the components provided by the hardware manufacturer. A vertical division of a computer system into 'product levels' is given in Table 4.

TABLE 4 Vertical Structure of a Computer System

System Components	System Users
Computer Hardware	Operating System Programmers
Computer Hardware plus Operating System	Subsystem Programmers
Computer Hardware plus Operating System plus High Level Languages	Application Programmers
Computer Hardware plus Operating System plus High Level Languages plus Applications Languages	Application Users

Table 4 shows that one way in which the different levels can be characterised is by the language used by each set of users. Another way of viewing the division can be shown by considering the original schematic machine. The three line program necessary for entering other programs can be regarded as the primitive operating system, it could in fact be hard-wired into the machine structure. The data it receives must be expressed in binary machine language. The program which is entered, however, has to have two statements: the first which passes control to the entered program, and the second which passes control back to entering program when it is complete. Had the entered program not passed control back, being itself a more sophisticated executive program, then it too could accept and administer the running of other programs. This makes the vertical divisions in Table 4 correspond to levels of control, where data at one level provides working processes for the level above.

The relationship between this collection of vertical divisions and the horizontal divisions which are created in a modular system is interesting. They both reflect a functional classification, and the complete division between the sections is the product of a conscious design choice. The aim in creating a modular system is to retain flexibility, that is the ability to change a system without having to scrap the whole of the original system; it also spreads the load of risk should a component fail. The vertical divisions on the other hand, though they also provide versatility, do so in a different way. Each level down in the hierarchy serves a wider range of users than a higher level. A machine language routine can be used by all higher levels, whereas a particular applications program will be used by a small group of people with a special interest. Creating the vertical division means that a higher level process can be scrapped without having to return to basic machine principles to design and construct a new one.

Man-Machine Systems

If reference is made to the original information flow diagram in Chapter 1, Figure 2, it will be remembered that the forms of data which flowed between the boxes labelled 'organisation' and 'computer system' were restricted by the forms of data which people in the organisation could understand, in other words, interpret for meaning. It is clear that only those people who could interpret and use binary codes were in a position to use the early machines. Starting from this base an increase in the number of machine users could have resulted from three developments. Firstly, the number of people trained to use binary codes could have increased. Secondly, a more efficient use of the people who could

communicate directly with the machinery was made in organisations to act as intermediaries for other people. Thirdly, the form of input data and output data was made more naturally acceptable to people in general.

In the short term there was a fairly effective limit to the advance which could be made from the first two approaches, and because of the extensive nature of data preparation using machine codes, the third approach was probably the only one which provided any long term potential. The major difference in the form of simple codes used in the schematic machine and a written instruction in English is that the former has a fixed number of symbols whereas the latter can be of variable length. The first advance in this direction was made in the interpretation of arithmetic expressions of the form '(A+B)*C+D'. Many methods were investigated, but it was the publication of a mathematical model of a grammar by Naom Chomsky in 1956, based on his analysis of natural language structures which provided a general way to create formal languages of a freer structure.

Starting with a set of basic characters as an alphabet, the problem was to distinguish the strings of characters which were valid sentences in the language, from those which were not, and then to interpret the valid sentences into the appropriate sequence of actions. Since most of the languages of interest would contain a very large number of sentences, the first difficulty was to represent in a relatively compact way the valid sentences from the language. It was plainly impossible to store all the possibilities.

The approach which had to be adopted was to find a procedure which could determine if an arbitrary sentence was in the language or not, based on a limited set of rules. Another way of viewing this problem was to find a set of rules from which all the valid sentences in the language could be generated. It is this form of generating system which Chomsky called a language's grammer. The formalised definition of a grammar requires four basic concepts:

a) Syntactic Categories which classify different objects
b) Objects which are Terminal or Variable Symbols
c) Productions which are relationships between strings of terminal and variable symbols showing valid forms of substitution
d) Source Symbol which is a variable from which all sentences in the language can be derived by the appropriate substitutions of variable symbols, governed by the rules expressed in the list of productions.

A grammar 'G' can therefore be summarised by (V_n, V_t, P, S) where V_n is the set of variable symbols, V_t the set of terminal symbols, P is the set of productions, and S is the variable symbol which acts as a source from which all the valid sentences can be generated. Depending on the nature of the productions it is possible to define a variety of language types, but only some of them are possible to use for creating simple testing procedures using computing machinery.

There are several ways in which a grammar definition can be used to establish whether a stream of input characters can be interpreted as a valid language sentence. The source symbol for the language can be taken and then, by systematic substitutions using the list of pro-

ductions an attempt can be made to match the input string. Alternatively, the characters in the input string can be compared with the productions and by substitution of valid alternatives an attempt can be made to reduce the original string of characters, which must by their nature be terminal symbols, back to the source symbol. Once such a recognition process is successful, it establishes that the data is in an acceptable form.

The grammar allows more than this to be achieved, however; it also gives as a result of the recognition procedure a classification of the input string into a collection of hierarchically ordered syntactic categories. It is this structuring of the input string of characters into higher order elements which can then be used to create a stream of machine level commands which will interpret the input data into a sequence of actions. The first phase of this process is called syntactic analysis, the interpretation phase is called semantic analysis.

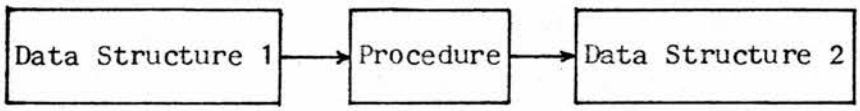
Shannon and Weaver (1969) define three levels at which the problems of communication can be considered:

a) How accurately can the symbols of communication be transmitted: the technical problem

b) How precisely do the transmitted symbols convey the desired meaning: the semantic problem

c) How effectively does the received meaning affect conduct in the required way: the effectiveness problem

It is clear that, following the input of a statement into a computer, its interpretation in the semantic phase is widely different from the interpretation which a person reading the statement would make. The machine process appears too primitive to say that the machine interprets the meaning of the statement. However, the concept of meaning is difficult to define. If a person wishes to find the meaning of a word he can look it up in a dictionary. There he will find other words or sentences which have the same meaning. This is essentially the same procedure expressed in Figure 10.



Meaning as Data Transformations

Figure 10

At first sight there seems to be no difference between this process and the machine operations on data structures which have been described in the earlier parts of this chapter. However, it can be argued that if the second data structure, in other words the new words found in the dictionary do not have a meaning the person will be no wiser. To start with this suggests that the process is that of associating the meaning of one set of words to a new word, and secondly the meaning established in this way will depend on the procedure adopted. For example, a word - as a data structure of undefined meaning - can be given different meanings in different languages, and for that matter different meanings in the same language as shown by the two sentences:

"The door was heavy as lead".

"Lead him down those stairs".

This leads to the conclusion that a word's meaning is independent of the data structure used to represent it, and that it is something which has not yet been defined.

G. Frege (M. Firth, 1964) considered the meaning of a word to be some test which was associated with the word which could be used to establish whether an unknown object could be denoted by the word or not. The definition of a word in a dictionary will provide such a test where the explanation is understood. This interpretation of meaning can be successfully extended to apply to the primitive level of direct sensory experiences. If blue light falls on the retina it presumably creates some internal effect which can be stored. If this effect is then associated with the word blue, then presumably any further stimulation by blue light will create the same effect. By comparing the new effect with the old effect it will be possible to classify the new effect as that of blue light.

Using this idea as a starting point, it is possible to suggest that the meanings of words used by people are a complex hierarchy of transformations which rest on primary classifications at the level of the senses. It also suggests that, at this basic level, the meaning of a word is very directly related to the mechanisms which convert the external forms of sensory data into internal forms for storage. The process is comparable with syntax analysis as the process which establishes whether a new string of data can be called a statement in language 'G' or not. However, in this latter process, the only words which are being allocated meanings are the names of the syntactic categories.

The string 'CAT' is recognised as a <name> but as a name, unless it is associated with another recognition procedure which can, for example, distinguish cats from other animals, has no meaning beyond those associated with all names.

Several lines of research have resulted from this form of analysis. On one hand, the machine system can be directly linked to its environment by its own sensory inputs, instead of the indirect link through language statements. A variety of different forms of machine perception are being studied under this heading. This approach would appear to provide the same level of primary inputs received by a human being, and perhaps eventually provide the basis for the total range of meanings for words which can correspond with human experience. On the other hand, there is the problem of using the meanings of words in the reaction to language statements. This requires, among other things, the ability to deduce from new statements valid conclusions which can act as the basis for future action, from already stored information. The starting points for this approach can be found firstly in the development of theorem proving algorithms, and secondly in the work being carried out on the structure of data-bases.

The initial problem in machine perception is the reduction of a continuous stream of low level data into a coherent and consistent pattern which corresponds to objects which exist at higher levels in the process of perception. Early experiments in this form of analysis were carried out by Guzmann in MIT for visual perception. The basic input was a stream of grey levels obtained by scanning a photograph. The first stage was to convert the grey level information into a

series of object boundary lines by a form of spatial filtering. Once this had been done an analysis of the junctions between these boundary lines made it possible to construct configurations of objects in space which could have created the original data. In general terms it would appear that all forms of data analysis from statistical techniques using time series to picture processing can be classified under this approach as a form of perception. It would appear that any invariant property of the original data could act as the basis for object recognition at a higher level in such a process.

Minsky and Papert (1969) in their book 'Perceptrons' discuss the properties of a range of retina like devices which can discern different kinds of spatial properties. The operation of these machines is closely akin to spatial filtering in geographical analysis, and in picture processing: the randomly linked perceptron for example being the counterpart to the random spatial sample. A series of different approaches to the same problem is presented in the Proceedings of the IFIP Working Conference on Graphic Languages edited by Nake and Rosenfeld.

Another approach which will eventually link up with this kind of automatic perception, is presented in T. Winograd's study 'Understanding Natural Languages'. Winograd develops a system which not only uses an English language form of input and output but also which models the meanings of the words used. The system simulates the actions of a hand-robot where the world in which it is acting is made up from a series of toy blocks on a table. The hand can be instructed to perform a range of operations on the blocks, of varying complexity, some which

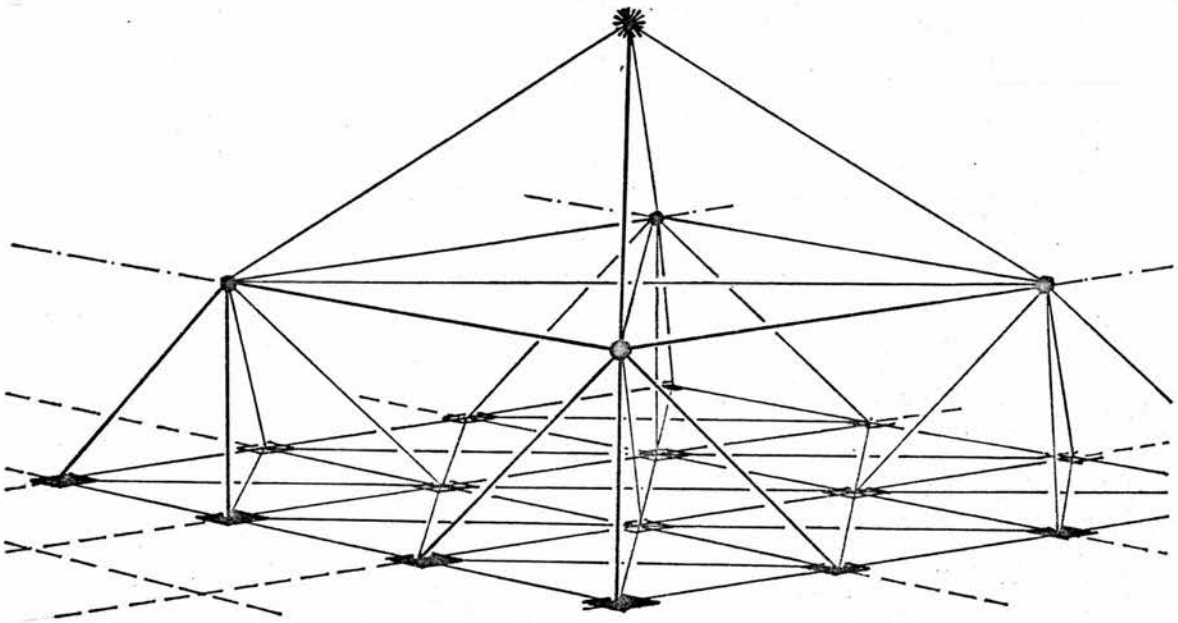
can be directly carried out and some which involve preliminary operations to make the requested action possible in terms of the behaviour of the blocks, for example, under the action of gravity. The system can deduce from its initial model of the world in which it is operating the consequences of various actions by using a deductive subsystem called PLANNER. Though this experiment is limited in this case to a small hypothetical world, Winograd suggests that it could be a model for other applications "which involve integrating large amounts of knowledge into a flexible system".

The significance of this work on 'intelligent' systems is not just that it allows a form of man-machine communication which is closer to man-to-man communication but that such systems can create their own view of their environment. Winograd presents the results of the actions performed by his hand-robot in a graphic display of the hypothetical world in which it is operating. The geometrical model of the objects in the scene serve more than to create a display. This model provides the associated description to the names of objects which is an important component of the meaning of these words when they are used to instruct, or communicate with the robot. It seems relevant in this context, that the meaning of many simple object names occur as mental images of typical objects, rather than as a verbal description or a dictionary definition. Though in this case the model is used to simulate the environment, its presence would appear to be necessary in the same form for the robot to evaluate the consequences of new actions. To link this system to a machine system for viewing the real world, such as a TV camera, some way must be found for creating the

internal model used for decision making, from the raw input data; and this links the discussion back to the parallel processing retina like devices examined by Minsky and Papert.

The importance of parallel processing is appearing in another area of computer applications. This is where computer networks are being set up to serve banks, universities and other organisations which have many machines under their control. There seem to be many natural advantages to having data processing units distributed in different locations rather than centralised in one place which the cost and scarcity of early machines demanded. This can be done within a single system because data transfers can be made through the existing telephone network. Though these distributed machines are presently used in a conventional way, new possibilities are emerging, as the number of devices multiplies. The potential developments can be illustrated by considering the schematic network which could be set up for local government departments. Starting with the three tier system of government, and assuming that each district has its own machine, then the layout and linkage between these machines could be that shown in Figure 11 . The squares represent the lowest level - the district, the circles the regional authorities, and the star represents central government.

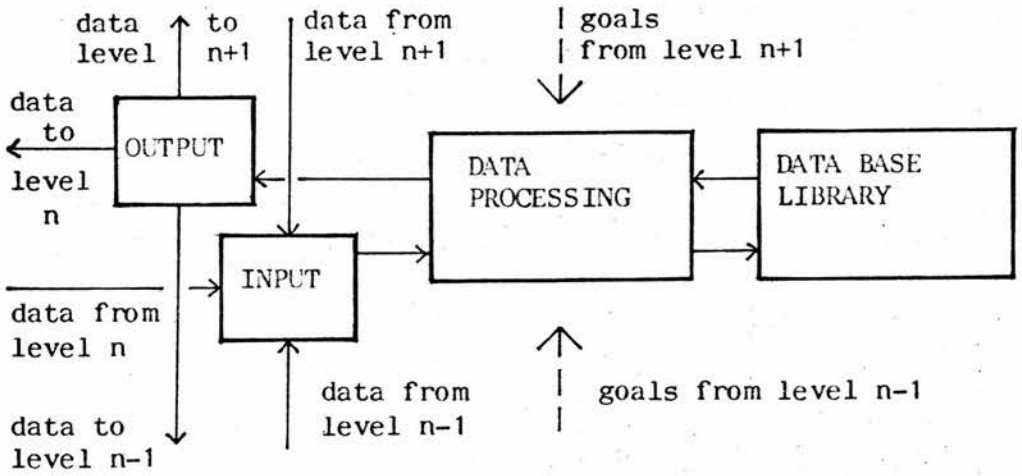
Though this arrangement is very diagrammatic, it does show the massive data collection potential of such a system. The parallel collection and local processing of information could provide the higher levels in the system with a continuous view of the behaviour of the total system,



Schematic Network for a Government Information System

Figure 11

in a way that is only attempted at a primitive level by the collection of census data. More than this is in prospect, however, each unit in this hierarchical network is a processing module outlined in Figure 12. This means that data at one level can be summarised and compacted locally before it is sent to a higher level. If all units in one



Basic Processor as a Module

Figure 12

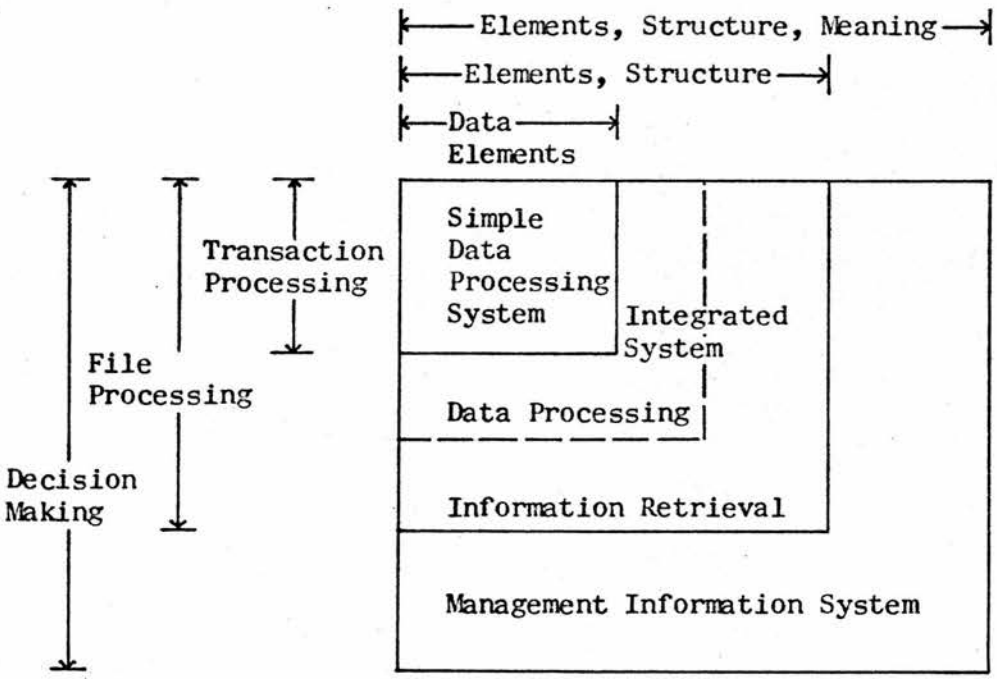
level perform the same operation it can be seen that this network is operating as a perceptron (Papert, Minsky, 1969). The behaviour of a particular perceptron depended on the operations and the linkage pattern between different processors. Neither of these is restricted in practice to the simple geometry shown in Figure 11 , or to primitive operations since each local unit can be a full scale computer.

Perhaps an even more interesting prospect is the reverse flow of information. Except for very slowly changing phenomina, central policies and laws have to be constructed to be applied to all local situations. Leaving aside the idea of equal rights for all people in front of the law, this is an administrative necessity. What the network structure permits is the equivalent in the data processing world to the independent but accountable interpretation of policy decisions by local government officials, but while taking account of local knowledge is possibly able to provide a more detailed and swifter response, based on a wider spread of information than is possible now.

The only insight into the possible advantages and disadvantages of such an arrangement is provided by the money system. The information being value, and the data being cash, either as currency or in the wide variety of credit forms, it is possible for a very large system of exchange to operate smoothly, and without any form of overall planning to allow people to coordinate their activities to everybody's common advantage. The disadvantage of such a system can be seen in the way that money can localise power, and in the way that time lags

in the system can lead to oscillations such as the trade cycle. Greater flexibility may create less stable behaviour in the total system.

The latest developments in the basic technology on which computing is based also reinforce this trend towards parallel processing. The ability to place the major portion of computer processing units onto a small number of silicon chips, means that the price of computers as they have been described at the beginning of this Chapter, has been reduced to the point where they can be mass produced for a much wider market. Furthermore, they can be used in a redundant way so that parallel processing becomes an economical proposition at all scales of operation.



Classification of Computer-Based Information Systems

Figure 13.

Figure 13 shows the extension to the set of hierarchical levels in a Computer System given in Table 4, proposed by J. D. Aron (1973). This classification includes the developments being made towards 'knowledgeable' systems.

Having discussed the growth in computer technology working up from the development of the first machines to the present possibilities, the next stage is to consider the applications of these developments in the future. Based on the arguments in Chapter 2, the final outcome must be left to look after itself. However, no development is completely random, and it can be seen that though the use of the kind of networks which have just been discussed cannot be predetermined, they are, like the road system, an infra structure which has to be designed and implemented based on the expected demands of the total system. This kind of prediction poses problems which are considered in the next Chapter.

CHAPTER 4

COMPUTER BASED INFORMATION SYSTEMS AND FUTURE PLANNING

COMPUTER BASED INFORMATION SYSTEMS AND FUTURE PLANNING

In the previous Chapter an outline was given of some of the main developments in computer technology which provide the existing 'components' from which computer based information systems may be constructed. The evolution of new products was discussed, as new areas of application were found and new techniques developed. The constructional, trial and error aspect of this work was emphasised; and it is clear that the present classification of components is not final at any level, since the process of product differentiation cannot be considered complete until the potential advantages offered by computer technology have had time to work their way through the 'whole system'. In practical terms it will probably never be possible to say when this has happened. Even the impact of a particular new technique is hard to trace beyond the point where its immediate applications have been considered. However, it seems reasonable to say that even if innovations at a basic level in technology could be halted, it would still take some time before experiments into the applications of what is already available, cease to be productive.

In this Chapter the alternative design approach is considered. Instead of starting with a collection of components as building blocks, this approach starts with a 'total system' and subdivides it in an attempt to find an appropriate arrangement and definition of its parts. It has already been observed that this process is difficult to carry out without being biased by existing technology. The primary function of this approach, where the required components are unknown, is to provide a framework in which useful applications for new processes may be 'recognised'.

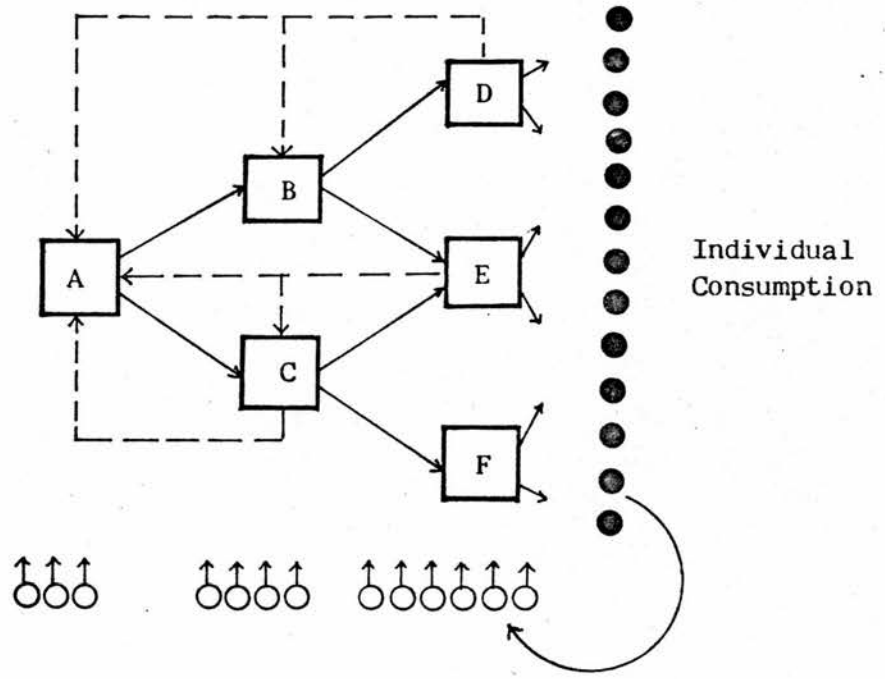
Without some recognition process it is hard to see how any useful invention will ever be located, particularly if it is created by accident.

The problem in taking this approach was how to define an appropriate system to start the analysis. The difficulty was to find a system which could be used to express the change which a new technology would produce, while at the same time having a classification of components which would not depend on any particular technology. Another difficulty which is perhaps a different aspect of the same problem, is that it is not possible to take an organisation or even a whole industry as the context for the discussion when as fundamental a change in technology as the adoption of automatic data processing is being considered. It seems to be necessary to include all the activities of a region or a nation within the system considered.

The starting point for this discussion is provided by the diagram given in Figure 1. This is a modification of the relationships normally expressed in an input-output matrix. The aim is to show all the intermediate products which are dependent on a particular process and to show the link between this process and the range of goods and services which are consumed by individual people. It can be seen that this diagram is an expansion of Figure 1, Chapter 1, and represents a closed system since the collection of people who are the consumers are the same collection of people employed in the various production processes represented by the boxes A - F, in the diagram.

From this diagram it can be seen that the concept of new technology

must be considered as a relative term. It suggests that, for a particular process 'A', the number of intermediate stages which have to be passed through before the individual consumer is affected is larger for a 'change in technology' than for a simple change in product.



Dependent Products and the Ultimate Consumer

Figure 1

For certain design operations such as designing household articles, the designer can use his own experience as a consumer to guide his creation of new products. Where the process is several stages removed from the consumer, this is clearly more difficult. New forms of product created by processes several stages removed from the individual consumer may make it possible to completely restructure the production processes for all the intermediate stages. The overall complexity of such a readjustment to the system may well be beyond the control of any external planning process because of the number of activities which

have to be coordinated with each other. This means that these changes have to depend on the mechanism outlined in Jenner's model to bring them about (Chapter 2).

If the primary process 'A' is taken to be computer services then the immediate demand for its products will be found within existing organisations, and in the production of existing products. These uses will be a function of the existing technology, and will not necessarily represent the optimal use of computer services generally. There may be potential products from 'A' which could create different intermediate organisations between 'A' and the ultimate individual consumers. To identify this kind of possibility requires the design analysis to be started at the level of the consumer and then to be worked backwards towards 'A'.

The advantage of doing this is that the general needs of people can be taken as relatively constant whatever the nature of technology used to supply them. These needs, if expressed generally as a need for food, shelter, warmth and so on rather than for specific items such as bread or butter, will not change in nature even if they change in quantity. This system is independent of the physical environment and the demands of this system on the physical environment can be changed to serve the requirements of the human components of the system. It is necessary, however, to unite these demands under a hold-all classification such as energy - again to avoid forms of description which imply the use of particular technologies. It is by considering the ways in which the basic needs of people can be satisfied that provides the least restricted

starting point for analysing the potential developments of a basic new technology.

It has become apparent that using computers provides a way of carrying out tasks which previously required people to read and understand language statements. It has already been seen that the impact of machine use in this area will affect the organisational structure of many processes carried out by people at present. To get some general view of the overall possibilities offered by these new techniques, it seems to be necessary to pose a series of almost naive questions about the nature of the service provided by machines, language and organised behaviour to the mass of individual people.

It seems reasonable to expect that the benefits which result from the continued development and diffusion of computer based methods will be an extension of the benefits which people enjoy at present from the use of languages and machines, and the acceptance of organised behaviour. There are two components to these benefits. The first is the ability to obtain knowledge about what exists in the environment at present, and what will exist in the future. The second comes from having greater power or resources with which to take action, either to control what is happening or to avoid its future consequences. The use of machines greatly extends the scope for taking action by harnessing natural energy sources; the use of languages makes it possible for a person to obtain information from a wider area than that covered by personal experience, and the co-operation of people in groups, among other things, provides an insurance cover against accidents and failure at the level of the individual.

Probably the most significant property which these three entities have in common is their ability to transfer the knowledge of one generation of people on to a following generation. This naturally leads to an accumulation of knowledge over time, with the corresponding increase in the inherited advantage. One aspect of this cumulative effect depends on the processes of education and learning. Each child has to learn to speak, people have to learn to use machines, and a major part of formal education is designed to prepare people for roles within an existing organisational structure.

Learning can be classified into three types for the purposes of this argument. At the primitive level there are the trial and error methods of gaining experience. At this level there is very little that a second party can do to help, apart perhaps from reinforcing success and certain aspects of the preliminary stages of learning to speak would appear to be at this level. At the second level there is a form of apprenticeship learning. This is necessary where a skill has to be acquired by repetition based on imitation, or the guidance of a 'master'. It is the third level which seems to be the most important to the overall accumulation of knowledge. At this level the individual can acquire knowledge indirectly about things he has never personally experienced. The way that he can do this is intimately connected to the nature and structure of language.

Once a person has learned to speak and even more so when he has learned to read, he has access to more potential knowledge than he will ever have the time to acquire from direct experience. The power of the

language system is such that he will be able to understand this information well enough to make practical use of it. This is the same property which allows one person to pass on to another information about relationships which exist in the environment which the second person has not observed. This communication depends on a common perception and a similar classification of the events which have already been observed.

It is possible to regard this classification as information which has been accumulated by the language structure. This built-in information can be expressed as the outcome of multiple experiments of the form: "If we agree to call these experiences by this name, are the perceived relationships which they hold to each other consistent with the relationships which the formal structure of the language will impose on them?". It is the generating power of the grammar of the language when it is linked to the classification implicit in the pattern of meanings associated with words, which makes it possible for one person to learn from another using less time or energy than would be necessary to learn from first hand experience.

An experimental classification will be discarded if it produces non-sensical results when it is used within the everyday constructions of natural languages. This is not quite the same thing as saying that people will not use definitions which do not make sense, though it is very close to it. The emphasis in the first statement is on the extensions to the use of a classification which existing operations in a language will automatically create. It is assumed that there was some

reason for setting up the classifications in the first place; in other words, some area where its application worked. Even where a classification is only partly successful in producing valid statements outside the area which led to its creation, it may well be retained as the best method of expression so far found. However, a person learning such an incomplete scheme will have to obtain a personal experience of the inconsistencies in the system of representation before his knowledge can match that of his tutors. Apart from internal or logical inconsistencies in the system of representation, there is no other way that a person learning the language can become aware of its limitations. Any improvement in the system of representation will presumably improve the efficiency of this education process.

The use of names in language depends on consistent patterns being discerned in what is perceived. It would appear that any property which was invariant, that is a relationship which did not change when a data structure was transformed, could be considered 'an object' and given a name. By using classifications which relate to such consistent results from transformation operations, it appears possible to condense the representation of information. It is this form of data reduction which is perhaps the most general way in which theory making can be viewed.

The ability to compact data is very closely related to the idea of pattern. Gregory J. Chaitin, in an article - 'Randomness and Mathematical Proof', attempting to define randomness rejects many existing methods stating,

> "Clearly a more sensible definition of randomness is required, one that does not contradict the intuitive concept of a 'patternless number'."

Chaitin then says that this definition has only been possible to formulate fairly recently, based on concepts derived from information theory. The basic idea is that a patternless number cannot be compacted.

> "The smallest programs capable of generating a particular series are called the minimal programs of the series; the size of these programs, measured in bits, or binary digits, is the complexity of the series. A series of digits is defined as random if series' complexity approaches its size in bits."

In contrast, the non random or 'patterned' number can be shortened by using a generating program. In the same article Chaitin quotes an example of this argument being used by Ray J. Solomonoff to provide a general definition of scientific theory making. The basic scheme for this definition is given in Figure 2.

Observations	Predictions	Theory	Size
0101010101	01010101010101010101	⊢Ten Repeats of 01⊣	19
	01010101010000000000	⊢Five Repeats of 01 Followed by Ten 0's⊣	40

Figure 2

In the example given in Figure 2 it can be seen that the direct compaction of the observations would amount to "five repeats of 01". Since the observations can only be a sample of the total range of observations there are many possible extensions of this series. Of

Of the two examples given, the simplest is the one which expresses the pattern in the observation as 'repeat 01 n times'. This 'theory' can be expressed in 19 characters as opposed to 40 characters used to express the alternative theory. This example leads to the conclusion:

> "The task of the scientist is to search for minimal programs. If the data are random" - i.e. patternless, "the minimal programs are no more concise than the observations and no theory can be formulated".

When expressed in the bare form of the example given in Figure 2, the concept of a theory needs to be distinguished from the direct compaction of the available data. In this context it is reasonable to use the term model to represent the compacted data, i.e. '5 repeats of 01' and the term theory to express the idea that this pattern can be extended in this case 'n repeats of 01'.

In practice, where the data are more complicated this distinction is far more difficult to make. A detailed or more precise discussion of the problems of theory making lie beyond the scope of this study. What is of interest is the fact that a model or theory is able to express in a small collection of ordered data objects the pattern in a very much larger body of data which result from observation or measurement. By applying the correct procedure to the model data it is possible to recreate the original observation data and by extending the process it is possible to predict new observational data. Once a pattern generating procedure has been learned and can be referenced by a name in a language, any relationships between objects which conform to this pattern can be learned about in a very simple way.

COMPUTER BASED INFORMATION SYSTEMS AND FUTURE PLANNING

The use of language is necessary to teach people to use machines. In many cases the practical description of how to make use of a machine to carry out a task is simpler than the description of how to perform the same task by hand. Certainly, the description of how to use a machine is simpler than the description of how it works, or the description of how to build it. Consequently, the machine can be considered to contain built-in information, and its mass production and use can create a wider dissemination of 'knowledge' than would exist without it.

The direct benefit from a machine would normally be regarded as the way in which it extends a person's power to act. This is an advantage which can be inherited in a very direct way, simply by inheriting the machine as a physical object. This property, which depends on the working lifespan of the machine, is reflected in economies by its classification as a durable good. With adequate maintenance and the replacement of damaged parts, its lifespan can be independent of the people who designed it and even particular people who know how to use it.

A simple physical machine, for example a lever, improves a man's power to act, but does not take part in the choice of when and where to act. More complex machines are able to adapt their own course of action to achieve a specified goal. This requires some form of measuring device which indicates whether the machine is moving towards the goal or away from it. In overall terms, this does not change the nature of the machine. It merely means that more complex operations can be regarded

as a single task. The choice of when and where to act is still the human prerogative. In the case of more complex machines the task will be expressed in a more general way.

Extending this principle, it is possible to regard the role played by people in an organisation as similar to that played by machines. This implies that the choice of action is made by one person and the implementation of the choice is carried out by another. By selecting the appropriate description of the activity it is possible to regard the operation of the whole of an organisation as machine-like. It is clear that there is a limit to how far this idea can be taken. In the case of the simple machine, the coupling between its components is different from the coupling between the components of an organisation. Where one person instructs another to perform a task it is only the transfer of a goal description. The person who receives the instruction is not impelled to carry it out as the component of a physical machine is impelled to operate. The instruction communicated through language can be rejected and, if it is not, the person receiving it will have to find his own way of carrying it out.

The structure of an organisation can be inherited in much the same way that a machine is inherited as an object. This can be done because the structure can be defined in a way which is independent of the people who work in the organisation. This is achieved by describing the roles and their inter-relationships in such a way that people can be educated to fill them. A decision-making role which requires the skills of an engineer, for example, can be occupied by any engineer as far as many

organisations are concerned. The role of a mother in a particular family cannot be separated from an individual person. Where this distinction between the roles and the people who fill them can be made, then the structure of these roles is something which can have a lifespan which is independent of the human lifespan. By learning to live within an organisation, an individual obtains the advantages of processes which may have taken generations to establish. This will often be the cases whether the person knows how the organisation operates or not, so it is again possible to regard the organisation as containing built-in information.

The organisational structure of the total community has evolved so that it contains a balanced set of roles, adequate for the support of all the individuals within it. This balance is expressed in Figure 1 by the ultimate consumers for all goods and services being the same group of people who produce them. It is clear that the machines, language and organisational structure are all closely inter-related in a functional way with this balance. On one hand the development of language forms of communication would appear to be a necessary antecedent to the evolution of complex organisations. On the other, it is difficult to see a purpose for language existing outside the framework of corporate action.

At first sight machines appear to be of secondary importance to this primary system of human organisation and language. However, machines contribute to the available energy of the system. The energy resources of a community will clearly affect the priorities of its goals. In the

primary system alone, this available energy is restricted to the sum of the individuals' physical efforts. Consequently, the stock of machinery will very much affect the necessary roles in the organisational structure and also the priority or values given to different kinds of activity.

The first stage in analysing the general impact of machine data processing can thus be achieved by saying that it will result in an increase in the active knowledge which is available within a community. What is meant by active knowledge can be illustrated in the following way. From an individual's point of view, there is a practical limit to the amount of information which he can access in a given time period. Though in a sense there is no limit to the amount of information which can be accumulated, for example, in libraries, any individual is limited by the amount of information he knows or can look up and assimilate in a given time. Consequently, the information which can be used, in other words the active information or knowledge, is, like the body of skill, limited by the total number of trained people in the community.

The advantage will appear where the educational resources needed to provide a body of people trained to carry out a particular task are reduced when people have to carry out the task by computer based methods rather than carry out the same task manually. There will be a net gain, even where there is not a direct reduction of this form, if the speed of the machine makes it possible for a person to supervise many more tasks with the same knowledge that he needed to carry out the tasks by hand.

COMPUTER BASED INFORMATION SYSTEMS AND FUTURE PLANNING

At first sight the relevance this has to planning future development is that the scale or the resolution of detail which can be controlled in designed activities will be greater, and the point at which the lack of information makes it necessary to rely on the self-regulating nature of the overall system, is pushed further back. On more detailed analysis it is not at all clear what this means in practice, since it is not possible to estimate what the greater 'knowledge' for the average individual will do to the overall system. This takes the discussion into considering the effect of new media of communication. There may be different reactions by individuals to the amount of available information. One possible result could be that the speed at which information becomes available merely increases the instability of the overall system, cyclic changes such as switches in fashion and switches in political attitudes, could speed up and the actual size of units which could be effectively planned or designed in the 'human' environment be kept very much the same. The complexity of the planning operation would be increased but the overall result would only be marginally different.

At this point the significance of distinguishing between the human environment and the physical environment becomes clearer. The increase in the knowledge about the physical environment will make advances in the scale of operations which can be controlled and this area has been the first to be exploited using computer technology. This approach applied to the human environment raises political issues which lie outside the scope of the present investigation. Questions such as the rights of individuals to privacy and other rights of individuals lie at

the heart of these problems. What has been taken as a starting point is the idea that the amount of information which could be accessible to individual people could be increased by using computer-based technology. How such information is used must, to some extent, be left to sort itself out as the result of individual choices in the manner discussed in Chapter 2.

It is clearly possible to extend this form of analysis in many directions. For the more specific studies presented in the remainder of the thesis it provides a general objective. This objective is to find ways in which information can be assimilated in a more efficient way than is presently possible. Graphic forms of communication are capable of holding large quantities of information in a readily digestible form. The present restriction of their use is the time taken to create drawings and displays even where the desired result has been fully worked out. If appropriate drawings could be created as swiftly as ideas can be expressed in a spoken language form, bearing in mind the time it takes to learn to speak, then there is an outside chance that a considerable increase in the speed and effectiveness of communication between a machine and its user will result for certain types of information.

SECTION II

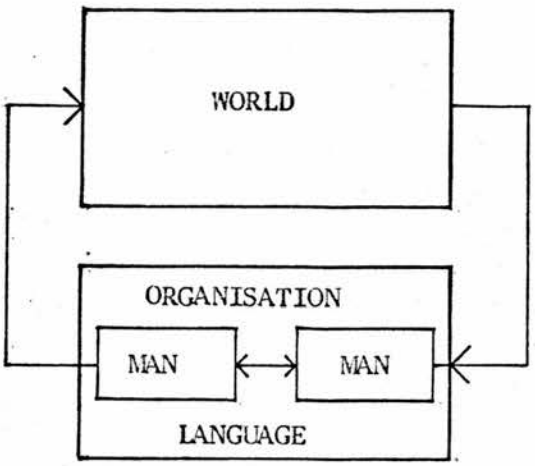
SPATIAL MODELS

CHAPTER 5

SPATIAL MODELS AS DATA STRUCTURES

SPATIAL MODELS AS DATA STRUCTURES

In this and following Sections, the problems of representing and using spatial information in a computer environment are examined. This demands a computer model to be created, using machine and program components which can potentially produce explicit statements about spatial relationships. Alternatively, the model may be required to create other forms of output such as drawings and scale models to communicate more complex spatial information to the system user. The starting point for this investigation was a model based on the properties of Euclidean space, since these correspond to everyday experience and are consequently employed in producing graphic output. Models of space are generally discussed in the context of the system defined in Figure 1.



System Context to Spatial Models

Figure 1

Man's primitive knowledge of space is provided by the sensory inputs of sight, hearing and touch. It must be assumed that the need to com-

municate detailed spatial information from one person to another, led to the development of language forms capable of expressing this information. It is clear at a basic level, however, that the properties of space are used to model themselves. Though the way in which this occurs in a scale model is fairly apparent, it is easy to overlook the fact that the formal properties of language depend on orders of data objects arranged in time or in space.

The evolution of language structures which can express the properties of space, or of objects in space, is presented in Jammer's book "The Concept of Space" (1969). The first attempts to systematically define these properties are attributed to the ancient Greeks. The early descriptions and definitions took on a variety of forms. Some equated space with a void, in other words a complete absence of anything; in contrast, others regarded space as an entity which permeated the universe, passing through solid bodies as well as places not currently occupied by bodies. The search seemed to be for a set of concepts which, when used as the basis for argument, in other words their logical extension through the natural use of language, did not lead either to contradictions or to conclusions which were incompatible with common experience. Questions such as:

> "Is space a finite or an infinite extension?"

were argued using questions of the form:

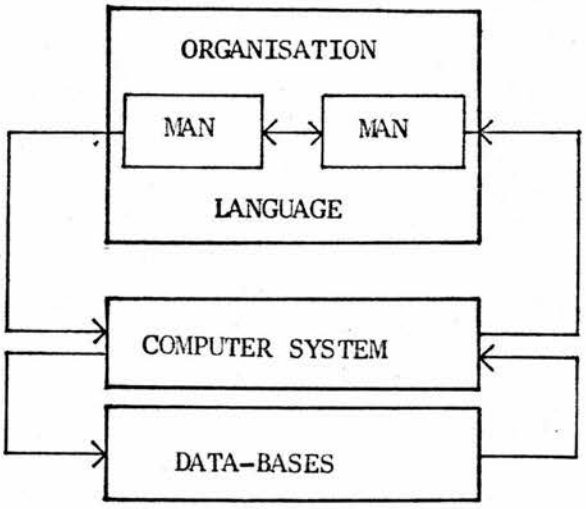
> "If a man is standing on the edge of the universe what happens when he stretches out his hand?".

This argument was the beginning of a debate which has continued as

a central issue in the development of physics and mathematics, and Jammer concludes in the final paragraph of his book:

> "Our knowledge of large scale as well as small scale properties of physical space is intimately related to the progress in cosmology and microphysics, respectively. And as long as these branches of scientific research fail to offer satisfactory solutions to their fundamental questions the problem of space will be classified as unfinished business".

If the use of spatial models in the computer environment is restricted to a study of the system shown in Figure 2 then most of the problems of the nature of physical space can be side-stepped. The problem becomes that of expressing the language forms used by man to describe spatial relationships in a machine form of data, and constructing machine processes for manipulating this data in ways which correspond to the external use of the language.

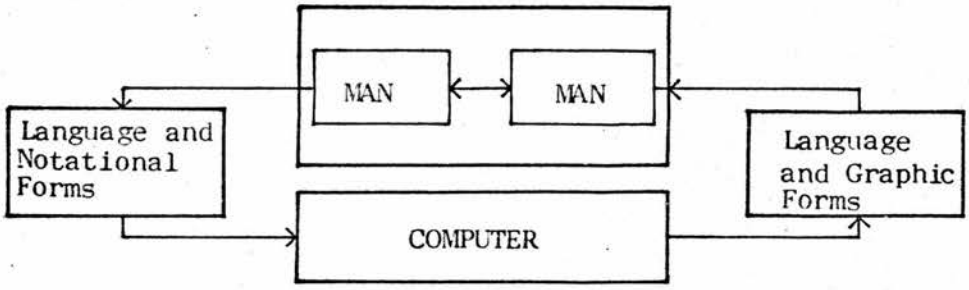


System Context for Spatial Models

Figure 2

Since machine perception is excluded from the initial stages of this

study the forms of communication available for transferring spatial information are shown in Figure 3. The computer is built to process



Communicating Spatial Information

Figure 3

language and digital data, consequently information has to be entered in this form. Human beings can often comprehend the same information more efficiently from other forms of data such as drawings. The emphasis in the first stages of this work has as a result been on how the input language forms of data may be converted into appropriate forms of graphic output. Within this context the process which is of interest can be expressed by the data transformations in Figure 4

DATA INPUT	DATA MANIPULATION	DATA OUTPUT
Operations on Language Model →	Computer Model	
	Operations on Computer Model →	Graphic Model or Language Model

The Primitive Computer Model of Space

Figure 4

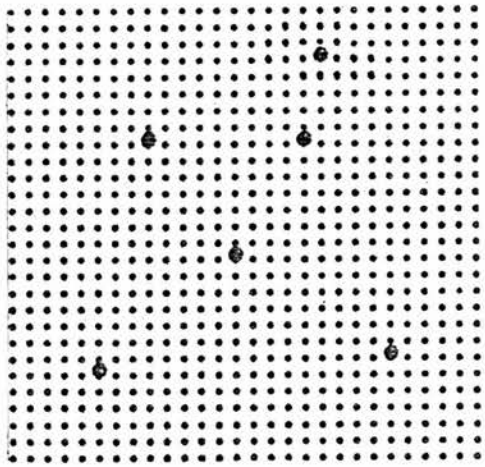
The simplest and perhaps most readily used computer model of space can be created by sampling a spatial distribution on a regular grid and

storing the sampled values in an array. The operations which must be carried out on this data structure can be found as standard array accessing functions present in any high level computer languages such as FORTRAN and ALGOL. In fact this model can be implemented using the mechanisms at the machine level of a computer system, because the model corresponds to the primitive storage structure of the machine. Its properties are that the data consists of a set of points, finite in size, where the property of each point can be accessed by the integral indexes of its array location. The dimension of the array will be the dimension of the space being represented. Within a two dimensional array it is possible to represent points, lines and area zones. Within a three dimensional array it is also possible to represent surfaces and volume zones.

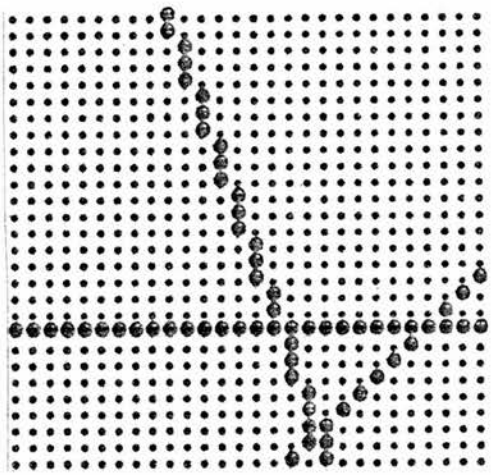
The important aspect of this model is the direct way it relates to some of the commoner output devices. Printing out a symbol to represent the value stored in each array location, in a table matching the structure of the array, will produce a primitive computer map. The maps used in the shopping centre location model in the final section are examples of this very direct way of obtaining graphic output. Where this form of output is required and other forms of spatial model are being used, then they have to first of all be converted into this array representation.

There are two restrictions which accompany the use of this model. The first is that there is a limit to the total size of a grid, for the simple and practical reason that there is a limit to the amount

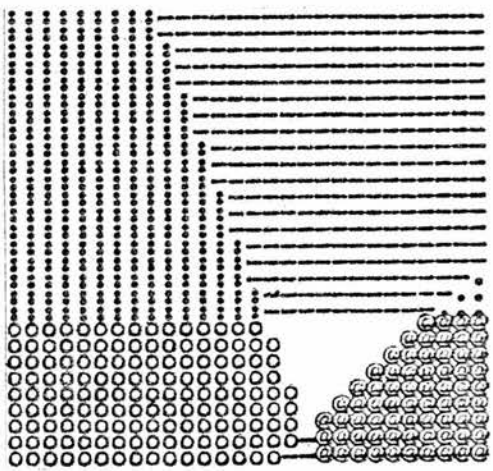
Points in a Plane



Lines in a Plane



Zones in a Plane



Representing Points, Lines and Zones as a Grid

Figure 5

of storage space in a computer. This limit imposes an upper level on the resolution of detail which can be achieved for any study area. The higher the dimension of the model the faster this limit is reached.

The second problem arises from the way that data is assigned to the grid. In the Boston Metropolitan Core study presented in the final section, the values associated with points on the grid were generally averages or maximum values taken from the zone immediately round the grid point. The topographic height values on the other hand were point values. To be able to relate spatial distributions in the same study area it is important to use the same sampling grid, otherwise it is necessary to know more about the real space from which the data are collected in order to know the correct way to interpolate values between the original grid values provided.

The limit imposed on the total size of a grid data set, by the amount of available storage space, where high resolution is required can be overcome by a variety of different data compaction techniques. The simplest example of these techniques can be seen in a display file created for line printer graphics. Where neighbouring values in a row of the array are the same, the row can be shortened by a simple packing procedure.

The string of values:

sssssbbbbbbbbbbbbbbbbbbbccddddddddddddddddddd

can be represented by:

5s,20b,2c,17d

In this form eight values are required to store 44 values. It must be

noted that this approach may not always result in a reduction of space requirements:

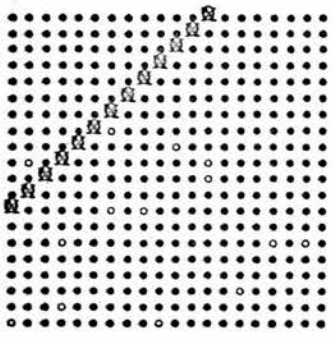
ababababab 10 spaces

1a,1b,1a,1b,1a,1b,1a,1b,1a,1b 20 spaces

but: 5(ab) 4 spaces

The price which has to be paid for this reduction in storage space is that the array accessing routine cannot be used to locate a point with this new data structure. Either the compacted data has to be expanded into an array and then the array accessing routine used, or the index location of the point has to be interpreted into a new storage address which matches the new data structure before it is possible to find the contents of a particular grid point. In both cases an extra operation is required to provide the same output obtained from the previous data structure.

Another form of data compaction can be achieved when storing an object represented in the grid data structure by storing the index locations of the points which make up the object. Taking a two dimensional space as an example, consider the line shown in Figure 6.



A Line in a Plane as a Point Set

Figure 6

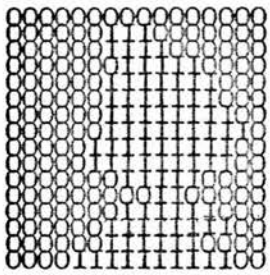
The whole array in Figure 6 would require 400 storage spaces. If merely the indexes of the positions of the points making up the line are stored: (1,8; 2,9; 3,10; ... 13,20) then the storage requirement is 26 memory cells.

When storing a point all that is needed are the two indexes of its array location. (If the size of the grid is known this can be reduced to one number). When representing the set of points which make up an area zone in this manner, the area should cover less than half the total number of cells in the grid for a reduction in the overall storage demand to be achieved.

If the area zone is represented by its boundary points, in other words the points making up the line which divides the total space into regions of inside and outside, then a more substantial reduction in storage demand will result. The way that this may be achieved is by taking each grid point classified as either I or O depending on whether it is inside the zone or outside it, and comparing it with its neighbouring cells. If all the surrounding cells are the same as the test cell then it can be ignored; if one of the surrounding cells is different then the test cell's location is output with its value, in this case I or O.

It has already been noted that the grid based model is closely related to the storage structure of most computers. A possible hardware implementation of the grid based spatial model has been simulated by the PAX system (Pfaltz, J.,1975). In this system grid maps are constructed

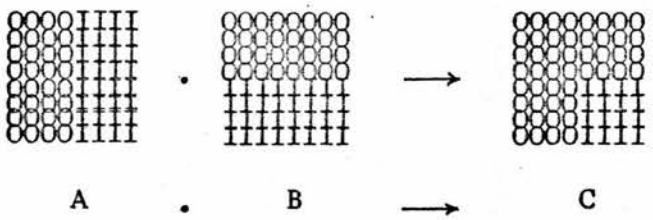
at the bit level in the machine so that 16 words of 16 bits each can store the array shown in Figure 7.



Bit Plane Representation of Aerial Zones

Figure 7

The hardware implementation means that logical operations can be carried out on these bit patterns so that the area of overlap for two distributions can be calculated by anding the two sets of bit patterns together as shown in Figure 8.



Pax Plane Operations

Figure 8

If each of the PAX planes shown in the example above is given a name A, B, and C then the operation defined by the expression:

$$C = A.B$$

can be interpreted into the spatial operation illustrated in Figure 8 The important aspect of this action is that the boolean expression

is itself a data structure which can be operated on in a meaningful way without necessarily having to interpret the names in the expression into their associated bit patterns.

It is this property which makes it possible to use other algebraic expressions as building blocks for spatial models. Given two points in a plane it is possible to interpolate points between them on a straight line by using the parametric equation:

$$x = p + k(q-p)$$

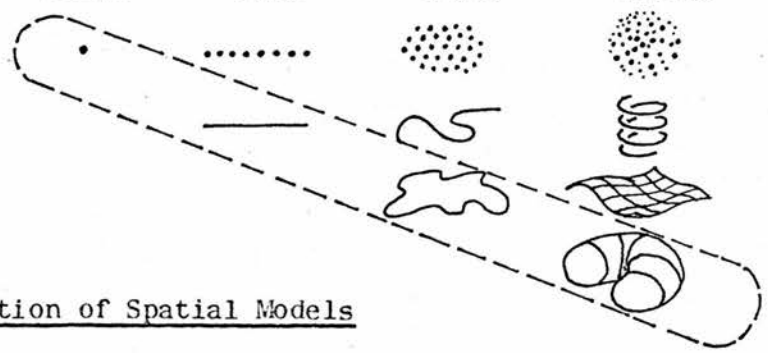
where x is the new point, p, q are the original points and k is a parameter which can be set to any value between 0 and 1. When k=0 x=p, when k=1 x=q. For values between 0 and 1, k will generate a point between p and q on the straight line through them. This makes it possible to use the two points p and q as the representation of this straight line segment. The intermediate points can be calculated whenever they are required. More than this the form of the equation of this line has the properties that if two line segments cross then the intersection point can be calculated using the relationship:

$$p_1 + k(q_1-p_1) = x = p_2 + m(q_2-p_2)$$

to give the value of k and m necessary to determine x. In other words, by fitting the values of p_1, q_1, p_2 and q_2 into the appropriate procedure in the correct order the value of x will be output. There is a variety of algebraic forms which can be used in this way, for example the straight line in the previous example could be expressed in the form $a.x + b.y + c = 0$. In which case, to define a particular line three numbers for the coefficients a,b,c would have to be provided. The

intersection point of two lines defined in this way can also be found by an operation on the ordered sets of their coefficients. In the previous forms of compaction for line data, based on a grid, the intersection point of two lines could only be found by reinstating the grid. The important development in the use of line equations is that this form can itself be used in operations to create new data.

Entities such as the coordinates of points - whether expressed as grid indexes or in real numbers - coefficients of plane equations, and other more complex structures become data objects in their own right when they form the basis for higher level operations. In fact the operations which can be carried out on different data objects can provide a variety of classificational schemes for different spatial models. Data objects which can be processed in the same way belonging in the same class. The initial properties which were used to differentiate types of spatial model depended on the dimension of the 'containing space'. In Figure 9 , it can be seen that points in two dimensional space are different from those in three dimensional space because the coordinates used contain two and three numbers respectively.



Initial Classification of Spatial Models

Figure 9

In fact this scheme, except as a very loose classification for descriptive purposes, was not found to be very useful. There appeared to be many other equally valid ways of dividing spatial models based on different procedures. In the next section though the investigation is carried out principally under the heading of volumes - three dimensional structures within a three dimensional containing space - because the data structures can equally well be used to define zones in a plane as zones in three dimensional space, a better classification seemed to be 'Zones in Space'. In terms of the data objects shown in Figure 9 this grouping covers the elements in the main diagonal of the table. It can be seen, however, that whatever grouping is taken the results are not altogether satisfactory, at this level of generalisation.

Another initial distinction which was made was between spatial data and non-spatial data. This division was again made on the basis that the operations which would be carried out on these two data types, would be different. For many applications the implied idea that the description of spatial location was independent of the description of properties which would be associated with a location worked very well. For a grid map the spatial component was treated directly by matching a storage space with a point in the area being mapped, while the property data was represented by the value stored in this storage space. However, a blurring of this classificational distinction appeared when the map shown in Figure 10 was considered.

In Figure 10 the hill shading which must be considered as non-

locational data if the approach set out above is adopted, is clearly not independent of the 'spatial' or locational data.

Hill Shading [1]

Figure 10

It seemed the only way that these ideas could be developed was to examine in more detail the problems of programming operations on spatial data. The initial classification was not discarded, but it was clear that it could only act as a loose envelope in which to conduct further study. The first stage was to examine the problems associated with representing and using geometrical entities as data structures. It was felt that if a better classification could be achieved it would emerge as this technical work progressed. The

1. Map from Dissertation: Brassel, K. 1973.

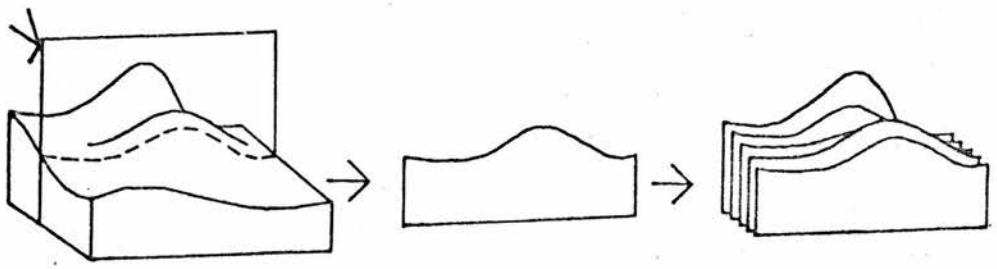
first goal was to create technical drawings. This meant starting with data which could represent objects in space, and at the same time be used to drive line drawing machinery. The study, presented in the next chapter, discusses the development of programs to produce block models of a topographic surface as line drawings.

CHAPTER 6

GEOMETRICAL OPERATIONS PRELIMINARY STUDY OBLIX

GEOMETRICAL OPERATIONS PRELIMINARY STUDY OBLIX

The first exploratory study completed in 1969, which used computer models of spatial relationships, resulted in the development of the OBLIX routine. The original aim of this study was to extend the graphic capability of the SYMVU program which was developed by Frank Rens in the Laboratory for Computer Graphics in Harvard University's Graduate School of Design. The original intention of the study was to modify SYMVU to permit surface contours to be drawn onto block model drawings. However, the program was difficult to adapt in the required way because of the range of control options which it already included. Consequently, the first stage in this study was to create a routine for hidden-line removal in profile drawings which could also be used with a contour generating algorithm. The advantage of starting from first principles was that it provided a chance to examine options available for implementing the geometrical operations which were needed for later studies.



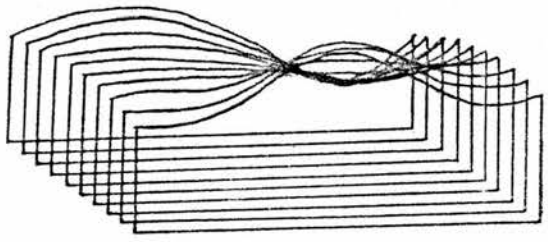
Profile Block Models: The Basic Principle

Figure 1

The construction of block models using profile-sections is conceptually simple. If a three dimensional surface is cut by a set of parallel planes and the intersection of the block with the planes is considered as a set of cutout cards, then if these card sections are superimposed on

each other, the one furthest away from the viewing position taken first, the result is a physical model of the surface in two dimensions. The hidden edges of the profiles are removed by the nearer sections being superimposed on them as shown in Figure 1.

The first stage in the program produced these profile sections as they would appear to the viewer. This was done by defining the boundary of a section as it would appear projected on the picture plane. If these boundaries were merely drawn out in their correct positions on the picture plane, then the result without hidden line removal would be that shown in Figure 2. The second stage of the program had to locate those parts of the section boundaries which should be invisible and remove them.

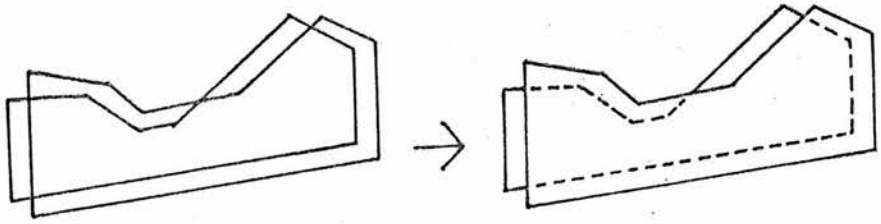


Profile Lines without Hidden Line Removal

Figure 2

The hidden-line removal in this case was carried out by processing section boundaries in the reverse order to that adopted when overlaying the card cutouts. The nearest polygon profile was drawn first, because it could not be obscured by any other section. The next profile in order, moving away from the viewer, was then processed. Where this new profile line, as projected on the picture plane, lay outside the previously drawn section boundary, it would be visible. When it crossed

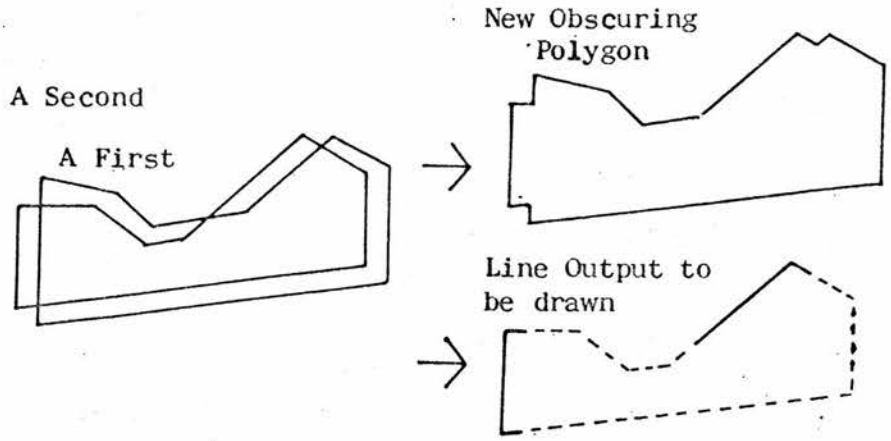
the previous profile boundary it would be hidden and therefore had to be removed as shown in Figure 3 . Once two profiles had been drawn in this way, the new obscuring polygon became the area covered by the two previously drawn profiles.



Hidden Line Removal for Profile Boundary Lines

Figure 3

The problem therefore was to create an algorithm which, given two polygons A-first and A-second, created the outputs shown in Figure 4

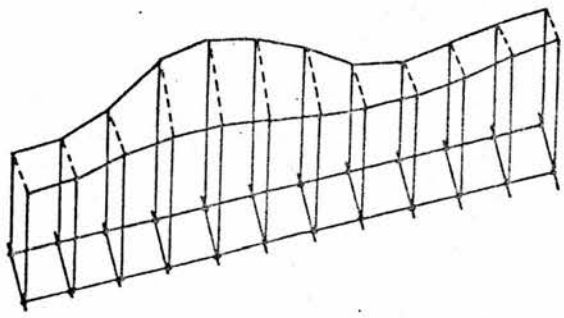


Hidden Line Removal Algorithm: Basic Principle

Figure 4

There were several simplifying restrictions adopted in the first experimental routine. To start with the profile boundary was, as in SYMVU, created from the row of height values taken from a regular grid

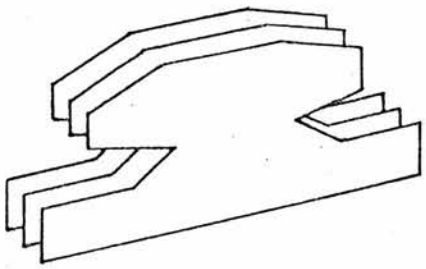
of data points. The first data sets used were produced as output from SYMAP. By taking the data values on a grid and scaling them vertically from the horizontal plane of the grid, the grid index (row, column) and the height value could be used as a three-dimensional coordinate. The row and column numbers had to be appropriately scaled by a row and column width, to create a final drawing but were adequate for internal calculations. The grid data automatically produced parallel, plane-section, profiles.



Profile Lines Generated from a Grid

Figure 5

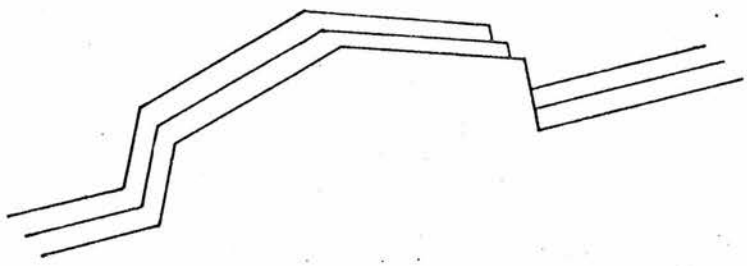
A profile created in this way could be treated as a simply increasing or decreasing height value 'y' for continuously increasing values of x (the row position). A re-entrant curve in the x direction of the form shown in the diagram in Figure 6 could not occur.



Complex Profiles

Figure 6.

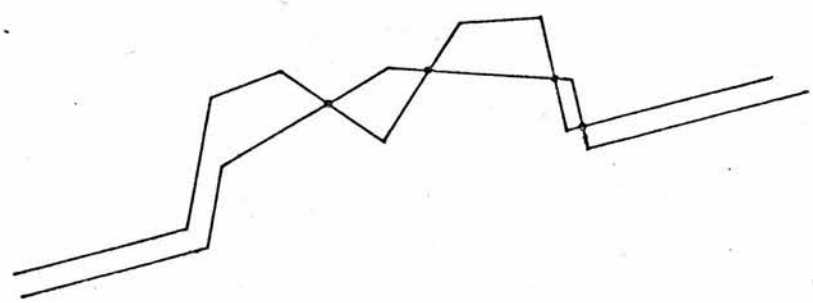
This property made it possible to remove the lower boundary of the section polygons to give profile lines of the form shown in Figure 7 These lines could be regarded as sections from an infinite loop.



Simple Profiles

Figure 7.

Each polygon was converted into an open boundary line shown in Figure 7. This meant that the profile lines could be tested from minimum x values through to maximum x values without backtracking. The first stage of hidden line removal became that of finding the intersection points in two lines.



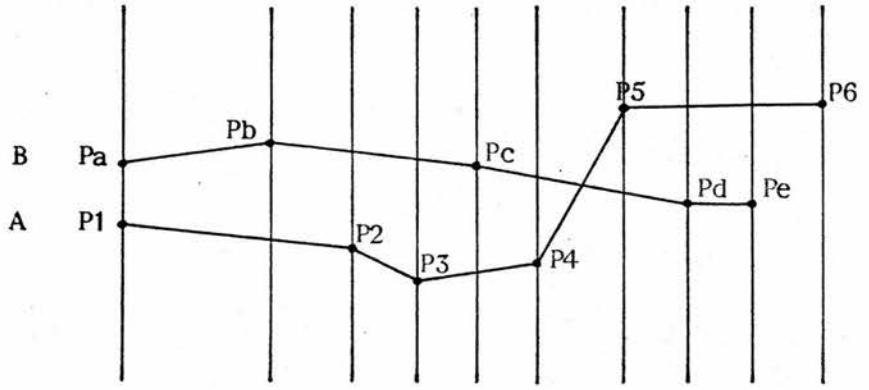
Line Intersections

Figure 8

Where there could be as many as 100 line segments in a profile line, it is clear that if each line segment in one line had been tested

for intersections with each line segment in the other line, a very large number of useless comparisons would have been made.

The strategy used to reduce the unnecessary work in finding line intersections can be illustrated by referring to the two lines in Figure 9.



Multiple Line Crossing Problem

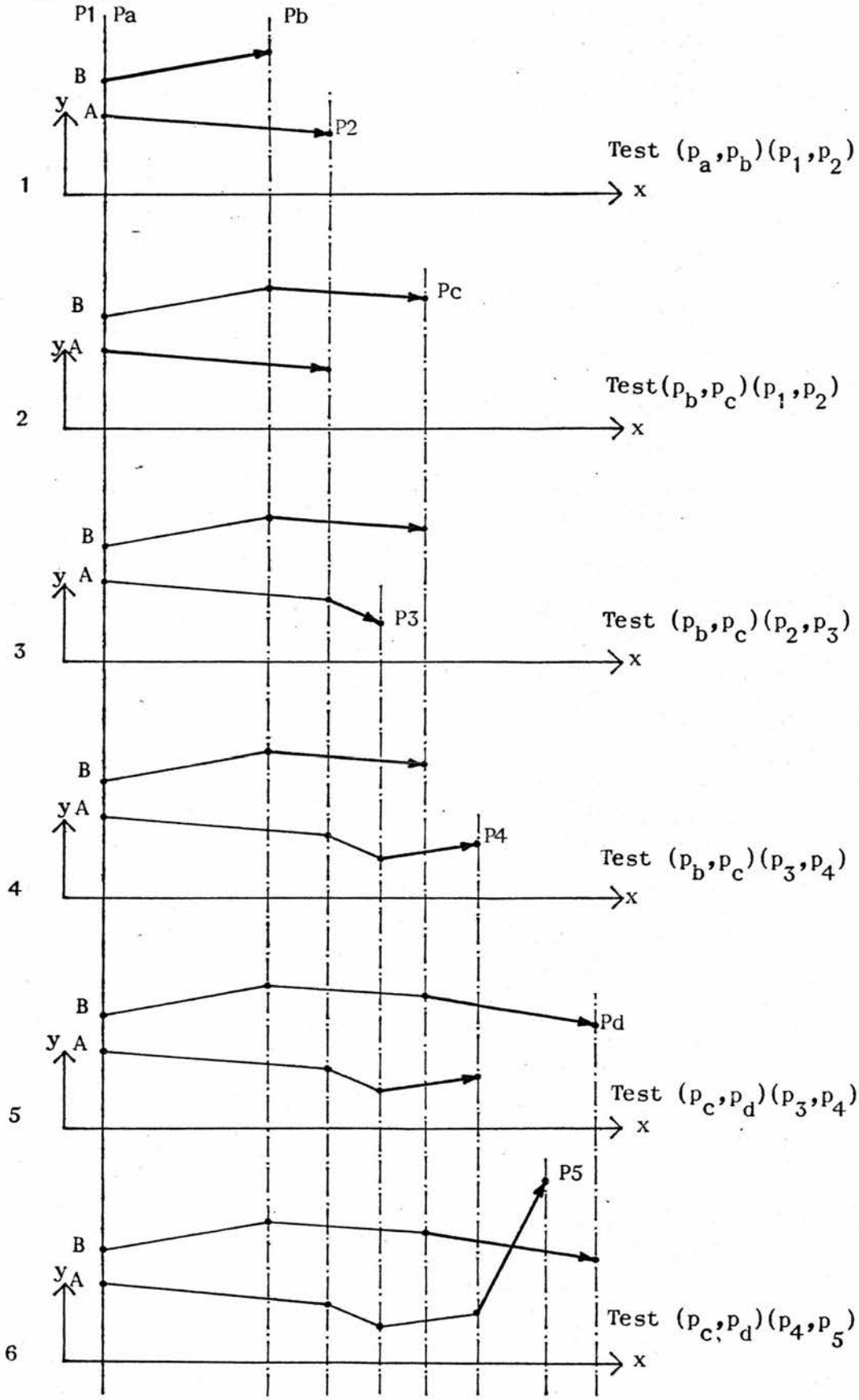
Figure 9.

In this diagram two lines A and B are shown which, for simplicity, have their two starting points p_a and p_1 with the same x coordinate. It is clear from the diagram that line segments (p_a, p_b) and (p_1, p_2) could intersect depending on the values of the y coordinates of the four end-points of the line segments, so these two line segments will have to be tested for an intersection point. However, p_b is before p_2 in 'x' order, which means that line segment (p_b, p_c) may also intersect (p_1, p_2) depending on the y value of the coordinates. The algorithm starts with the two coordinate lists of A and B:

A: $p_1, p_2, p_3, p_4, p_5, p_6$

B: p_a, p_b, p_c, p_d, p_e

GEOMETRICAL OPERATIONS: PRELIMINARY STUDIES: OBLIX



Line Segment Intersection Testing Strategy

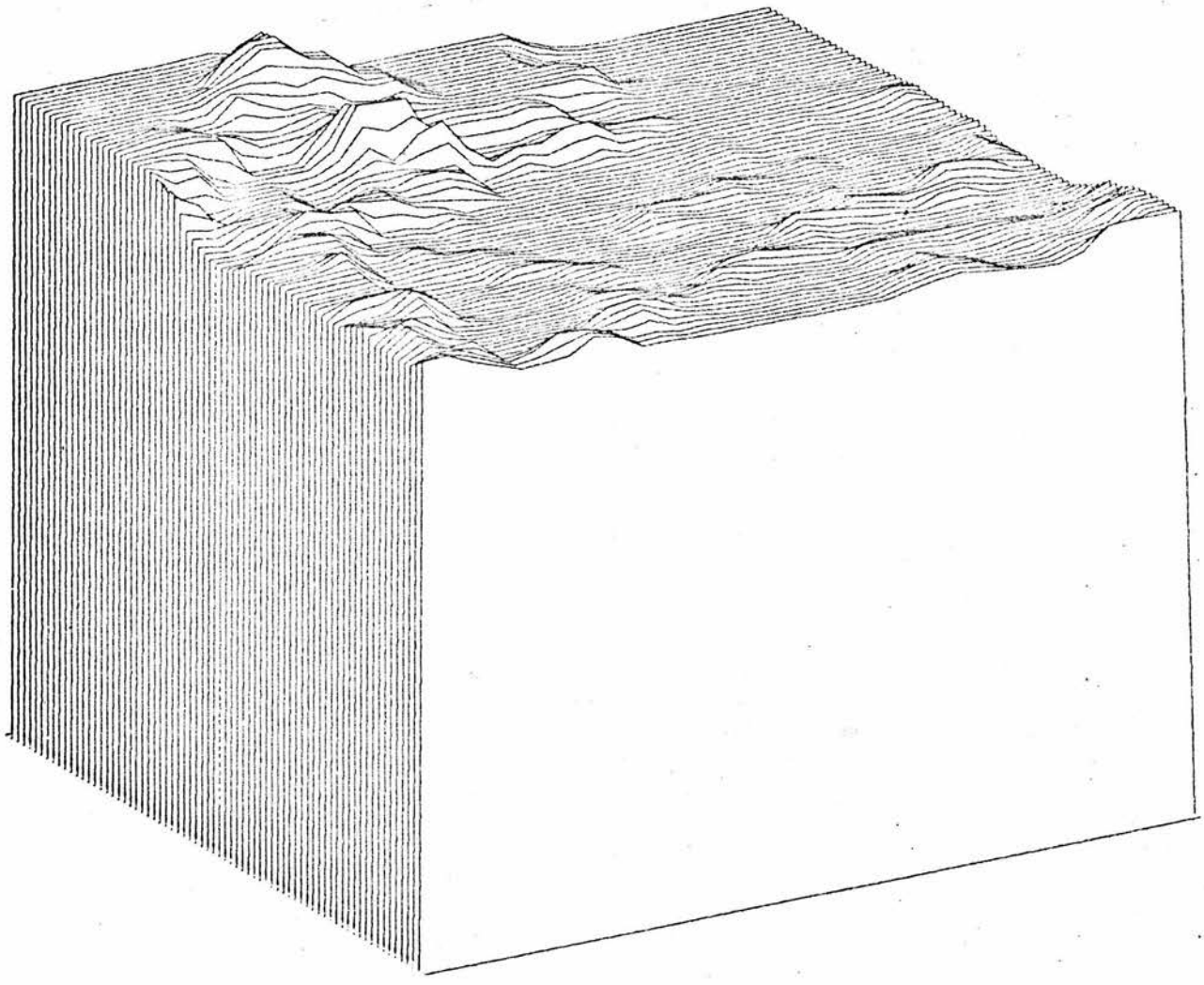
Figure 10

It takes the first pair of line segments from each list, it then compares the x coordinates of these line segments to see if they overlap in the horizontal direction, if not it increments to the next line segment in the list containing the segment furthest to the left (least x coordinate) until two segments are found which do overlap. In this example this first stage is avoided by making p_a and p_1 have the same x coordinate. When two segments are found which do overlap in the horizontal direction they are tested for a line intersection. The segment with the lesser x coordinate in its second point, that is the leading point, is then incremented and the procedure continued. For the example given above, the sequence of increments is shown in Figure 10.

This procedure makes use of the fact that the consecutive x values of coordinates in the profile increase in size.

The first implementation of this algorithm used a very primitive line crossing subroutine. What was needed was more than the calculation of intersection points, it was also necessary to keep track of which side of the obscuring boundary the current point of a new profile was on. If it was outside: in this case above or to the left of it, then it would be a visible point and would have to be output, firstly to be drawn, and secondly, to create the new obscuring zone boundary. If it was inside, then it could be ignored. In the first experimental program the first point of a profile was arranged to always fall outside. This meant that from there on by counting the number of intersections encountered, it was possible to know on which side of the line the current profile segment lay. One crossing or an odd number

GEOMETRICAL OPERATIONS: PRELIMINARY STUDIES: OBLIX



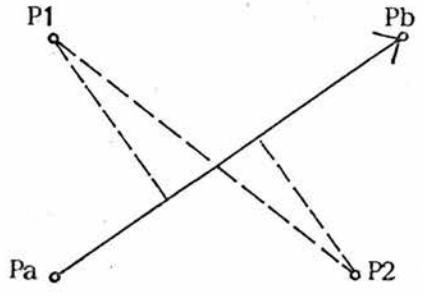
Profile Block Model

Figure 11

of crossing points meant that the current point was inside; two, or an even number of crossing points meant that it was outside. The difficulty with this procedure was that errors were cumulative. Since at this stage most of the errors resulted from special cases of line intersection, such as the intersection occurring at the end point of one line segment and the beginning of the next, it was necessary to improve this method, to get reliable results for debugging.

It was from a suggestion made by D. Cohen in the Computer Science Department of Harvard University, that this development was made. In principle, the two end points of one line segment were used to generate the coefficients of the line equation for the line passing through these two points, then the two end points of the other line segment were tested against this line to decide on which side of it they lay, see Figure 12.

This approach resulted in a procedure which was self-correcting. Where an unresolved special case caused an error, its effect was local and could easily be located when the resulting drawing was produced. Local errors could still create cumulative errors when the new obscuring polygon boundaries were created but the improvement provided enough evidence to make the location of errors a simpler task.



Line Crossing Test

Figure 12.

$a.x_1 + b.y_1 + a = k_1$

$a.x_2 + b.y_2 + c = k_2$

If $k = 0$ point lies on the line

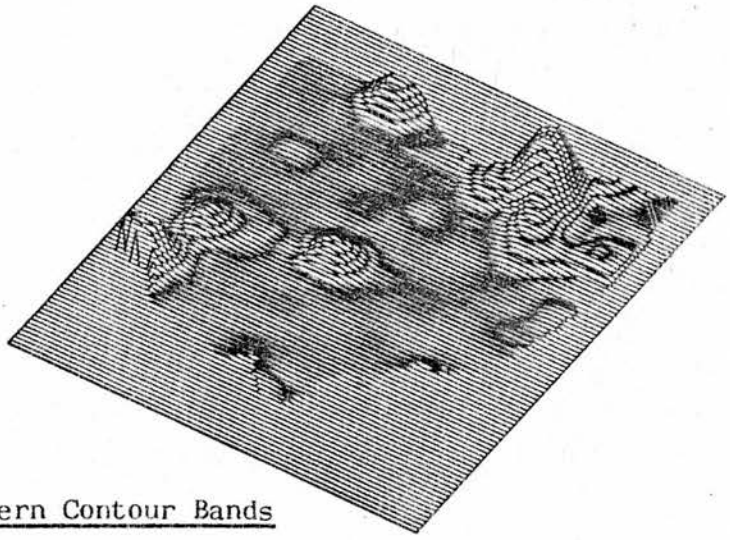
If $k > 0$ point is above the line

If $k < 0$ point is below the line

If $(k_1 * k_2 < 0)$ then p_1 and p_2 lie on opposite sides of the line through P_a and P_b.

GEOMETRICAL OPERATIONS: PRELIMINARY STUDIES: OBLIX

The first stage was completed with the output shown in Figure 11 The next stage was to create the horizontal contour lines. In fact, there were two problems to be solved at this stage. The first was to generate the contour lines, the second was to draw surface lines on the surface of the block model, with hidden portions of the lines removed. It became clear fairly quickly that these two problems were interdependent and had to be resolved together. An early experiment which only worked for a parallel perspective projection is shown in Figure 13 , where contour bands were created as moire patterns from overlaying an undeformed grid of lines on the profile drawing.

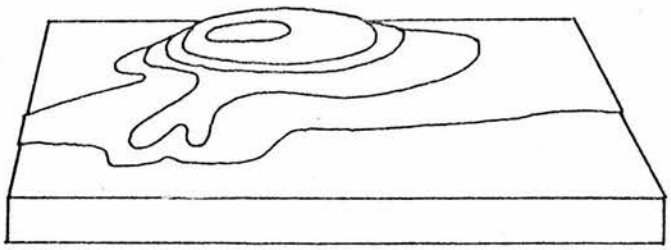


Moire Pattern Contour Bands

Figure 13

The surface contours, because they depended on the height values of the profiles, provided a convenient example of surface lines since they could be generated from the same data set used to generate the profile lines. In addition, it also allowed the general problem of applying surface lines, such as road networks to block model surfaces, to be investigated. The countour lines are in fact the same geometrical entities as the profile lines, in that they are the boundary lines of

section planes cut through the block model, horizontally instead of vertically.



Block Model from Horizontal Contour Planes

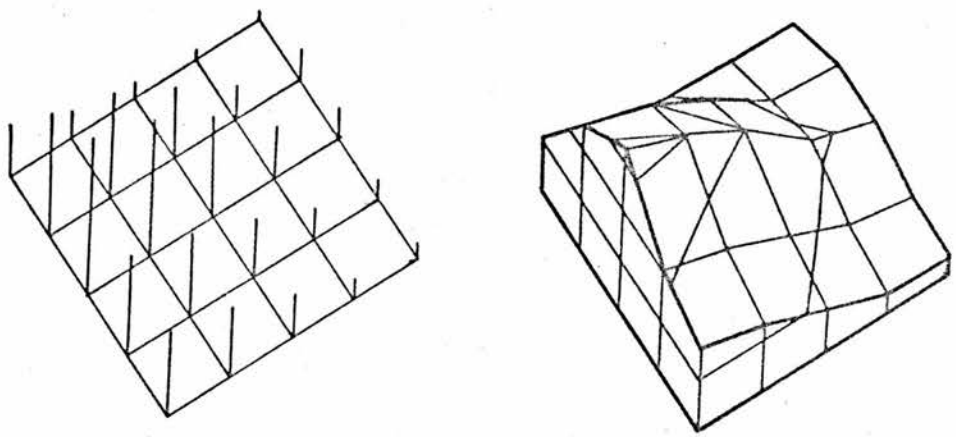
Figure 14

It can be seen in the diagram above, that the removal of the hidden portions of contour lines could be approached in the same way as the profile lines. By taking the contour lines nearest to the observer first, and then processing successive layers of contour lines in order moving away from the observer the task is that of creating successive obscuring polygons and drawing those portions of the next level of contour lines which fall outside them. It can be seen that in this case there would be no way of simplifying the obscuring polygon: it could occur as a collection of polygons at any particular level and each polygon could be any shape, though at one level the separate polygons would not overlap. The nearest contour level to the viewing point would be the one which lies furthest from the base plane, that is the highest contour level since the observer must be above the surface looking down on it.

This approach to the creation of contour block model drawings was not

attempted until later, when a general polygon overlay routine was constructed. The first stage was to create contour lines from the grid of data values used in creating the profile lines. The profile lines could easily be constructed from this grid by simple linear interpolation along a row or column. This was done by linking the vertical lines representing the point values on the grid, with straight line segments. The problem of creating the contour lines was more difficult since, at a particular contour level, the contour line would generally pass between the grid points, rather than through them.

If the original grid of data values is represented as the set of vertical lines scaled in height to represent the values of the grid points, then the general definition of the contour lines is the intersection of a horizontal contour plane with some surface which is interpolated between the data points as shown in Figure 15.

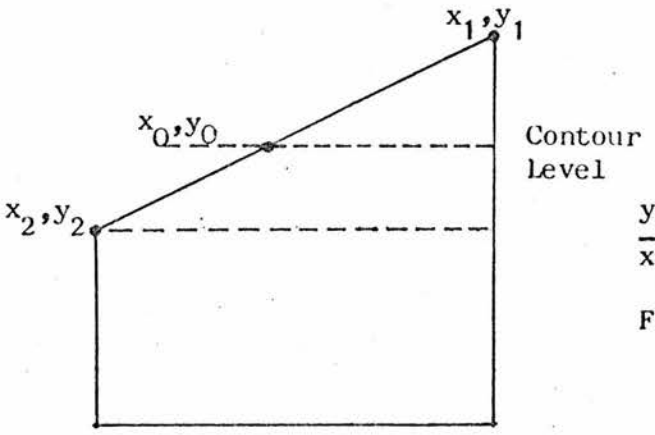


Interpolation and Contour Line Generation

Figure 15.

The position of a particular contour point could be located by simple proportion on either of the two sets of perpendicular profile lines

since they were made up of straight line segments.



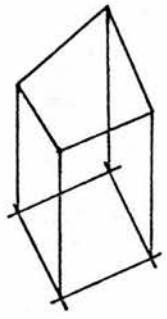
$$\frac{y_1 - y_0}{x_1 - x_0} = \frac{y_1 - y_2}{x_1 - x_2}$$

From which x_0 can be calculated.

Determining Contour Position on Profile Line

Figure 16.

The problem was to find the way to link these points on the profile to give the correct contours. The approach adopted was to take a cell of the grid representing the space between four data values.

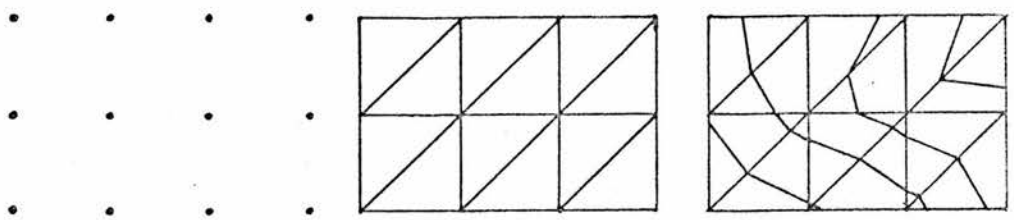


Interpolation Cell

Figure 17.

The profile line segments formed straight line edges to this patch of the surfaces and the problem was to find the simplest way of fitting a surface between these profile edges. The most direct way to do this was to insert a diagonal line across the cell, creating two triangular

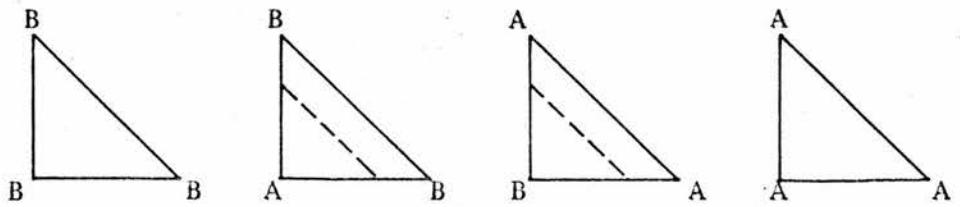
regions. Each of these triangles could be treated as a plane surface and the construction of contour lines would then be the intersection of the contour plane with these triangular patches.



Interpolation Using Triangular Patches

Figure 18.

The intersection of one of these triangular patches with the contour plane must result in a straight line segment which is a portion of the contour line. Because this segment must be a straight line it means that the boundary of each triangle can only intersect the contour plane twice. These points can be calculated as shown above. For a particular contour level the only way they can occur and the way they must be linked are shown in Figure 19.



A: Above the Contour Level

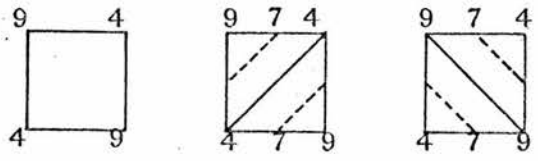
B: Below the Contour Level

Contour Line Patterns for Triangular Patches

Figure 19.

It is only the AB edges which are going to create contour line vertices.

A bookkeeping procedure to correctly link these points is relatively easy to implement. This solution was not the first one adopted, even though it was the simplest. The reason for this was that in the earlier experiments fairly large grid cells were used. It was found, in the situation where the pairs of diagonally opposite cell corners had high and low values respectively, that the linking of the contour points on the cell boundary lines depended on the diagonal used for triangulation.



The Choice of Triangulation for a Rectangular Cell

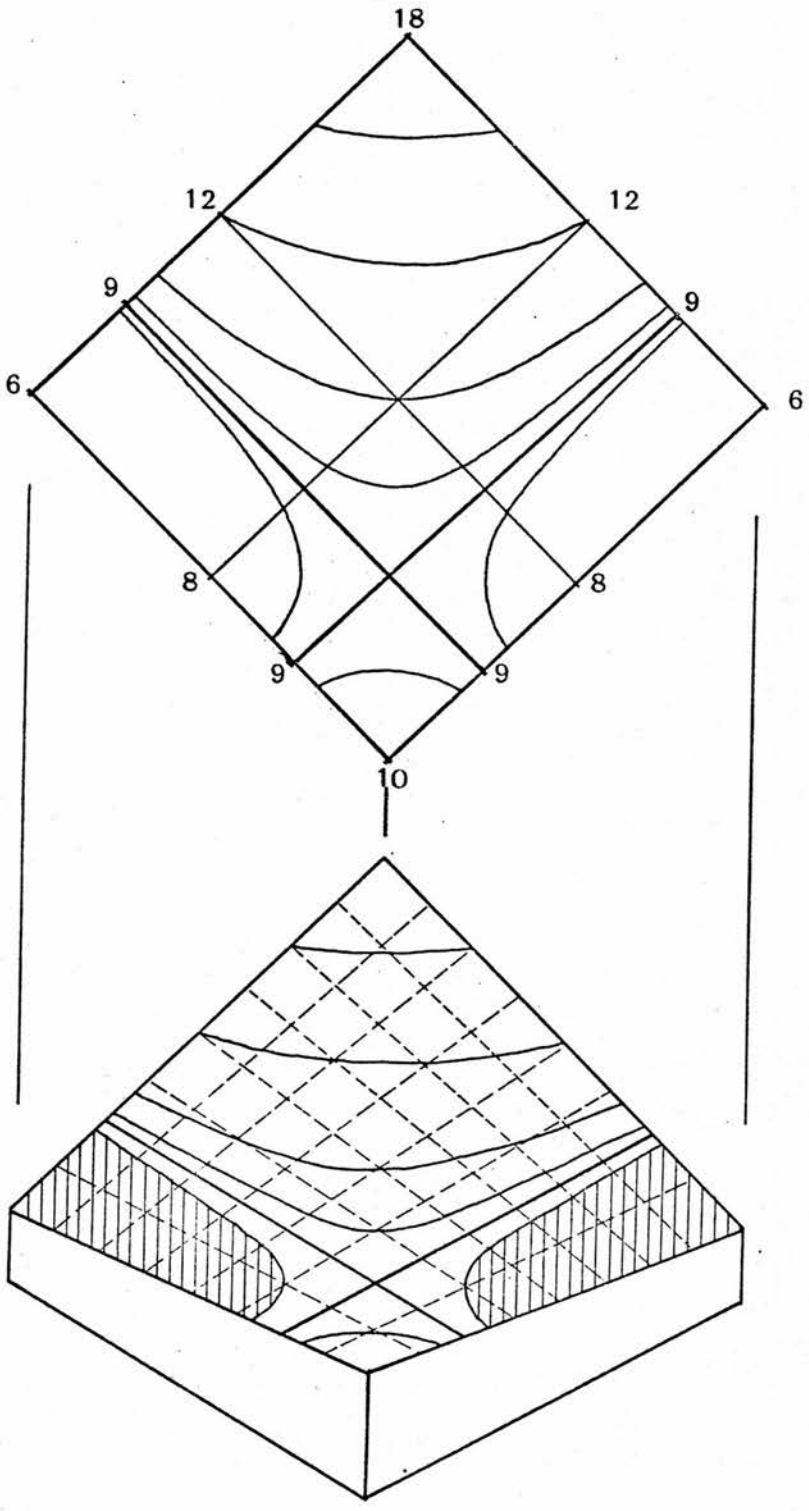
Figure 20.

It was felt that a symmetrical solution to this case should be developed, rather than one based on an arbitrary choice of diagonals. Such a solution was found by interpolating a single surface patch over the cell. The linear interpolation used to locate the ends of contour lines along the cell's boundary, when extended into three dimensions produces the hyperbolic parabaloid surface. Such a surface is uniquely defined by the four height values at the corners of a square or rectangular grid cell. What is more, the boundaries of the cell are straight line segments which lie in this surface. The contour lay out for this form of surface patch is shown in the diagram in Figure 21.

These contours are curves, being sections of a set of rectangular hyperbola. Figure 21 shows the way that such a surface patch is created by

GEOMETRICAL OPERATIONS: PRELIMINARY STUDIES: OBLIX

linear interpolation in two directions:

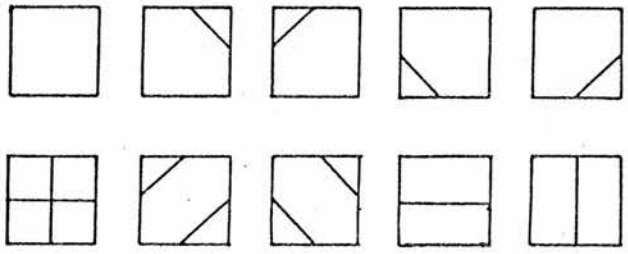


Two Way Linear Interpolation

Figure 21.

GEOMETRICAL OPERATIONS: PRELIMINARY STUDIES: OBLIX

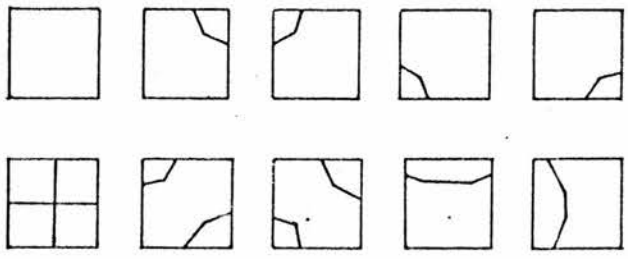
Where the cell size is small relative to the size of the whole block model it was possible to approximate these contour shapes by straight line segments. The simplest way in which this could be done was to calculate the points on the boundary of the cell at the contour level, as was done previously, and then link them in a pattern which was related to the hyperbolic parabaloid. There were ten basic line configurations which would be used for this linking process which are shown in Figure 22



First Order Contour Patterns for a Rectangular Cell

Figure 22

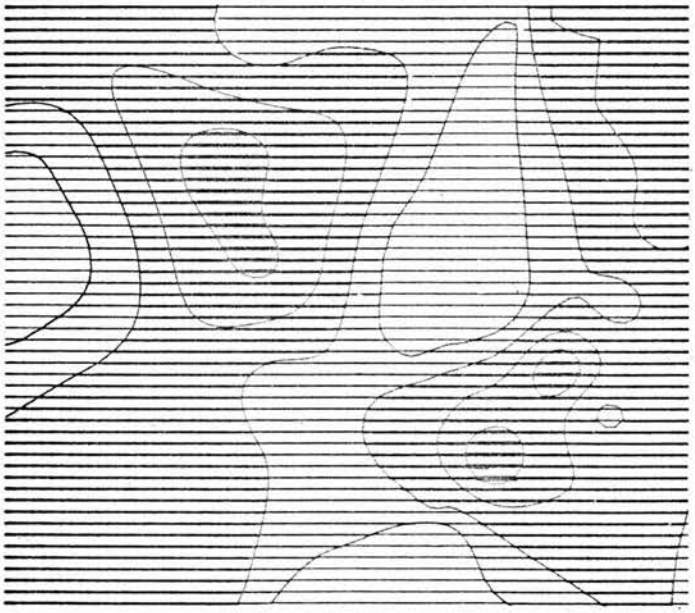
Where the cell sizes were relatively large, it was possible to use more complex pattern definitions for contour generations. For example, if contour points were interpolated on the two cell diagonals the contour line patterns in Figure 23 would result.



Second Order Contour Patterns for Rectangular Cells

Figure 23

The same effect could be obtained with perhaps a simpler linking procedure using the four triangular facets created by the two diagonals, the height of the centre point simply being the average value of the heights at the cell's four vertices. However, it seems that the simpler form given in Figure 22 is sufficient to create very smooth contours if the original data values are from a smooth surface sampled by a fine enough grid. The example shown in Figure 24 was generated using the first order contour patterns, from a surface created by the SYMAP interpolation algorithm (Donald Shepard, 1968).



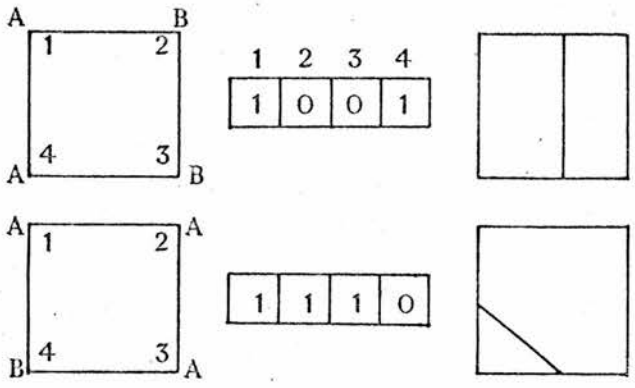
Contour Lines for a Smooth Surface

Figure 24

In the case of the simple straight line interpolation, the shaded areas on Figure 24 could be defined as either above or below the contour level. This meant that the ten line configurations could represent twenty different combinations of height values occurring at the cell's vertices. By always taking the corners in the same order, the combination of a particular set of corner values could be transformed into a pattern of

'aboves' and 'belows' represented by 1 and 0 as shown in Figure 25

This value interpreted as a binary number could then be used as an address to the stored linkage pattern which suited the combination it represented.



Selecting the Appropriate Contour Pattern

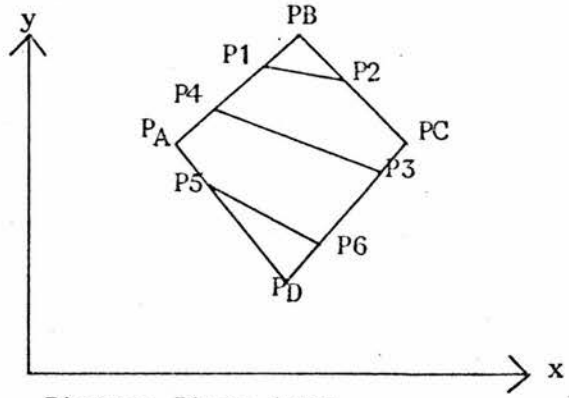
Figure 25

These stored patterns were very much like variable geometry characters: linear interpolation along the cell's boundary lines defining the actual shape the character would take in a particular situation.

Having created a contour line segment for a cell the process could be repeated for other cells. Since all the cells had to be processed to ensure that all the line segments of a particular contour level had been created, the cells were taken sequentially from each row, and the rows were also taken in their natural order. This created an unordered collection of contour line segments when the order in which they occur in a contour loop is considered, but this order was ideally suited to the way the hidden line algorithm worked. It was possible to select the order in which cells were processed to 'chase' a particular contour line

but this is an option which was left for a later study.

Once the contour generation algorithm had been successfully implemented, it became necessary to restructure the hidden line removal algorithm. The contour segments for a row of data cells could be drawn in at the same time as the new profile line, if they could be tested against the previous obscuring polygon, but the hidden line algorithm depended on the line incrementing strategy described above, which in turn depended on the direction of line segments being from left to right on the picture plane. If the projection of the data cell with contour line segments onto the picture plane is considered it can be seen that the new line segments are more likely to overlap in the x direction than not to, and that they are disconnected segments unlike the profile line segments.

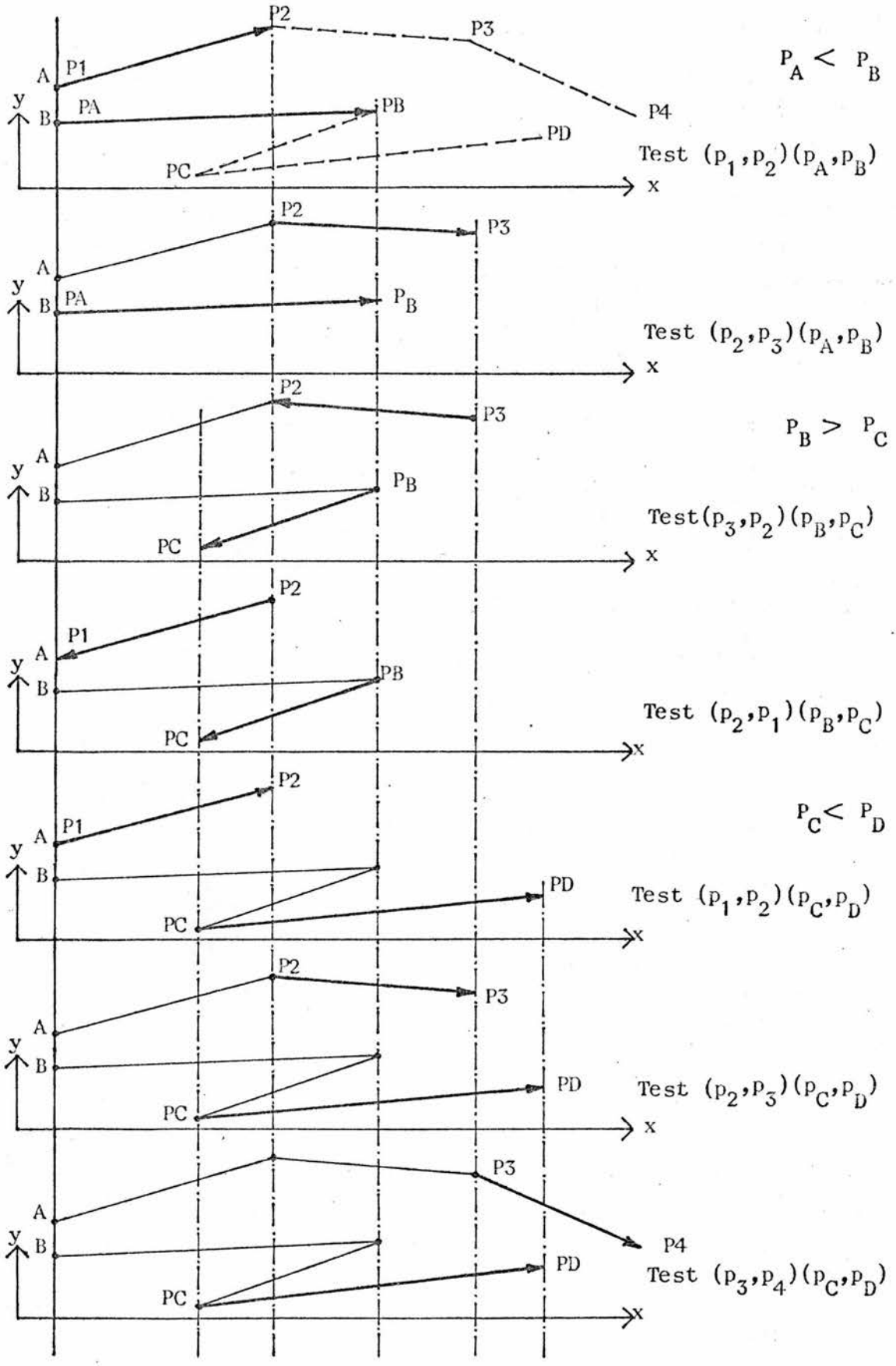


Picture Plane Axes

Contour Line-Segments, The Hidden Line Problem

Figure 26

These line segments could be compared with the obscuring polygon, but this would have to be done in a series of distinct steps. The pointer to the next vertex in the obscuring polygon boundary would have to be reset for each line segment. To make the task more difficult there



Modified Intersection Testing Strategy

Figure 27

was no way in which the direction of the line segments P_1P_2, ... could conveniently be arranged to always be the same, in other words that $x_1 < x_2$, so the hidden line algorithm had to be made to work for lines defined in any direction.

The method which was finally adopted was a modification of the previous strategy. Because the obscuring polygon boundary still conformed to the previous restrictions, and since only P_AP_B of the new line segments would affect the new obscuring polygon boundary, it was not necessary to use a general form of polygon overlay. By treating the line segments P_AP_B, P_1P_2, ... as a continuous line $P_AP_BP_1P_2P_3$... it was possible to use an incrementing strategy which was similar to the previous approach but was able to handle lines in the backward direction as well as the forward direction as shown in Figure 27.

Though it may not be altogether obvious from this limited example, this process was also able to restrict the number of comparisons for line intersections to a minimum. Where there are 100 or more cells in a line it can be seen that the comparisons would be localised to the area on the picture plans round the current line segment being tested.

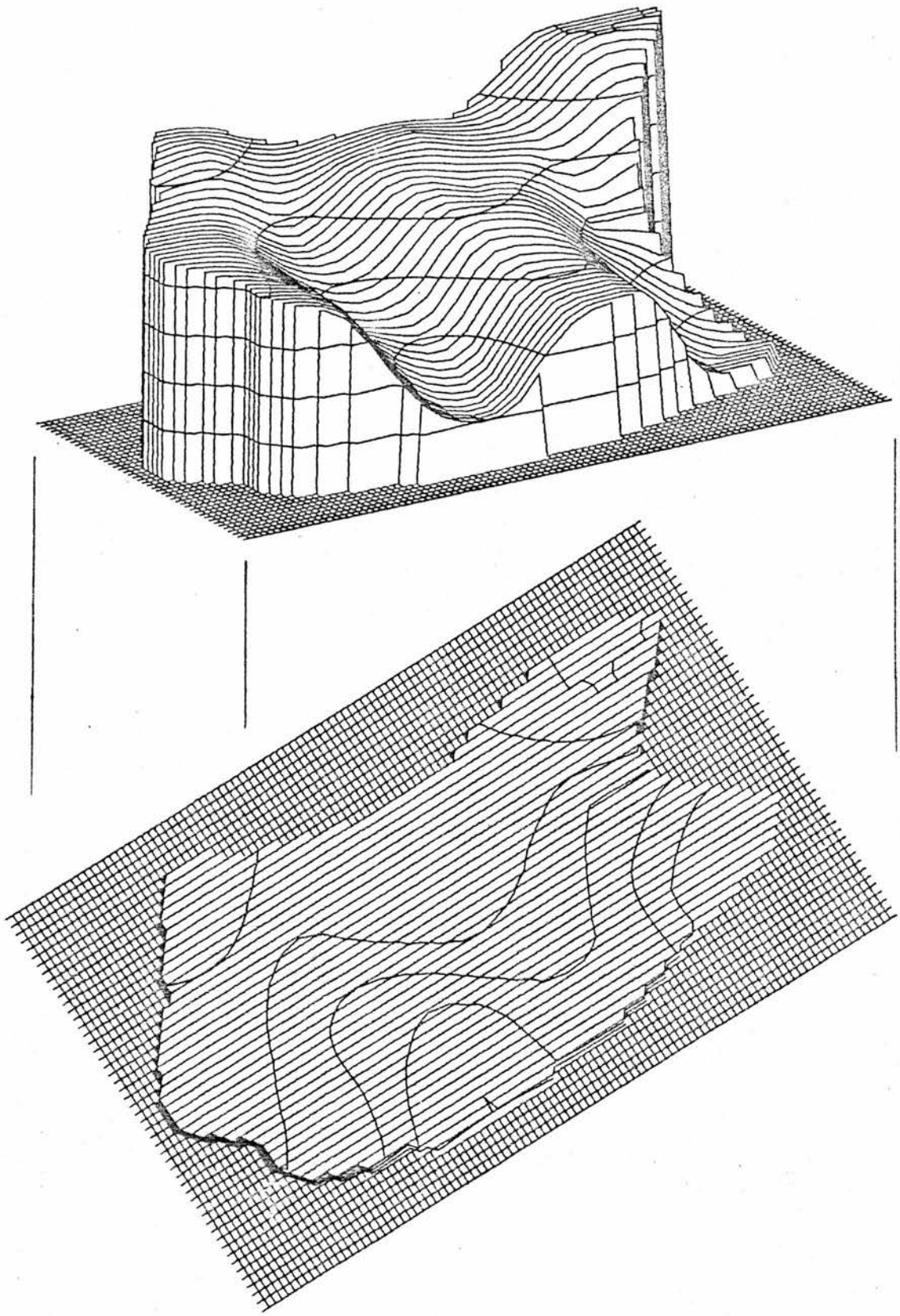
Treating the separate line segments P_AP_B, P_1P_2, P_3P_4, ... as the continuous line $P_AP_BP_1P_2P_3$... created the new line segments P_BP_1, P_2P_3, ... These new line segments were also tested for intersections with the obscuring polygon boundary. However, when it came to the drawing stage these line segments merely took the pen from the end of one line segment to the beginning of the next. They were consequently movement commands to the plotting device which had to be carried out with the pen 'up'. Because of the self correcting nature of the line

crossing test, it would have been possible to avoid the line crossing tests for these intermediate lines, merely using the incrementing process to set up the correct set of comparisons for the lines which would be drawn if they were visible. In the first implementation of this algorithm it was simpler to treat all the line segments in the same way. When it came to the drawing stage it was relatively easy to suppress the intermediate lines by making every alternate pen command a pen-up command.

After a series of test runs the first successful block model with surface contours was produced. By specifying different angles of viewing and using a parallel perspective projection the three quarter view and the plan view in the following diagram were produced. The data set used in this example was the output from one of the standard SYMAP teaching exercises, called Mantegna Bay. Following this illustration, the drawing of the hemisphere shows the hidden line removal in a more convincing way. In this example the data set was generated from the mathematical equation of a sphere.

The second block model has the contour lines drawn in as thicker lines than the profile lines. The development to the algorithm which permitted this to be done, required each line segment to be tagged by a name. These names could then be used to output the line segments representing different parts of the display to separate files. Each of these files could then be drawn in different line thickness or in different colours. This facility led to experiments with other forms of surface lines which could be applied to the block model displays.

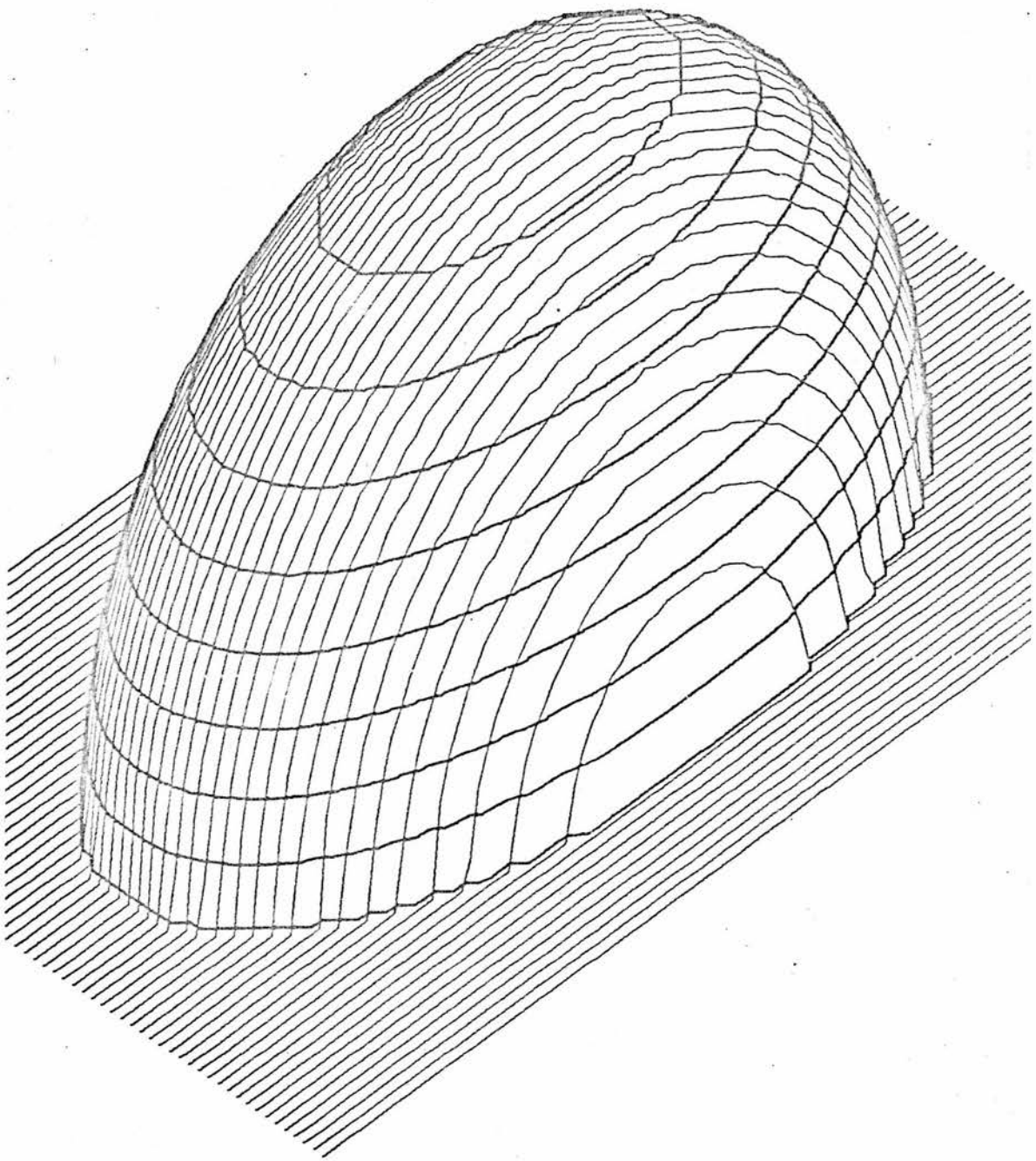
GEOMETRICAL OPERATIONS: PRELIMINARY STUDIES: OBLIX



Mantegna Bay Block Model

Figure 28

GEOMETRICAL OPERATIONS: PRELIMINARY STUDIES: OBLIX



Surface Generated from a Mathematical Function

Figure 29

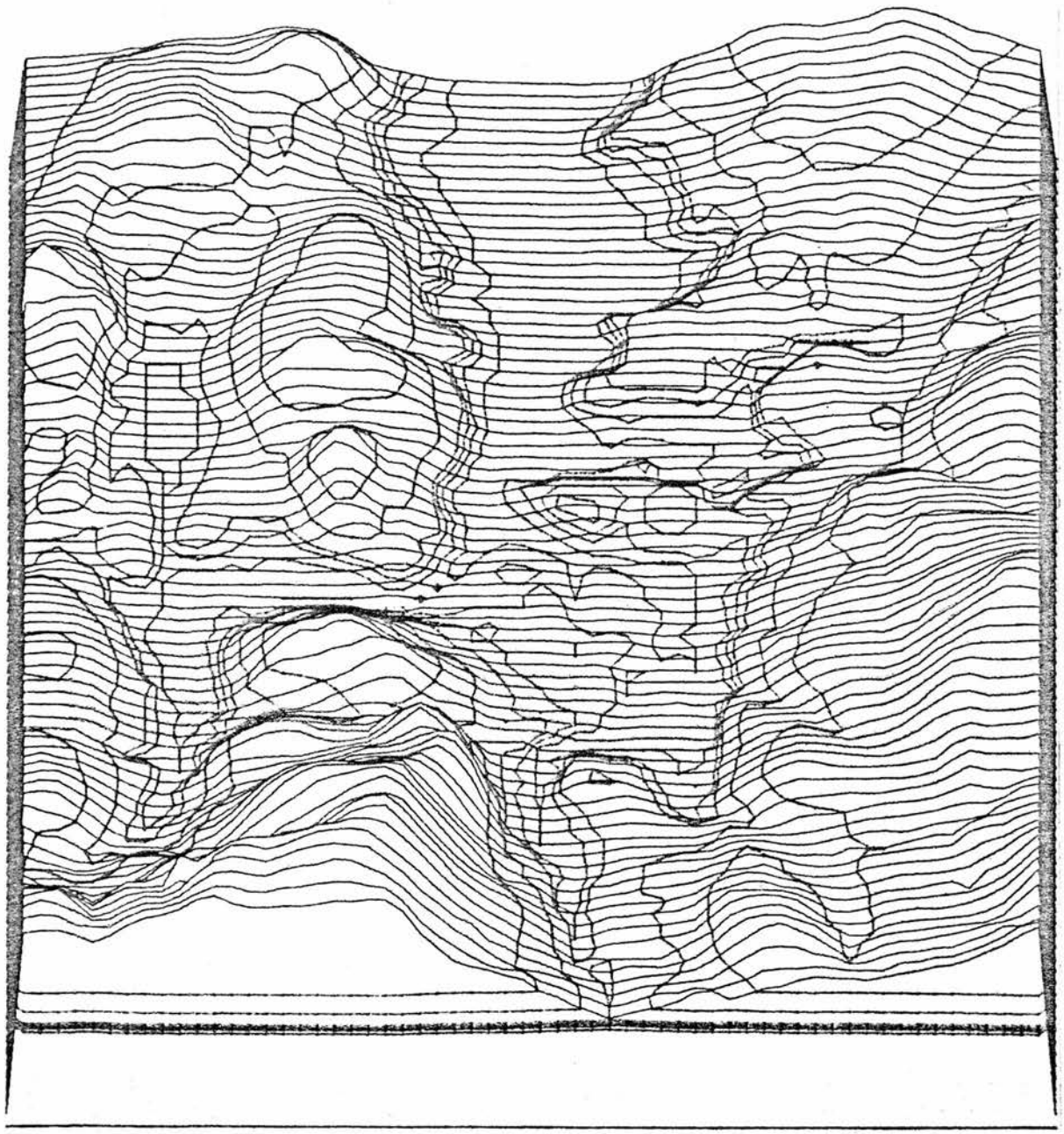
GEOMETRICAL OPERATIONS: PRELIMINARY STUDIES: OBLIX

Most of these early experiments were made using data prepared by Steinitz, Rogers and Associates in a conservation study being made of an area in Newhampshire around Honey Hill. A part of this study was to estimate the impact of a new dam proposed for the area. Consequently, the first series of drawings were block models of the area showing the contour levels which would mark out the level and incidently the extent of the water in the new reservoir. Three levels were defined which corresponded to the expected drought, normal, and flood conditions in the reservoir.

In the previous examples the contour levels were calculated in the contouring routine by dividing up the range between the maximum and minimum heights found in the data grid set into an externally specified number of equal interval contour bands. At this earlier stage, the aim was simply to improve the ability of a person using the block model drawing to locate a position shown on its surface which would accurately correspond with a map of the same area. However, when it became necessary to draw particular contour levels, as in the case of these water level contours, then it became necessary to read into the routine the required contour levels as data. This was simple enough to implement, and the examples of it can be seen in the drawings given in Figure 31.

The two block models show views taken from opposite ends of the valley in which the new dam was to be built. The contour lines which show the expected water levels can be distinguished from the remaining contour levels because they are more closely spaced. The other contour lines are standard equal interval contour lines at 100 ft. intervals and are included to indicate the surface relief more accurately.

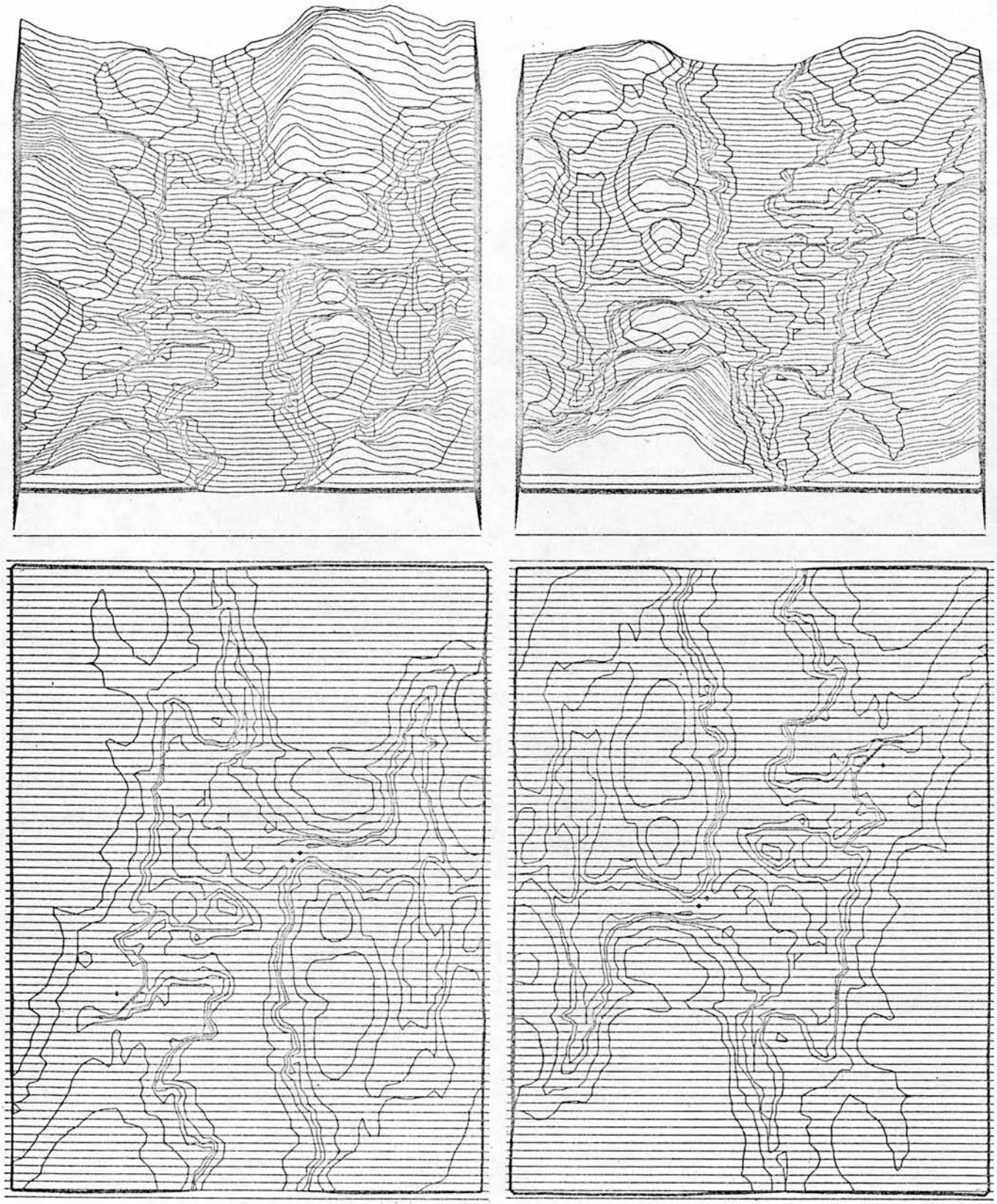
GEOMETRICAL OPERATIONS: PRELIMINARY STUDIES: OBLIX



Honey Hill, Original Size

Figure 30

GEOMETRICAL OPERATIONS: PRELIMINARY STUDIES: OBLIX



Honey Hill, Two Views of the Main Valley

Figure 31

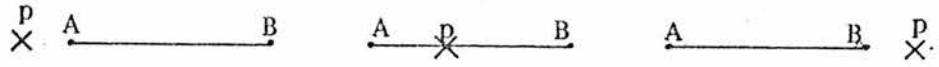
The plan view of the study area is also shown, and the water level contours can be distinguished more easily in this drawing. Each of these drawings was produced from the same data set: a grid of height values. In this case, these values were prepared by hand, laboriously taken from a published topographic map, rather than the result of an interpolation procedure as in the case of the previous examples. Once this data set had been prepared it could be used to generate new contour maps with different contour spacing and also block model drawings giving any specified point of view.

The next stage was to attempt an improvement in the plan view drawing. The drawing shown in Figure 24 shows the first experiment in this process. The profile lines were used to create a shading symbolism indicating height. These lines were divided into sections depending on which contour band they were traversing. Once this had been done and these lines stored in separate files, it was possible to use different pen sizes when they were being drawn to get the grading in tone, corresponding to increasing height. Similarly, it was possible to change the colour used for each contour band. This experimental drawing was created from a SYMAP data grid.

The profile lines used for hidden line removal were not divided up, but an extra line which was colinear with each profile line was generated for this surface symbolism. This procedure was adopted for convenience in creating the new obscuring polygon. Though it demanded a little more storage space and created extra computation it was a price willingly paid for a simpler programming structure.

The division of these shading lines for each contour band was modelled

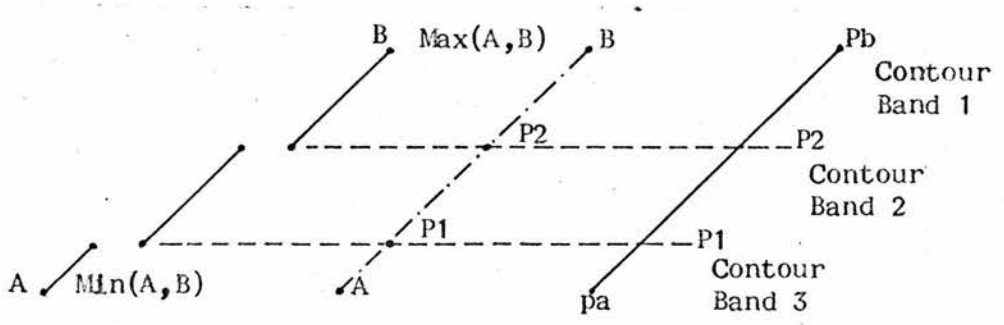
on the process used for contour generation. In this case the pattern of the lines was much simpler:



Profile Shading Line Patterns

Figure 32.

A contour level would either fall to the left of AB, within AB or to the right of AB. The problems arose when a line segment was steep enough to contain more than one contour break. This possibility meant that all shading lines had to be tested against all the contour levels. To reduce the computation necessary in these tests it was necessary to store the contour levels in ascending order. This meant that where automatically generated, equal interval contour levels were to be used with specifically defined groups of contours, the two lists had to be merged to give a single, ordered list of levels.



Shading Line Segments for More than One Contour Band

Figure 33

Starting with such an ordered list of contour levels (P_0, P_1, P_2, P_3) in Figure 33 then the lower of the two end points of line segment AB

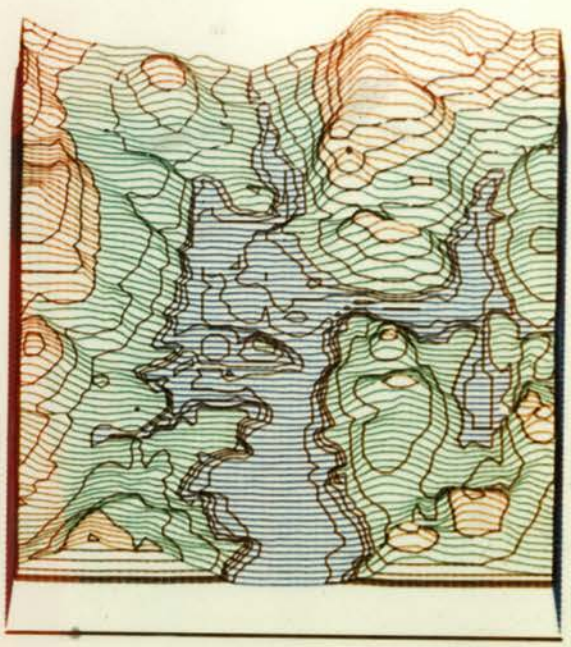
must be tested against these levels in order, until a level is found greater than the value of Min(A.B). Each successive contour level is then tested against the greater value of A or B to make sure that it is less than or equal to Max(A.B). Each contour level which satisfies these two tests is entered into a list of output points twice in order to create a list of separate line segments. Each point in this list is tagged by a name associated with the appropriate contour interval, for later sorting. It became apparent that the contour levels used to create this form of line shading need only be a subset of the levels used to create the contour lines required on the final drawing. As a result the levels defining these contour bands were entered as separate data for this section of the routine.

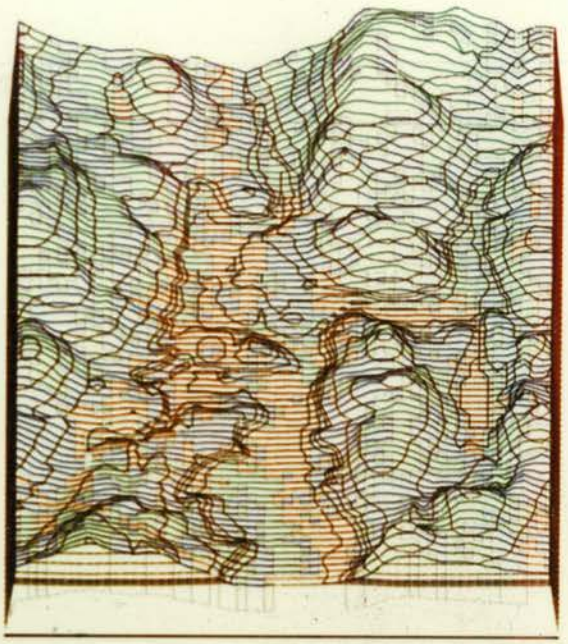
It became clear at this stage that the combination of different options which this program could provide would not always be compatible in a simple system. This became clear in these experiments when simple but basic changes had to be made to the order of processing, to obtain the different forms of output. It seemed premature at this stage of development to develop a complex control system until a wider range of possibilities had been investigated. It was only later on in this investigation that a system structure which could provide the user with a flexible but not too complicated way of controlling these different options emerged. Though a SYMAP type of control structure was investigated, it did not appear to give the versitility required. What finally appeared possible was a control language whose parsing and semantic analysis could be used to set up efficient program structures necessary for the different options.

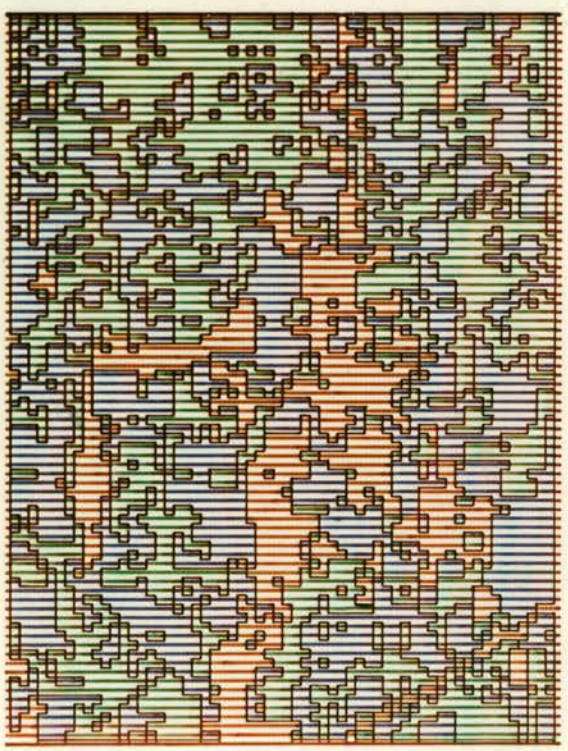
GEOMETRICAL OPERATIONS: PRELIMINARY STUDIES: OBLIX

The two left hand diagrams on the next page show coloured shading symbolism used on the Honey Hill data. In these contour drawings only three levels of shading were used. The lowest, in blue, shows the area which would be flooded by the proposed reservoir. The next level in green shows the areas of intermediate height. The final level in brown picks out the hill tops in the area. Colour was used rather than line thickness to distinguish these different zones, so that the plan view and the block model could be presented in the same form. If different line thicknesses had been used on the block model the changes in tone which would have resulted, would have competed with and destroyed the tone changes which the line spacing created and which was essential in providing the impression of plastic modelling in the surface's form. In fact, several drawings were produced in colour before a combination was found which was successful. It was found that the tone of the coloured inks had to be kept the same, and it was only the 'hue' which could be varied if the block model was to give the best results.

The two right hand diagrams and the frontpiece of this thesis show the final set of experiments carried out on the Honey Hill data. In these two drawings the aim was to show in the same diagram the effect that the new reservoir would have on the existing tree population. The tree distribution was represented by a coded grid, the data points used for height values and tree coding being the same. Each of these data points was classified by the predominant tree type surrounding it, from aerial survey data. The classification used in this example contained three classes: predominantly conifers, predominantly deciduous trees and, finally, predominantly open space.

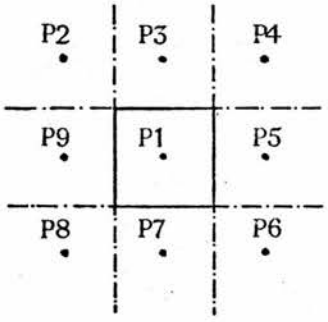






GEOMETRICAL OPERATIONS: PRELIMINARY STUDIES: OBLIX

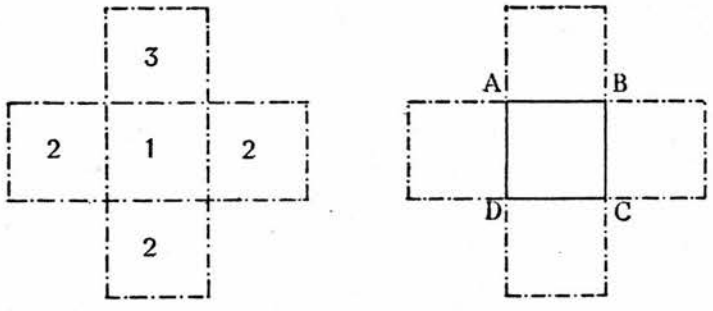
The first problem was to define the boundary from the grid data of homogeneous areas defined by this classification. The area represented by each data point was a cell in a grid formed by the lines which were the dual graph of the grid on which the data points lay. Another way of defining these cells was as the nearest neighbour polygons to their respective data points as shown in Figure 34.



Data Zone Boundaries: Dual Grid to the Data Point Grid

Figure 34.

If such a cell is the neighbour of other cells with different classifications in Figure 34. P1 is neighbour to P4, P5, P7 and P9, then it will be an isolated region, and will require the boundary lines AB, BC, CD, DA as shown in Figure 35.



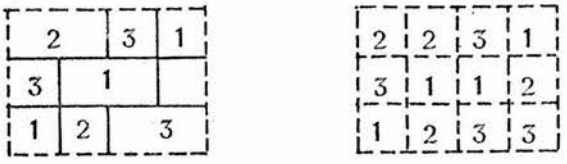
Zone Boundaries between Different Data Values

Figure 35.

Where two adjacent cells across a boundary have the same classification,

no boundary line is needed between them.

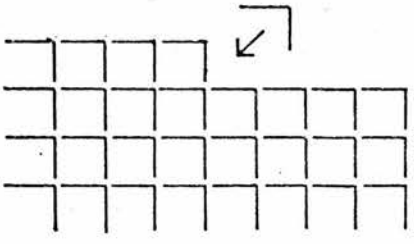
So, by testing the code value for one cell with the code value of an adjacent cell it is possible to decide whether a boundary segment needs to be created. By carrying out this process for all data cells the boundary lines round homogeneous regions will accumulate, correctly and consistently: Figure 36.



Accumulation of Boundary Segments

Figure 36.

Where the cells are being processed in their natural order in the matrix of grid values, this comparison need only be carried out for the two outside edges of each new cell, see Figure 37.



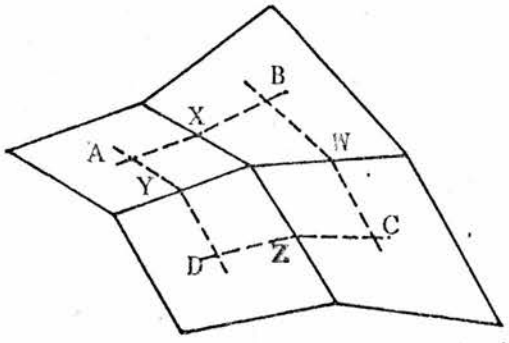
Accumulation without Duplication

Figure 37.

Such a comparison only required two rows of the data matrix representing the tree distribution in core storage at any one time.

However, these boundary lines had to be created in two segments since

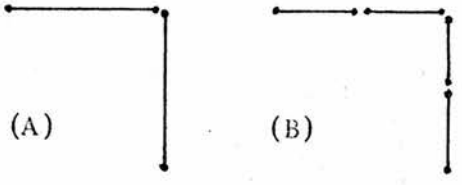
they lay on the surface of the block model. This was because they lay on the dual grid to the one used in defining the boundaries of the interpolation patches, used in the contour generation procedure, and the relationship between the two grids is shown in Figure 38 as a projected drawing.



Three Dimensional Relationship between the Two Grids

Figure 38

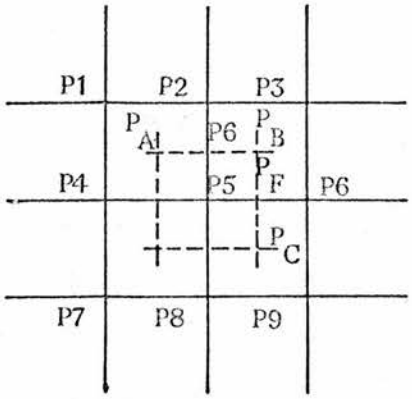
Consequently, the centre point of the patch used for interpolation is the vertex for the 'tree' grid cell. This means that what appears on plan as a straight line boundary for example AB crosses from one interpolation patch to another. Since the line AX is parallel to the side of the interpolation cell it will be a straight line segment even though it lies in the hyperbolic paraboloid surface. Similarly, XB will be a straight line segment; however, in three dimensions AX and XB will not be the same straight line. This means that the boundary line pattern (A) has to be created as (B)



Boundary Segment Patterns for Two and Three Dimensions

Figure 39

The three dimensional coordinates of the end points of these line segments can be obtained from the height data matrix in the way illustrated in Figure 40.

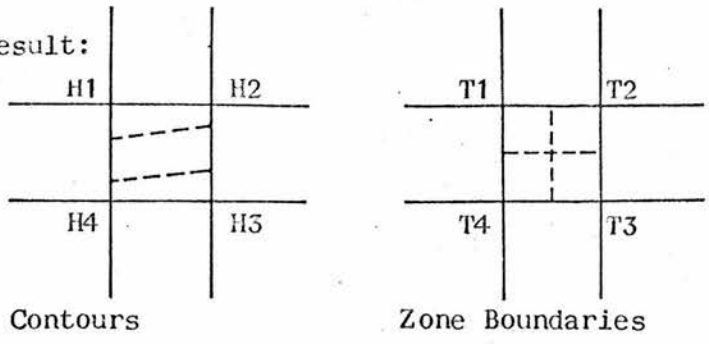


Calculating Vertex Coordinates by Linear Interpolation

Figure 40.

$$P_A = (P_1 + P_2 + P_3 + P_4)/ 4$$
$$P_E = (P_2 + P_5)/ 2$$
$$P_B = (P_2 + P_3 + P_6 + P_5)/ 4$$
$$P_F = (P_5 + P_6)/ 2$$
$$P_C = (P_5 + P_6 + P_9 + P_8)/ 4$$

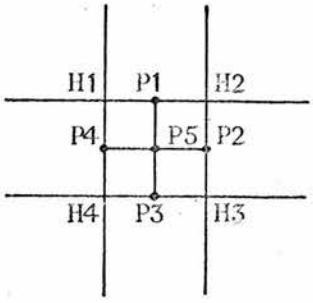
This, however, would require three rows of the height data matrix to be in core at the one time. Since four line segments may well have to be generated for each new cell, the approach shown in Figure 41 gives a simpler result:

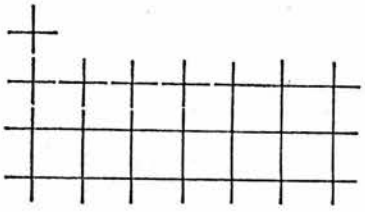


Line Segments Generated within the Contour Cell

Figure 41.

The line segments required for boundary lines are created which occur within the cell used for generating the contour lines.





Accumulation of Zone Boundary Patterns

Figure 42.

This requires less storage space and the three dimensional coordinates required for the end points of these line segments can be calculated in the following way:

$$P_1 = (H_1 + H_2)/2$$

$$P_2 = (H_2 + H_3)/2$$

$$P_3 = (H_3 + H_4)/2$$

$$P_4 = (H_4 + H_1)/2$$

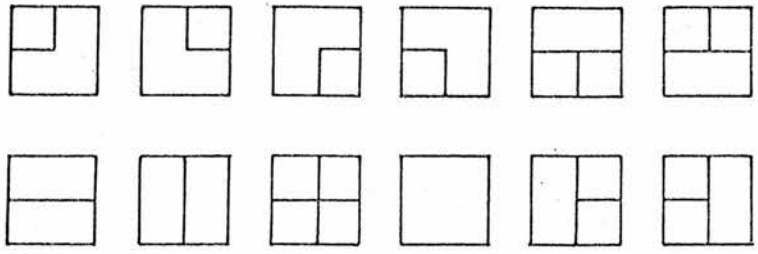
$$P_5 = (H_1 + H_2 + H_3 + H_4)/4$$

These points are simpler to calculate in this grouping than the previous grouping.

The choice and linkage of these boundary lines was achieved in the same way used for the contour lines. The basic linkage patterns shown in Figure 43 were placed in storage.

Again, it was necessary to tag the lines for sorting into different

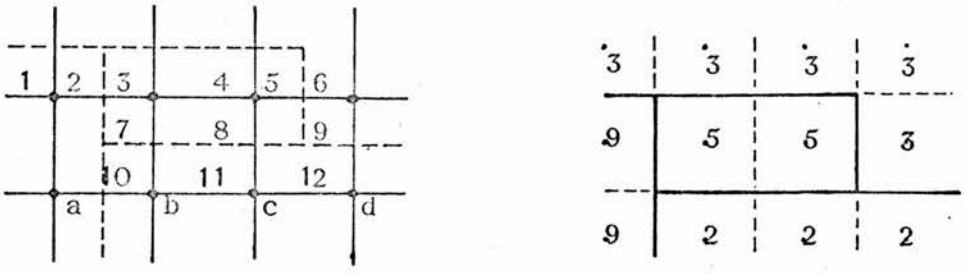
output files. It became clear at this stage that these lines, unlike the contour lines required two names, one for the value on each side of it. Though this was implemented, it was not fully exploited until a later stage.



Zone Boundary Pattern Generation within a Contour Cell

Figure 43.

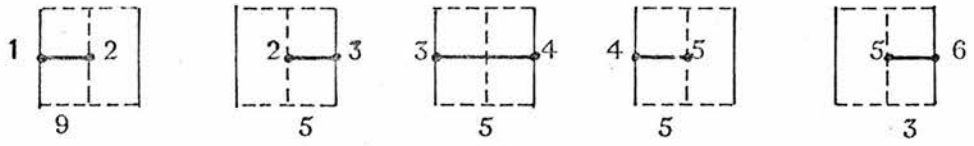
The boundary lines of the grid zones established, the next stage as with the contour lines which are also boundaries of zones, was to create zone shading. This shading since it had to work graphically on the block model as well as the plan view drawing, was created from lines co-linear with the profile lines. In the case of the surface zone shading where the zones are made up of rectangular grid cells, the line segment pattern is simpler to generate than was the case with the contour zones.



Shading Surface Zones

Figure 44.

In the diagram in Figure 44 , the shading lines fall on the boundary lines of the contour generation cells and run through the centre of the grid data cells. The value of the grid data for a cell in the dual grid is used to select the shading line used for that cell. For the cells shown above these lines become:

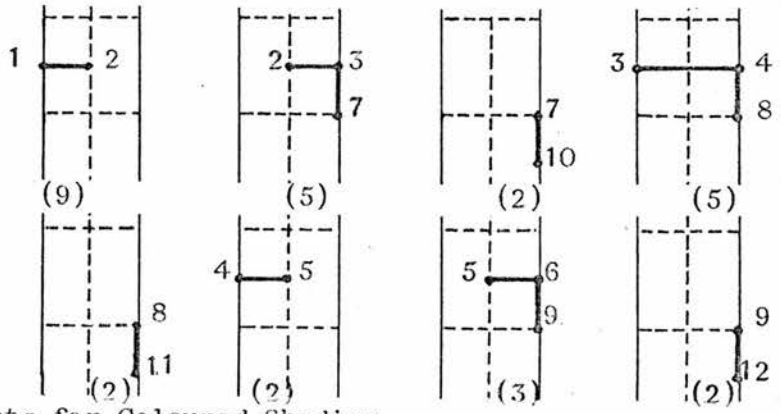


Line Segments for Different Colour Shading

Figure 45.

This classification of the shading lines can be carried out using the single row of zone shading values which corresponds to the zone grid cells centred on line AB.

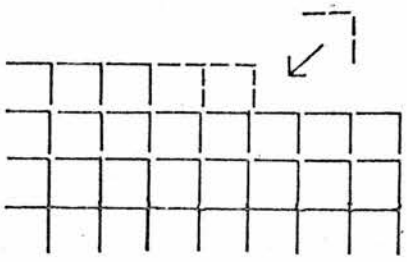
Though it is not shown in any of the examples this procedure was extended to give a cross-hatched form of shading. This was achieved by using two rows of the grid data; however, both these rows of data are already in core to establish the zone boundaries, so this does not create any new demands on data storage space within the routine. The shading lines which would be generated for the previous example are shown in Figure 46



Line Segments for Coloured Shading

Figure 46

These lines accumulate in the way discussed previously in the pattern shown in Figure 47.

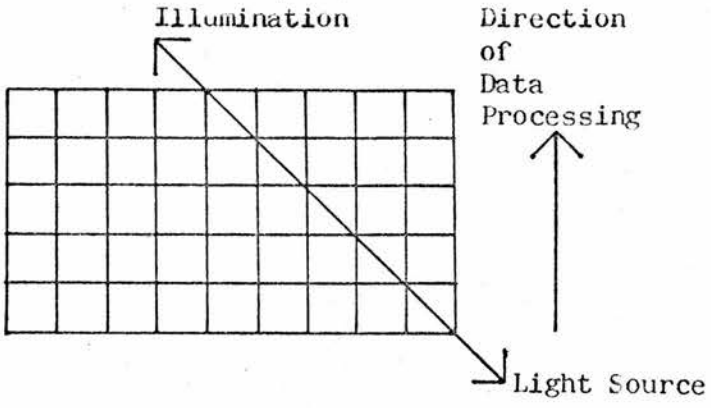


Accumulation of the Line Shading Pattern

Figure 47.

Another procedure which was developed in this early work, but which was only tested out at a later date, was a simplified form of hill shading. The creation of shadows is almost the identical process to the definition of hidden areas. The principal difference is the form the output from the procedure takes. In this case a simplified hill shading technique is discussed which is restricted by the orientation of the grid and the direction in which the grid is processed. The direction of illumination has to be accepted from the opposite direction to that used in processing data. For the original experimental routine this proved to be fairly restrictive; however in the system structure developed later it became possible to rearrange the data matrix and obtain a wider variety of results.

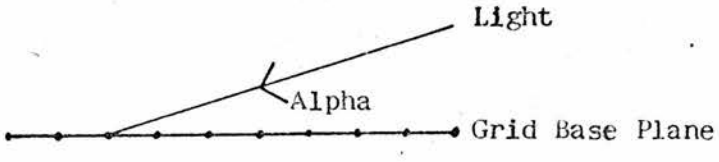
The intention was to improve the legibility of the plan view diagrams, but the technique also allowed shadowing to be applied to the perspective view of the block model. If a plan view of the data grid is considered as shown in Figure 48 , then it can be seen that the direction of illumination is the opposite to that normally used in



Grid Based Shadow Generation

Figure 48

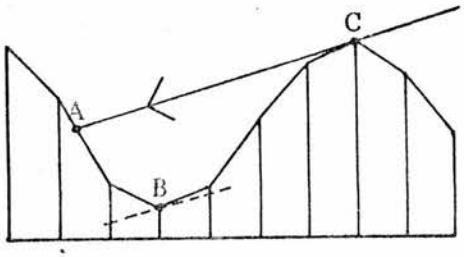
architectural sciagraphy. This was along the diagonal of the grid cells from the bottom right hand corner. Assuming parallel light rays with an angle of incidence to the plane of the grid of alpha, the simple section along one of these diagonals can be seen in Figure 49.



Section Through Grid Diagonal

Figure 49.

A modulated surface gives the relationships shown in Figure 50.

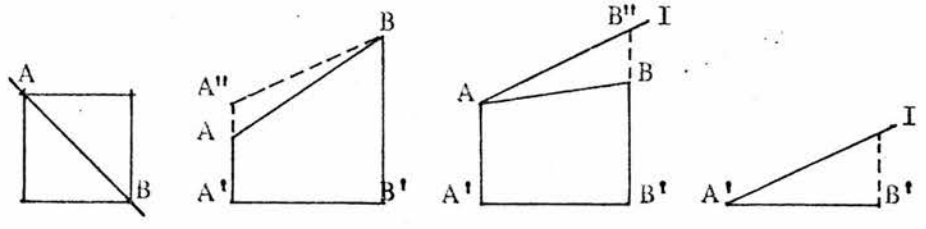


Section Through a Surface along a Grid Diagonal

Figure 50.

There are two sections to the shadow area AC. These are: the areas of the surface which are facing away from the source of illumination - BC, and the areas which, though facing towards the light source, have it obscured by some intermediated obstacle - AB.

The areas which are facing away from the source of light can be easily determined within the context of one grid cell as shown in Figure 51. If BB' is greater than AA'+X, then AB is facing away from the source of illumination. The areas which have shadow cast on them are a little more difficult to establish. The problem is simplified by having the direction of the illumination co-linear with one of the grid diagonals, since obscuring points for one grid point will lie along the line which passes through other grid points. A list of these shadow casting points

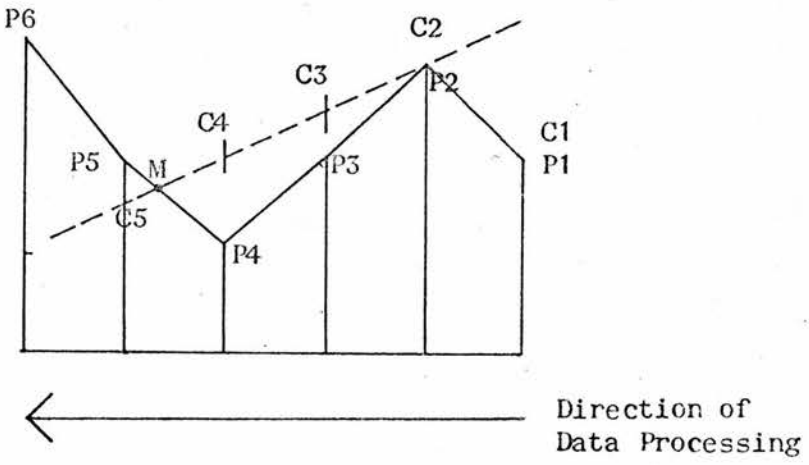


Shadows within a Single Data Cell

Figure 51.

such as point C in the previous diagram, has to be kept. However, once this has been done the procedure for evaluation whether a cell is in shadow is very similar and consists of the comparison of height values.

For one diagonal section line the process consists of:



Determining Extent of Cast Shadows

Figure 52.

Initialise the first shadow generating point: $C = P_1$

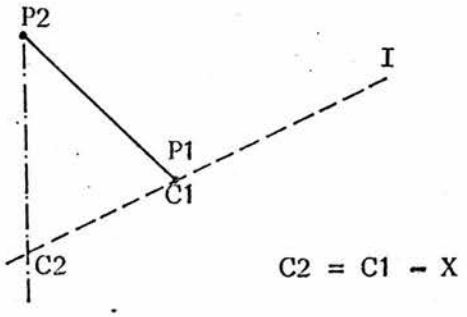
Take the first cell diagonal P_1P_2

If P_2 is greater than or equal to C-X, then the line is not in shadow.

There are three conditions which can occur for each diagonal line segment:

Case 1

Where the whole diagonal is illuminated as in the case P_1P_2, then both end points of the diagonal will be greater than or equal to the correspondong vertical heights of the shadow casting line, in this example C_1C_2

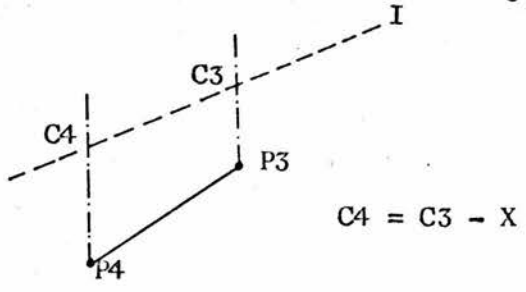


Diagonal Illumination Case 1

Figure 53.

Case 2

Where the whole diagonal is in shadow as in the case P_3P_4 in which case both end points of the diagonal will be less than the corresponding vertical heights of the shadow line C_3C_4.



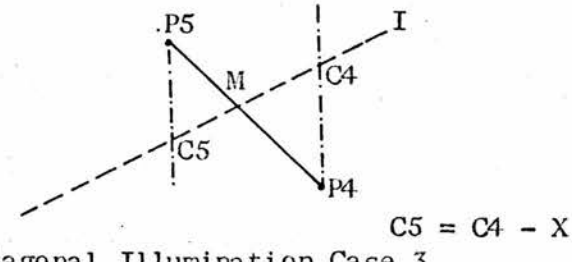
Diagonal Illumination Case 2

Figure 54.

Case 3

Where part of the diagonal is in shadow as in the case of line P_4P_5.

In this case $P_4 < C_4$ but $P_5 > C_5$ and the point M has to be evaluated.



Diagonal Illumination Case 3

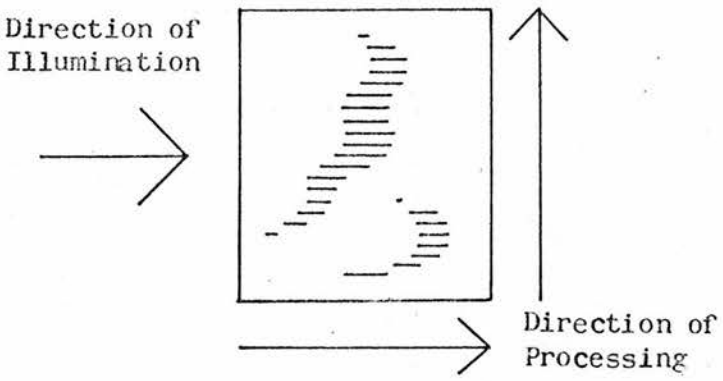
Figure 55.

This simple procedure is possible because of the form of symbolism used to represent the areas of shadow. Since the boundaries of the shadows are not being defined, and since the shading lines are co-linear with the cells diagonals, it is not possible to express fine detail. The level of detail is related to the grid cell size in relation to the total grid.

It is consequently possible to make simplifying assumptions when creating the shadow shading lines. In the case of this diagonal shading the main approximation is that the straight line segments along the diagonals of the cells will lie on the surface interpolated over the cell. Where the hyperbolic paraboloid patch is used this will not be strictly true and will depend on using a large number of cells within a drawing to make it work graphically. It must be noted, however, that if the triangulation described above using the diagonal along the direction of illumination is used then this procedure will not be an approximation. However, it will still be necessary to use sufficient grid cells for the diagonal lines to read as a reasonable form of shading.

It is necessary in the case of diagonal illumination to store an array of shadow generating points to be able to test the current row of height values against previously occurring high points. This can be compared with the use of the obscuring polygon boundary in the tests for hidden lines. However, the choice of illumination along the diagonal considerably simplifies the process. A simplified creation of shadows can also be achieved where the illumination is along the rows or the columns. The diagonal direction was originally chosen to satisfy

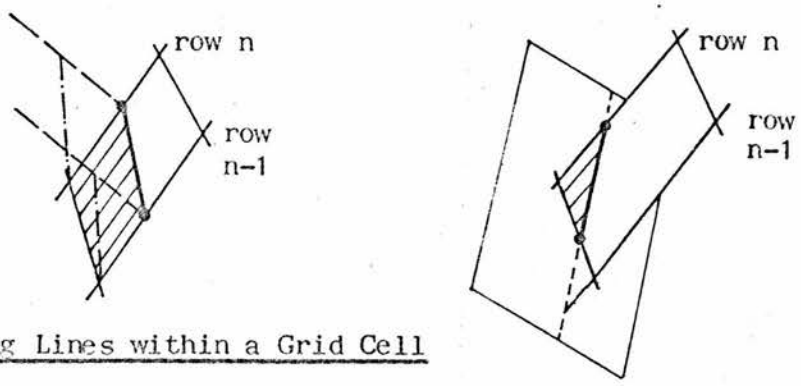
an existing convention. Illuminating the surface from the left along the rows results in the greatest simplification. It is no longer necessary to store a full array of shadow generating points. Only one point needs to be stored for the row which is currently being processed.



Hill Shading Parallel to the Grid Rows

Figure 56.

This direction of illumination has the advantage that the line segments generated along the row will lie in the surface of the hyperbolic paraboloid patches with no approximation being made. This means that more than one line for shading can be created per grid cell, and also should

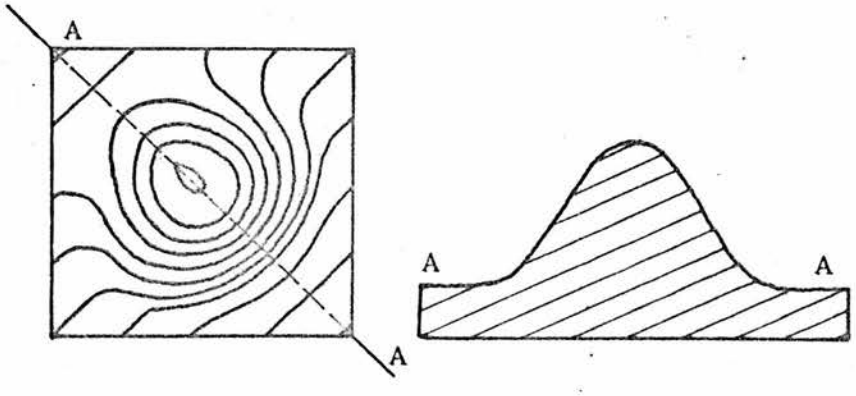


Multiple Shading Lines within a Grid Cell

Figure 57.

it be required it is possible to define line segment boundaries to the shadow areas along the row, by taking two rows of data as shown in Figure 57. This can be done using a simple testing procedure.

Both the diagonal form of shading and the horizontal form can be used on an oblique view of the block model. Though the original intention was merely to improve the legibility of the plan view drawing, there are cases where surface shadows on the block model may also be useful. The restriction on the direction of illumination means that though this shading is a useful graphic device it is limited in the ways it can be used to illustrate the effects of real shadows. Where correct forms of illumination are required, then the orientation of the grid has to be chosen to coincide with the required direction of illumination. Where grid data points are prepared by hand as in the case of the Honey Hill data set, this is impractical except in rare special cases where the importance of the result warrants the time spent in preparing the new data. Where the grid's value is defined by an interpolation procedure which fits a surface through arbitrary data points, then it is much easier to rotate the grid's orientation to suit any angle of illumination.



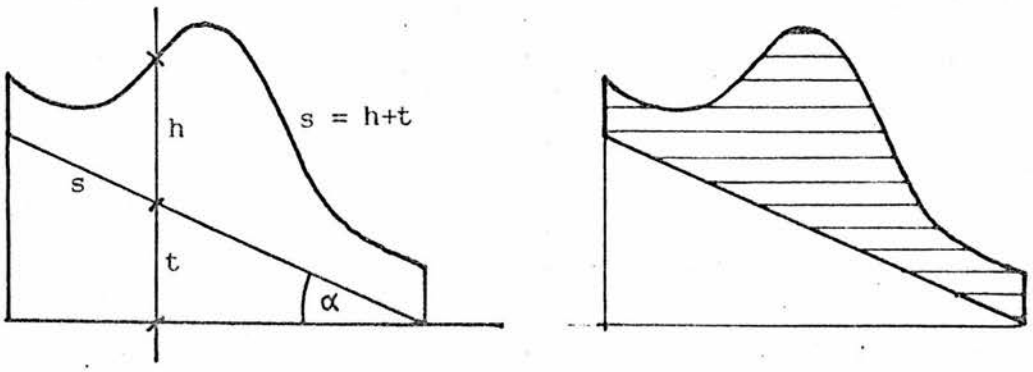
Oblique Contours

Figure 58.

GEOMETRICAL OPERATIONS: PRELIMINARY STUDIES: OBLIX

An alternative approach to making the plan view of surfaces more legible using line drawings was proposed by Tanaka (1932). In essence, this method consists of generating oblique contour lines, and when these are presented in plan view the graphic effect is very close to the use of hill shading. This method has been successfully implemented as a computer program by T. Peucker in the Department of Geography in Simon Fraser University.

Though the OBLIX contouring algorithm was originally designed to produce horizontal contour lines it is not necessary to modify it to create this effect, if certain restrictions are accepted. The contour line-segments in OBLIX are created from the height values at the corners of a grid cell. These height values are assumed to lie in an orthogenal direction to the base plane of the block model. If these height values are modified as shown in the diagram in Figure 59.

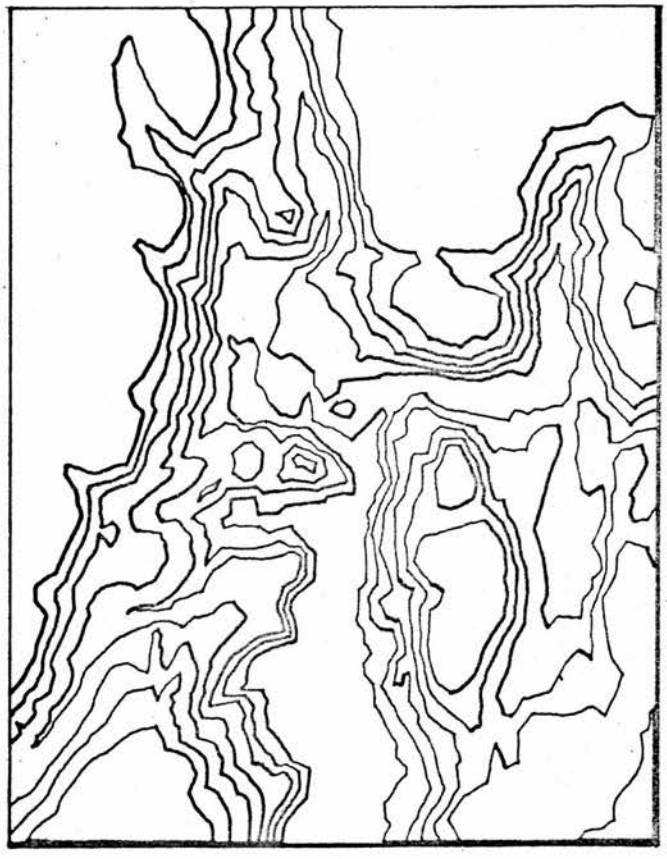


Creation of Oblique Contours

Figure 59.

and the horizontal contours created from these new height values, the result, when presented in plan view is the same as that obtained from oblique contour lines. The limitation of this approach results from the possible sizes of '$\underline{t}$', which depends on the angle taken for α. If

$\alpha = 90^{o}$, it is clear that t = 00 , which cannot be catered for. There will therefore be some angle between 45^{o} and 90^{o} which raises the value of 't' to an unmanageable size. However, it appears that the angles between 0^{o} and 45^{o} give the best graphic effect, so this turns out in practice to be a minor restriction.

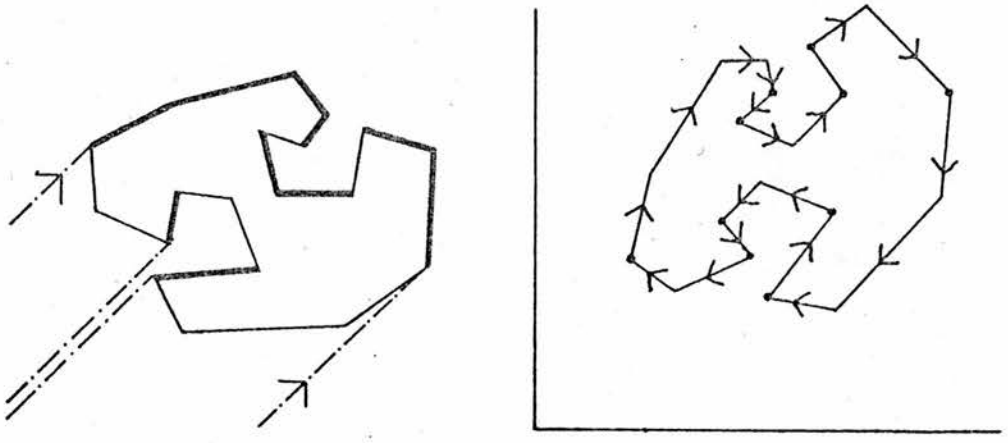


Shadowed Contour Lines

Figure 60

Using the technique shown in Figure 60 , it is not necessary to consider the three dimensional effect of the illumination to obtain a reasonable graphic effect. Ignoring cast shadows, and assuming light rays to be in planes parallel to the base plane, any contour line on a surface facing the light is rendered with a thin line and conversely any contour line which is on a surface facing away from the light is rendered with

a thick line. This effect can be obtained by operating on the co-ordinates of the contour line segments. The way in which this is done is illustrated in Figure 61 . If the contour line segments are rotated so that the direction of illumination is parallel to one of the axes, then the direction of the line segment can be used to classify it as a thick or a thin line.



Line segments going left are thin.
Line segments going right are thick.

Procedure for Generating Shadowed Contours

Figure 61

Another graphic effect can be obtained using the hachuring technique. This can also be created from cell by cell processing. Where triangular plane facets are used to interpolate values within the cell the technique is easier to implement than where the hyperbolic paraboloid patch is used. Given a cell shown in Figure 62 , it is possible to construct a line from the centre point of the cell, along the surface of each of the two plane facets, in the direction of maximum slope, so that the line's

Generating Hachure Lines for Each Grid Cell

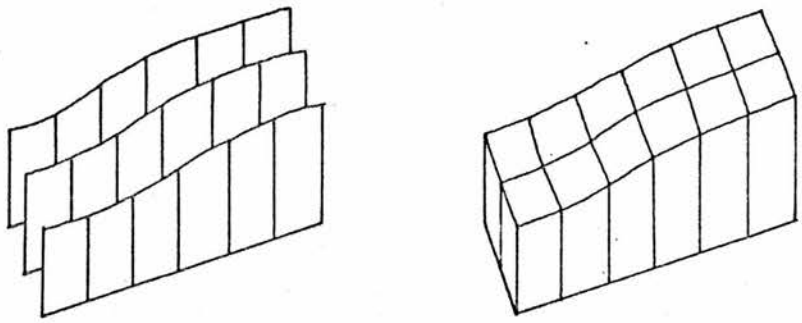
Figure 62

length is proportional to the steepness of the slope. These line segments will be spread over the map in an even distribution. A better effect could probably be obtained if some way of varying the density of these line segments could also be found which was also simple to implement. One possibility which cannot be achieved using a cell by cell procedure is to create the hachure lines at some regular spacing along a contour line.

Related to this technique is a method developed by D. Sheppard in the Laboratory for Computer Graphics for indicating the drainage basins formed by a particular terrain model. If a line is drawn in each cell indicating the direction of maximum slope then the cumulative effect will be a map which shows the direction of run-off. The boundaries of these catchment areas defined in this way cannot be determined by the cell by cell approach. It is only in the resulting drawing that the edge of each drainage area becomes apparent. A possible technique for automatically locating the boundaties of these zones was investigated, based on the relationship between peaks, pits and passes (Warntz, 1967). However, it appeared that several passes through the data would be necessary to implement such a procedure and consequently it was left for development later when better system facilities had been prepared.

GEOMETRICAL OPERATIONS: PRELIMINARY STUDIES: OBLIX

Another set of boundary lines which it was found possible to generate using simple cell by cell processing developed from the work done on the hill shading procedures. These boundary lines were the edges of the



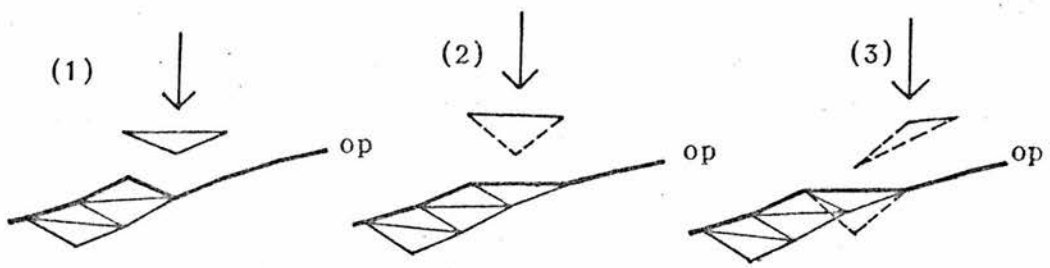
The Graphic Effect of Profiles in Contrast with Patches

Figure 63

visible sections of a surface when it was viewed from a particular point. To implement this idea it was found that the profile method of generating the block model drawing had to be abandoned in favour of a true surface representation. The difference between these two can be seen in the diagrams in Figure 63.

Though most of the surface lines have been generated using surface patches, the drawing has been created using profile sections. This was done to simplify hidden line elimination. The difference between these two ways of creating block models appears in the creation of the new obscuring polygon. For the second method it was necessary to update the obscuring polygon after each new surface patch has been added to the drawing. Figure 64 shows the procedure where triangular patches are employed.

The obscuring polygon merely defines the current silhouette of the



Updating the Obscuring Polygon Boundary for Patches

Figure 64

section of block model which has been processed. It is only when the next row of cells have been processed that any part of this polygon boundary which still remains in the new polygon which can be regarded as part of the boundary of an invisible area. This means that the obscuring polygon line segments must be tagged so that, on one hand they are not output twice as boundary lines and, on the other, output prematurely before they have been established as the required boundary segments. It is also necessary to retain enough information to be able to reconstruct the three dimensional coordinates of these boundary lines from the picture plane coordinates used in the hidden line elimination process. Storing the relevant sections of the obscuring polygon provides the front section of boundary line for the hidden area. The rear section of this line must be obtained by projecting the obscuring polygon back onto any forward facing facet cut by it on the picture plane. Again, the major problem is to recover the three dimensional coordinates from the picture plane coordinates. Though this process was briefly investigated, it was not implemented for the block model program. Its most

likely application appeared to be within two dimensional mapping programs. The areas visible from a road, for example, could be created by taking the union of all the areas visible from a series of positions along the road. A similar application had already been developed (W. Fetter, 1965) to find those sections of proposed aircraft flight-paths which lie outside the vision of existing radar stations. As the various options which have been discussed in this Chapter were added to the program originally designed to create drawings of block models, it became clear that a more versatile control structure was becoming necessary. This was not only to make the program simpler to use, but also to provide a framework in which some of the facilities not strictly related to block models, could be provided in their own right. This development towards a system is the subject of studies presented in the next Chapter.

CHAPTER 7

SYSTEM DEVELOPMENT

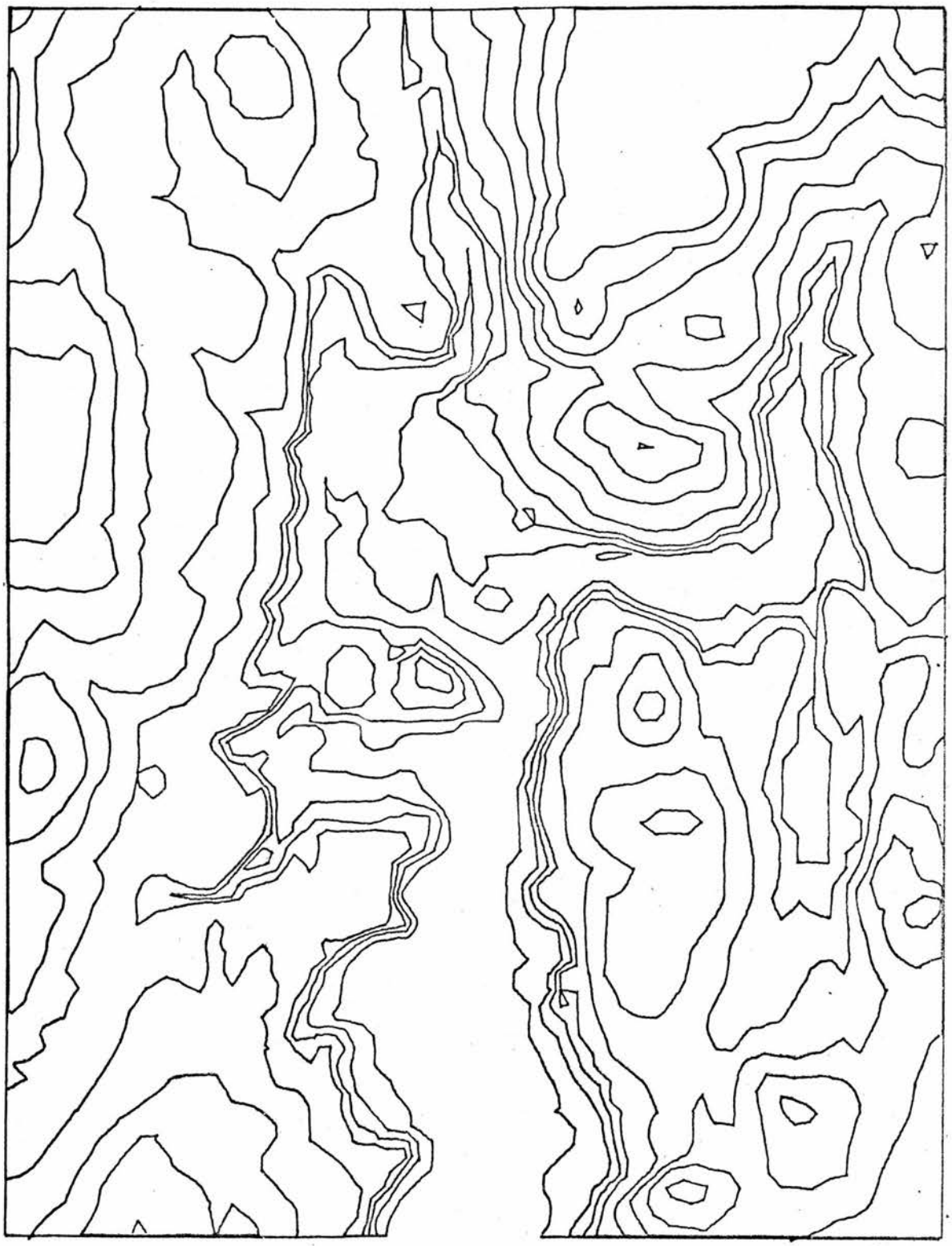
SYSTEM DEVELOPMENT

The original OBLIX program was written as a single routine. This was not because subroutines could not have been used, but because the program had to be made as fast as possible. As work progressed it became apparent that there were alternative ways in which the various processes could be carried out. Some of these alternatives, though not very efficient for producing block models, provided other useful facilities. Inevitably, this led to the adoption of a system approach; the aim being to provide a collection of basic processes which permitted a wider range of overall activities to be carried out.

The first stage in this development was an extension of the uses of the output files created by the OBLIX routine. These output files contained line segments from boundary lines and from surface symbolism, such as line shading. It was the boundary line segments which initially provided the data for alternative procedures. The line segments for zone boundaries and for contour lines were generated a row at a time and a column at a time. When these lines were drawn out they accumulated to give the continuous boundaries shown in the Figures 1 and 2. created from the Honey Hill data. By sorting these line segments into order so that they could be concatenated into polygon vertex lists, a series of further applications became possible for these data.

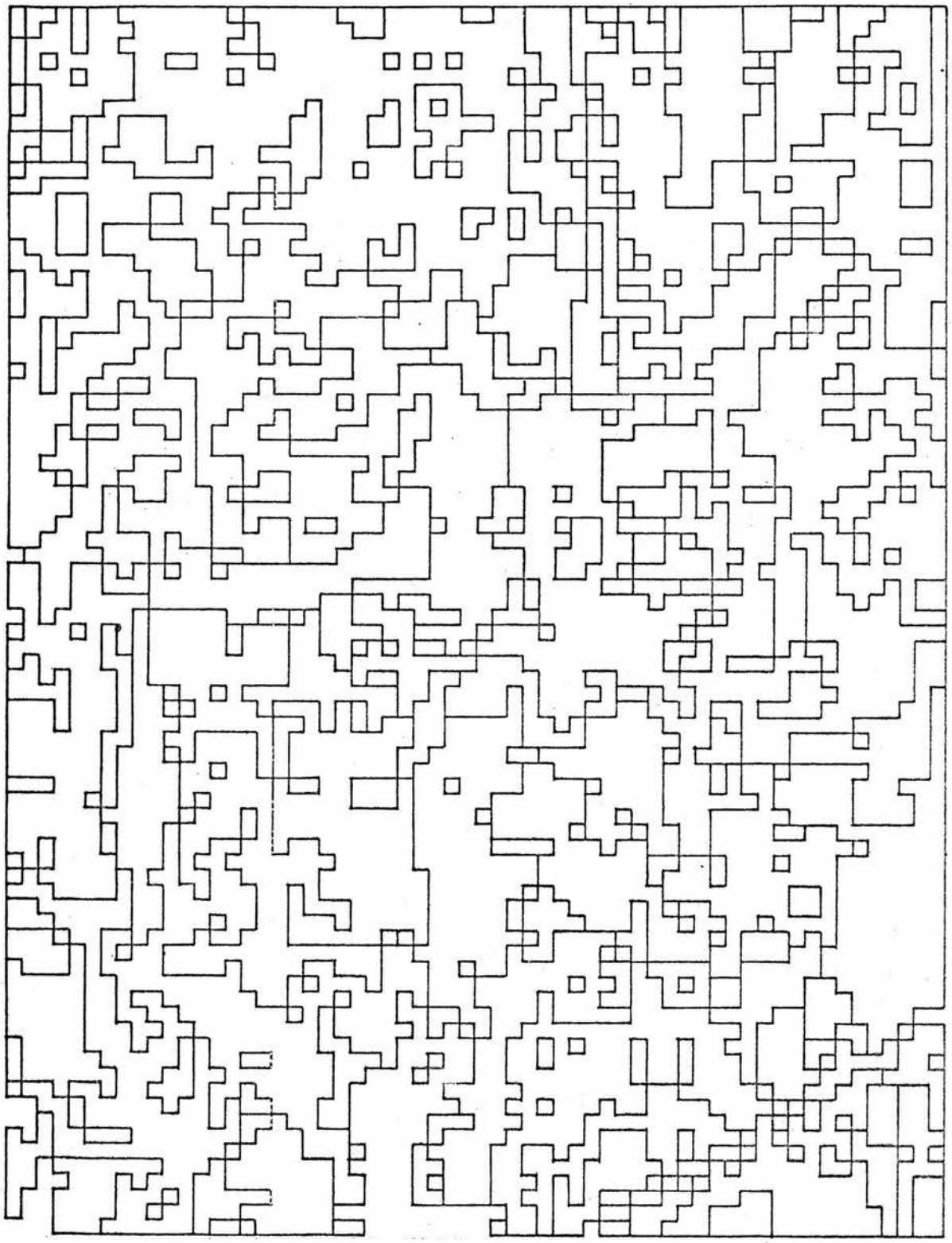
The primitive sorting operation to create this new data structure is given in Figure 3. The line segments are given as coordinate pairs. If the first coordinate of one pair is matched with a second coordinate of another pair, it is possible to concatenate the first list to give the second list where each coordinate only occurs once and the coordinates appear in the order in which they would be found following the boundary line.

SYSTEM DEVELOPMENT



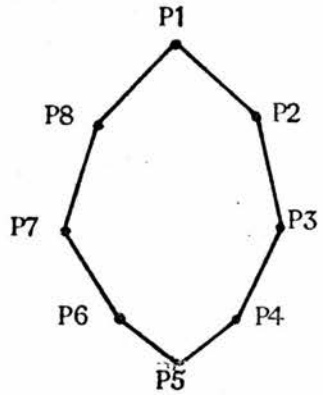
OBLIX Contour Line File: Honey Hill

Figure 1



OBLIX Grid Zone Boundary File: Honey Hill: Tree Distribution

Figure 2



Line Segment List		Polygon List
x6,y6	x5,y5	x6,y6
x5,y5	x4,y4	x5,y5
x7,y7	x6,y6	x4,y4
x4,y4	x3,y3	x3,y3
x8,y8	x7,y7	x2,y2
x3,y3	x2,y2	x1,y1
x1,y1	x8,y8	x8,y8
x2,y2	x1,y1	x6,y6

Concatenating Line Segments to Give Polygon Boundary Lines[1]

Figure 3

Once the data were in a polygon boundary form, it was possible to make use of a variety of procedures which had already been developed, such as point in polygon and line shading routines. In the case of these two operations, programs were written from first principles, again to provide a working knowledge of the alternatives available.

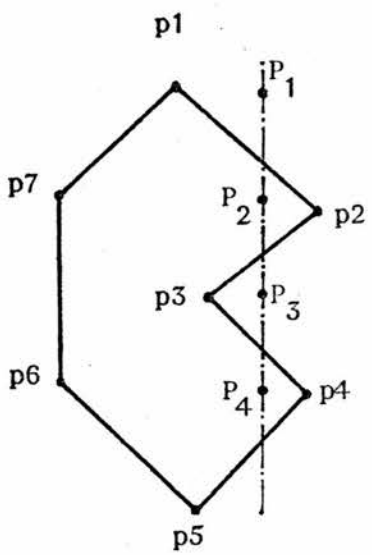
A point-in-polygon routine was written originally for a different purpose, as the principal routine in a preprocessing program to prepare data for redistricting. The basic data for this program consisted of a file of voters where each voter was associated with the geographical coordinate of his place of residence. The overall aim was to define voting areas which contained equal numbers of voters, taking into account the existing distribution of voting preferences. Though this task was carried out by the main programs starting from any arbitrary voting areas, it was hoped that by starting with reasonably compact voting areas, containing as far as possible the correct number of voters in each, expensive running time for the main programs could be saved.

1. See Appendix B, Subroutine Chain.

SYSTEM DEVELOPMENT

The technical problem was to provide an interactive system which allowed approximate solutions to be prepared quickly. This reduced to the problem of counting voters within externally defined zones. It was found that the system operator could, by incrementally adjusting a starting boundary network, create relatively inexpensively a range of starting points for the more sophisticated redistricting programs so saving the main programs from many preliminary cycles which would otherwise be necessary before arriving at a satisfactory solution.

The point in polygon procedure which was implemented is summarised in Figure 4 . The problem was to classify the point as either inside or outside the polygon, given a string of coordinates defining a polygon boundary and the coordinate defining a point.



Boundary Intersections above the Point		
P_1	0	Outside
P_2	1	Inside
P_3	2	Outside
P_4	3	Inside

Point-In-Polygon Test

Figure 4 .

If a vertical line through the test point is intersected with the polygon boundary, then it can be seen from Figure 4 that the number of intersections above the point can indicate whether the point lies inside

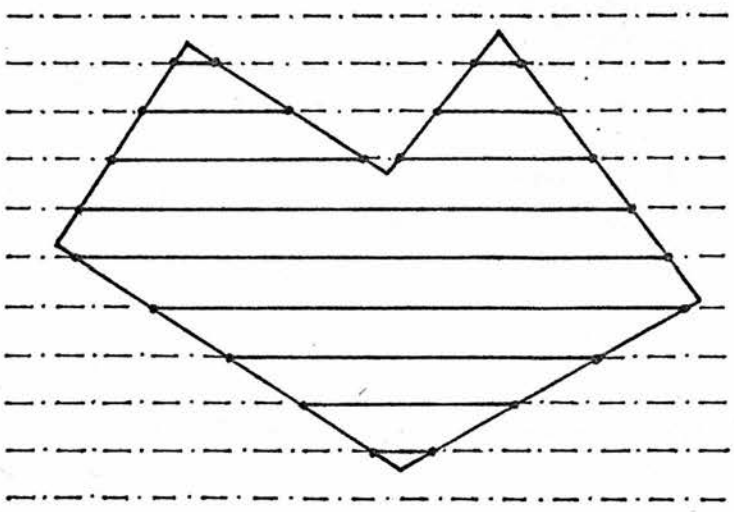
the polygon or outside it. In fact any line through the point can be taken but the raising of a vertical or horizontal line simplifies the testing procedures used. The same line intersection test written for the OBLIX hidden line algorithm was originally used, but it was found that a variety of simplifications could be made which considerably reduced the total routine.

Using a vertical line makes it possible to immediately discard all line segments where both the 'y' coordinates of their end points lie to the left or the right of the 'y' coordinate of the test point. This automatically leaves only those line segments which cross the vertical line through the point. If the point is then tested against these line segments in a similar way to that used in the OBLIX line crossing routine, it is possible to classify the intersections as lying above or below the test point. When counting the number of intersections above the point, an even number indicates that it is outside the polygon while an odd result indicates that it is inside the polygon. There are complications: a polygon vertex lying vertically above the test point creates counting problems and, when the point actually falls on the boundary, the result is indeterminate, but in principle this gives the basic idea.

The application of the point in polygon routine and a set of points to zone boundary file, or a contour file makes it possible to associate a point with other properties. For example, if a map is displayed on an interactive display and a point is indicated on the surface of the map, then it becomes possible for the system to print out the name of the zone in which it lies and extra data which could not be fitted onto the display. Similarly, in carrying out internal procedures the same kind

of facility is often required, as shown by the previous example where the number of people in each new zone had to be calculated. In any such system the point in polygon routine is a fundamental requirement.

Following the point-in-polygon routine a polygon shading routine was developed. The starting point was a line intersection routine, but again simplifications were found to be possible. The operation was akin to the hidden-line removal algorithm in OBLIX except that the line segments of a shading line which lay within a polygon were required as output.



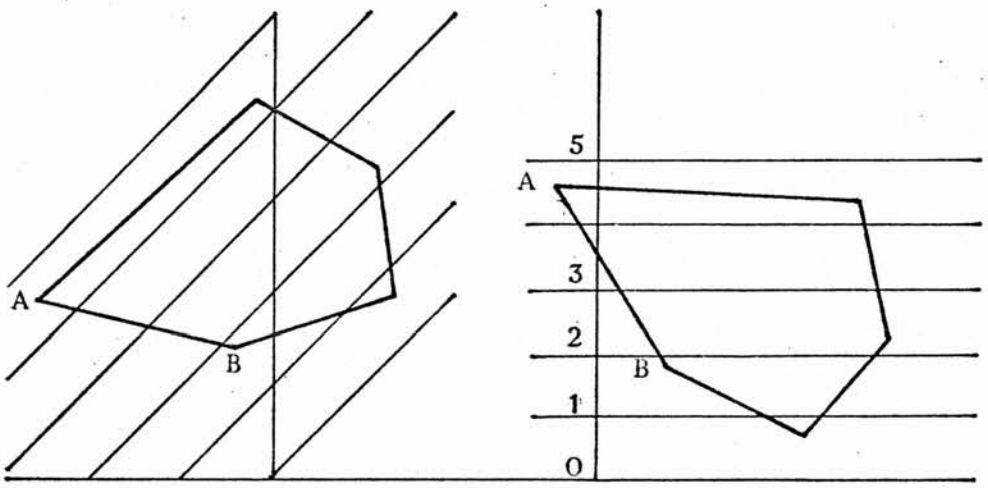
Polygon Shading

Figure 5

A variety of different forms of shading symbolism could be built up by overlaying grids of parallel lines with either different spacing or different orientations.

In much the same way that the use of the vertical cutting line created simplifications in the point-in-polygon routine, the rotation of the polygon boundary coordinates so that each set of shading lines could

be treated as though they were parallel to the X axis, made this routine simpler to construct. By converting the coordinates of the polygon boundary into units of the grid spacing it became a simple procedure to define the end points for the shading line segments. Each shading line coordinate would have an integral value for its 'y' coordinate, and the corresponding 'x' value could be obtained by linear interpolation between a boundary line segments end points. Diagrammatically, this process is shown in Figure 6 .



Line Shading Procedure

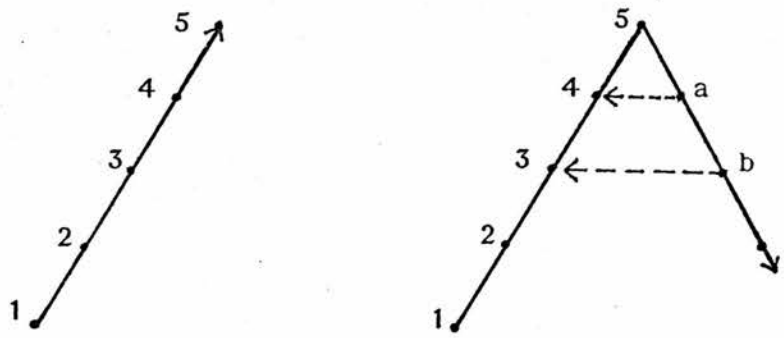
Figure 6

For the boundary line segment AB in Figure 6 it can be seen that after transformation the 'y' values for its two end points will be 4.6 and 1.8 respectively. The end points of shading lines will have the integer 'y' values between these two numbers: 2, 3, and 4, from which the corresponding 'x' values can be derived.

Once the end points of the shading lines had been established, the problem was to link up the correct end points. This was achieved by a coordinate sorting procedure. All the end points with 'y' values of 1

if ordered on their 'x' values, defined a single shading line. The first point was the beginning of a line segment, the second its end, the third the beginning of the next, and so on. Again, there were complications encountered in implementing this simple concept. Shading lines passing through vertex points as tangent lines to the boundary seemed to require special treatment. However, it was found that if a consistent sense of rotation was adopted in the order in which polygon boundary vertices were given round the polygon, then these problems could be resolved by using this rotational information.

The division of the shading procedure into two parts is an example of a system decision defining two procedures which could have other uses rather than one special purpose procedure. It was possible to construct a built-in sorting mechanism which linked the end points of shading line segments as they were created. If the direction of processing is giving increasing values of 'y', as soon as this is reversed new shading line points can be linked back to previous points in the reverse order to that in which they were created, as shown in Figure 7 .

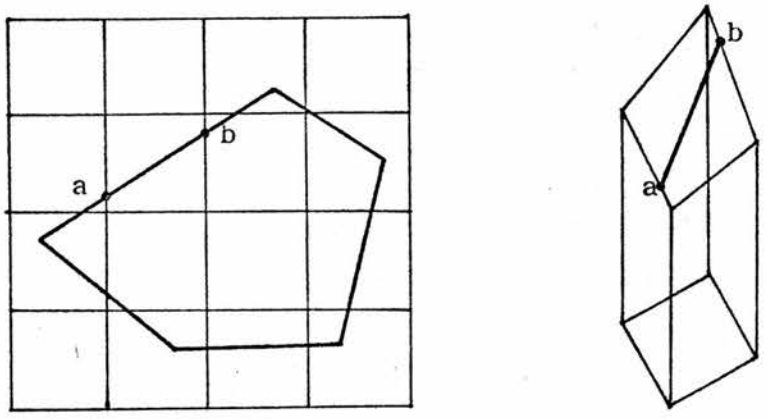


Built-in Linking for Shading Line Segments

Figure 7

It was not clear at the time when this decision was made that this division could be maintained. It seemed possible that, if shading became a critical factor in preparing graphic output, that it might become necessary to implement the shading algorithm in the second more direct form. However, the definition of the intersection points of a grid with a polygon boundary was applicable to another problem and the coordinate sorting was a basic procedure which could be used for many other processes.

In Chapter 6, surface lines on the block models were all generated from grid data, but it was stated that the method used for hidden line removal could handle other forms of surface line data, such as roads, railway lines and irregular field boundary networks: the kind of information typically found in maps. The problem was to project these lines up from the base plane onto the surface of the block model. The grid intersection procedure provided a quick way of doing this.



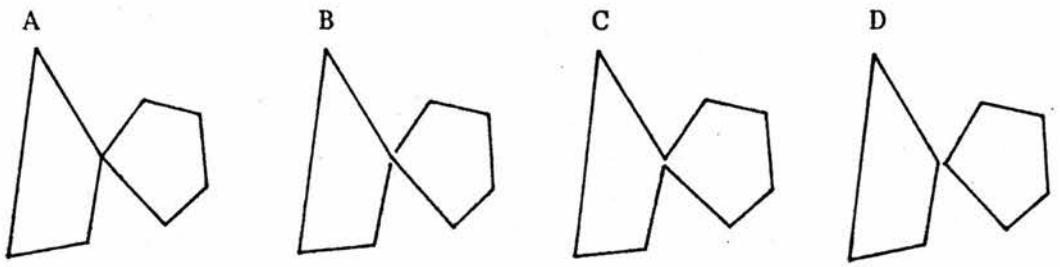
Projecting Lines onto the Surface of a Block Model

Figure 8

SYSTEM DEVELOPMENT

The polygon boundaries are processed in a similar way to that described in the case of the shading algorithm, except that two orthogonal grids are employed which match the data grid used for the surface height values. This creates a series of line segments with end points expressed in grid coordinates. If these line segments are then ordered to match the order of grid-cell processing employed to create the block model - using the coordinate sorting routine, then, while a cell is being processed, the same procedure used for calculating the three dimensional coordinates of the contour line segments can be used to provide the three dimensional coordinates for the polygon boundary line segment. Not only can this procedure be used for boundary lines; it can also be used for shading lines created on the horizontal base plane. This appeared to give the best strategy for creating surface shading which did not naturally fall in line with the data grid.

A further development which resulted from the work done on the line shading algorithm was a modification to the contour generating routine. The first version of this routine created line segments defined by the cell patterns given in Chapter 6. From these lines segments it was impossible to tell which side of the line was the higher section of the surface. Originally, this was not considered to be a problem since it was thought that it would be just as simple to link P1,P2 with P2,P3 as to link P2,P1 with P2,P3. However, it was when the so called self-crossing contour line was created that the difficulties became apparent.

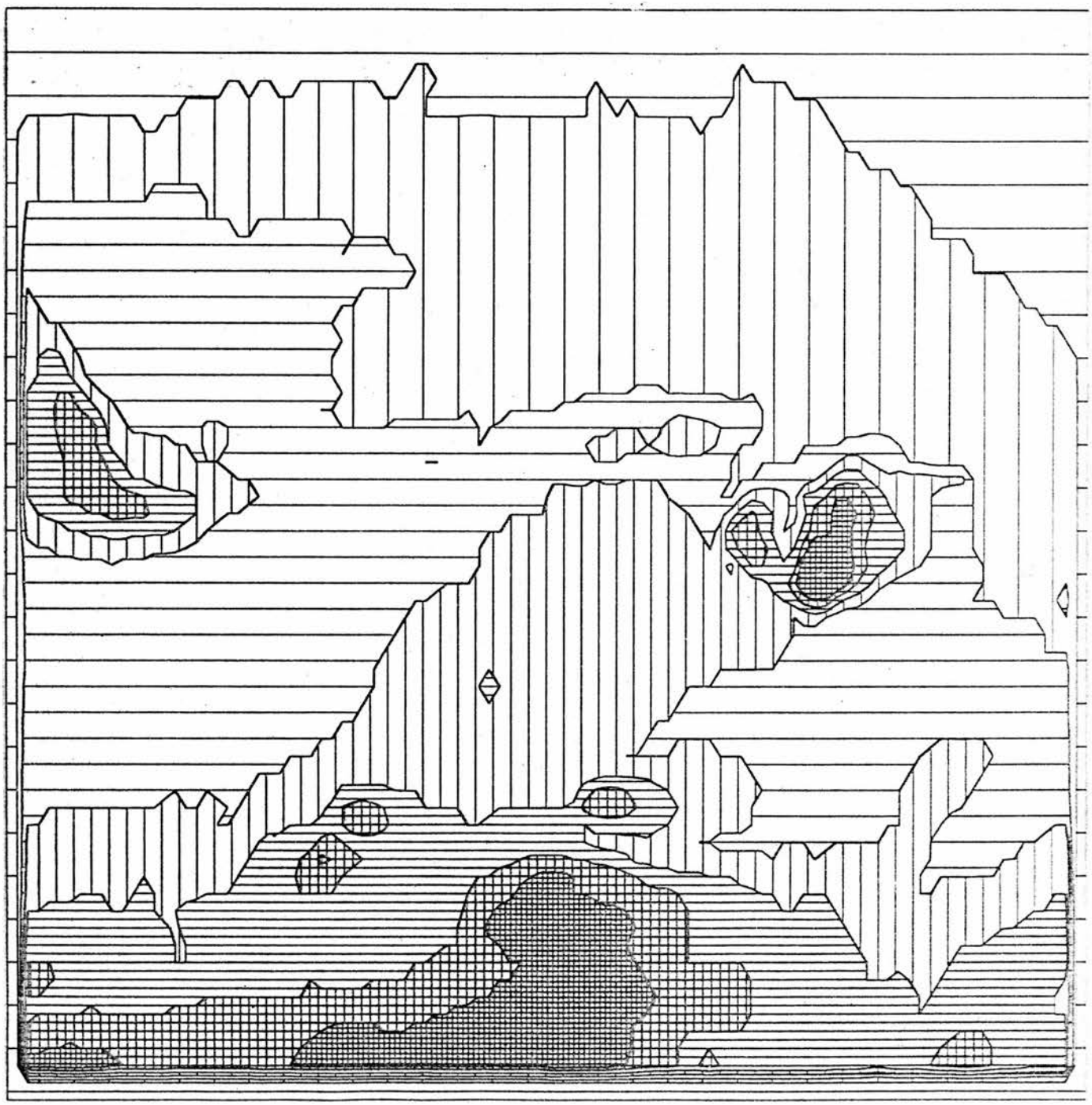


Self-crossing Contours

Figure 9

Without treating the direction of the contour line segments in a consistent way in relation to the zones on either side of them, it was not possible to prevent the polygon shown in Figure 9 A from being formed as the sequence of points indicated in diagram B. This crossing line created difficulties with the shading algorithm. Either of the two forms C or D had to be ensured if the contour files were to be used for creating shaded drawings. This was made possible by defining the pattern generation process so that the line segments would always be moving in a consistent clockwise or anti-clockwise direction about the higher zone. The shaded contour map of Edinburgh created by this modified process is shown in Figure 10. This diagram also shows a useful device which had to be employed to close all the contour loops so that the shading algorithm could be used. A ring of data values were placed round the original grid which were less than the minimum value in the grid itself. The effect of these extra two columns and rows can be seen round the edge of this diagram.

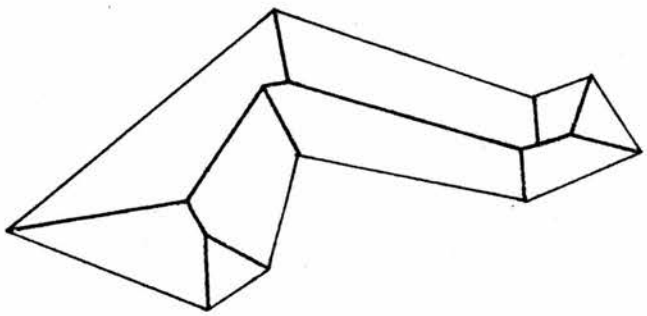
Other routines investigated which could be used with the polygon boundary data structure, included algorithms to calculate polygon areas, to determine polygon centroids and associated with this the centre of



Polygon Shading Used by OBLIX to Generate a Map of Edinburgh

Figure 10

shape which could be found by either shrinking the polygon, or by constructing the polygon skeleton shown in Figure 11



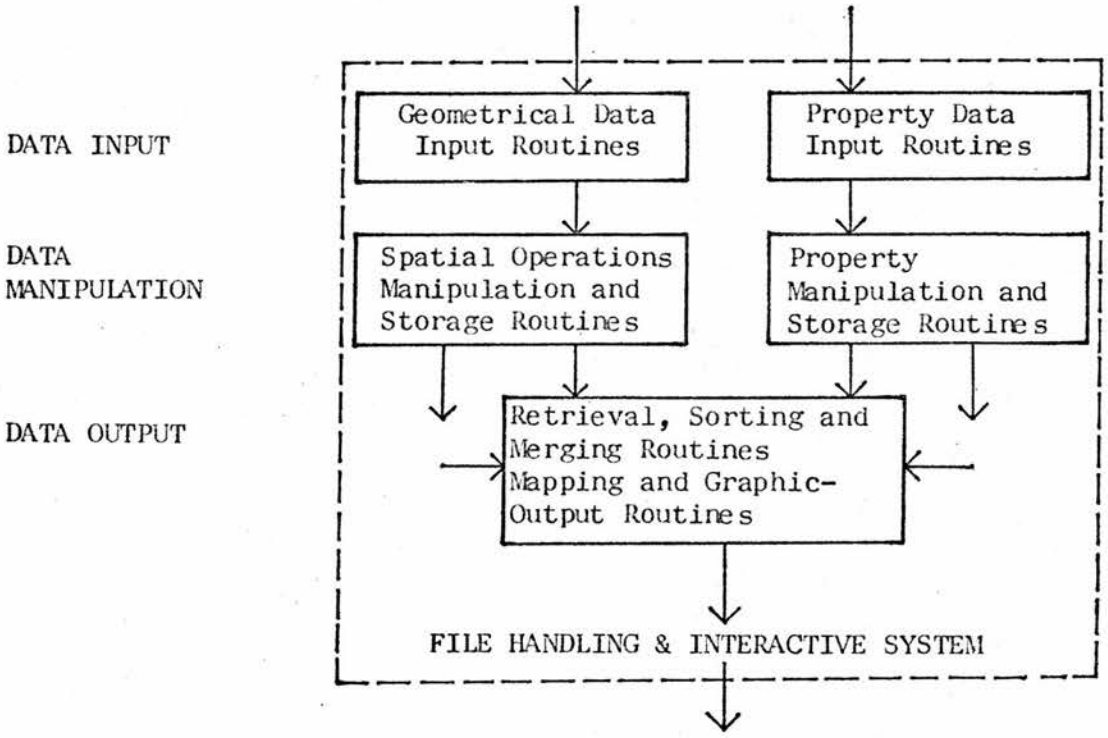
Polygon Skeleton

Figure 11

The first attempt to incorporate these procedures into a working system was carried out in collaboration with T. C. Waugh in GIMMS (Geographic Information Mapping and Manipulation System). The first version of this program was created for Leyland Systems Inc., Boston, in 1970, and its principal function was the creation of polygon maps.

The system structure used for GIMMS was based on a broad distinction between data which was classified as spatial data - in other words, the descriptions of zones as polygon boundaries, and data which was classified as 'non spatial' or 'content' data, for example the number of people who lived in a particular zone. This division could clearly only apply to the upper levels of the system; both forms of data would be processed by the same file handling routines. However, at the upper level the division reflects the fact that the operations which are performed on these two forms of data are generally speaking different and will require different routines to carry them out. The overall

structure shown in Figure 12 provided a scheme which was broad enough to cover most of the computer mapping facilities envisaged in the initial stages of this work.



Initial System Definition

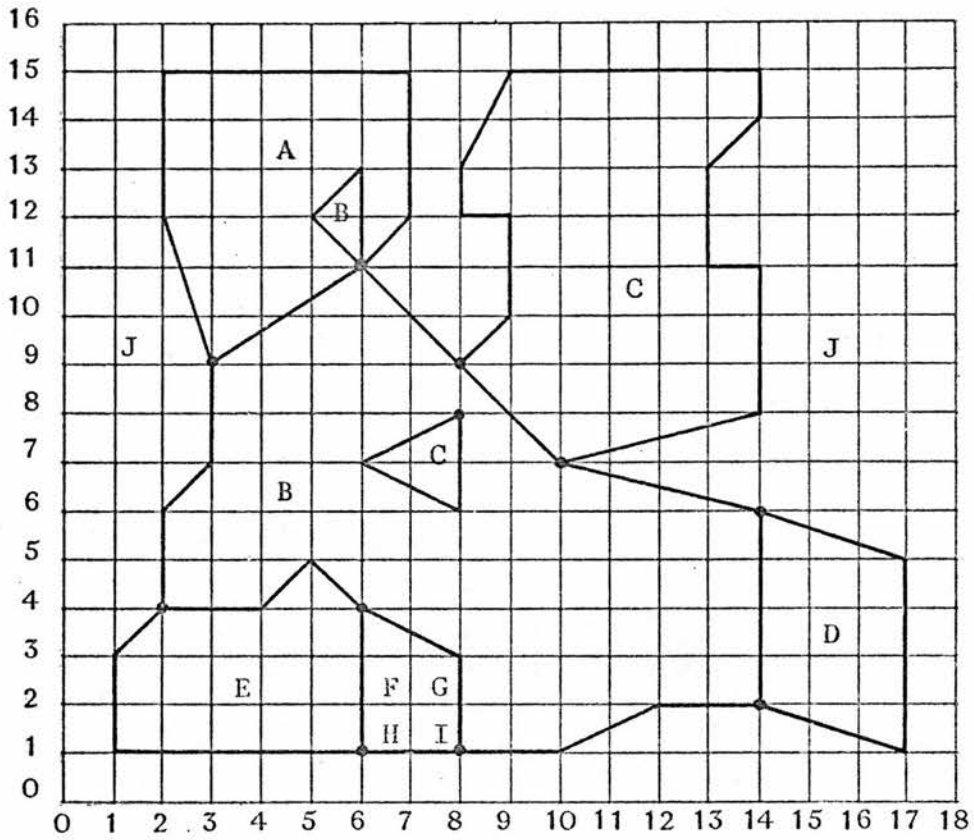
Figure 12

The data input routines were designed to convert the primitive input from digitisers or manual encoding techniques into an internal data structure which could act as a common base for a variety of manipulation processes. This provided the first stage in establishing a spatial description data-base. Test data used in the early experimental development of the system is shown in Figures 13 and 14.

These two sets of data were constructed to test the various system facilities which were being developed. The first problem which had to be overcome was entering the descriptions of the polygon boundaries. At

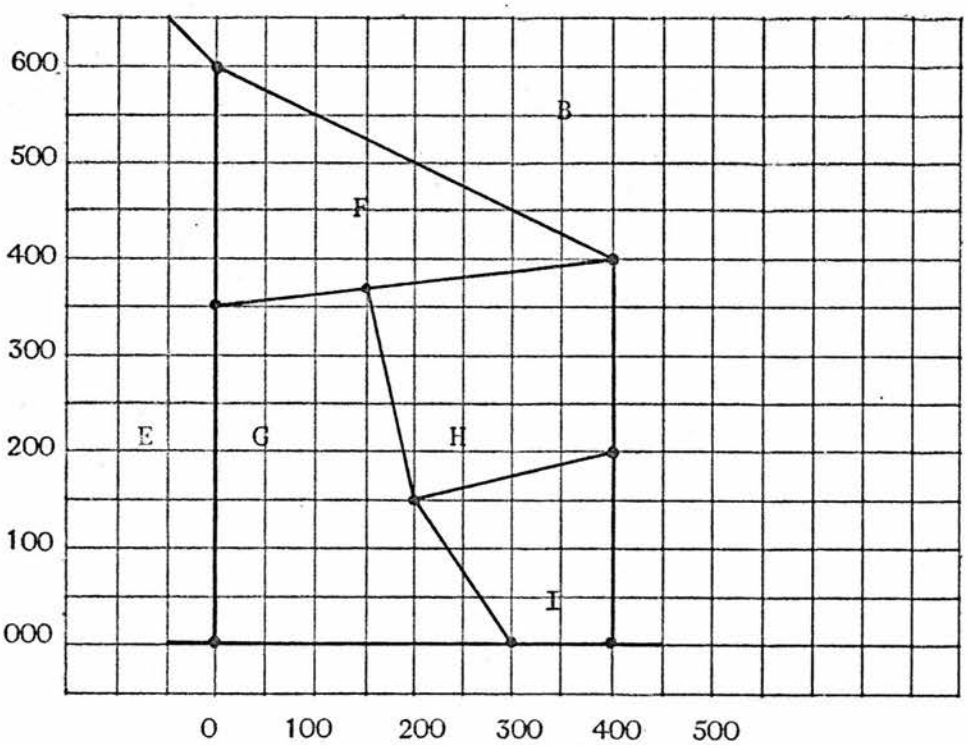
first sight it would seem to be simply a matter of digitising the boundary line of each zone as a string of coordinates. This had been the approach adopted by early workers in this area such as Warren Terryberry and Samuel Arms. However, the problem was that even for relatively simple boundary lines it was impossible to match the two occurrences of the same section of boundary between adjacent zones. Consequently, the approach adopted was to only digitise boundary line segments once and to indicate the two zones which they separated. This approach was proposed by B. G. Cook in 1967 in a paper "A Computer Representation of Plane Region Boundaries", and the dual naming of line segments which it proposed formed the basis for parallel work carried out on the DIME system.

The input data consequently was created as a collection of line segments. These were subsequently sorted into polygon boundaries as described previously for the contour line segments. The directional convention for digitising and naming these line segments had to be consistent. Since errors were easy to make, editing facilities were necessary in order to correct any incorrectly defined boundaries. Error messages could be provided, for example, where boundary line segments were missing. One development of the editing facilities was the capability to add more detail to an existing spatial description. The two data sets shown previously were designed to test this facility. Data-set II represents a later addition to be made to Data-set I. Naturally enough, the later data were digitised at a different scale and procedures had to be developed to handle the subsequent book-keeping operation of merging the two data sets.



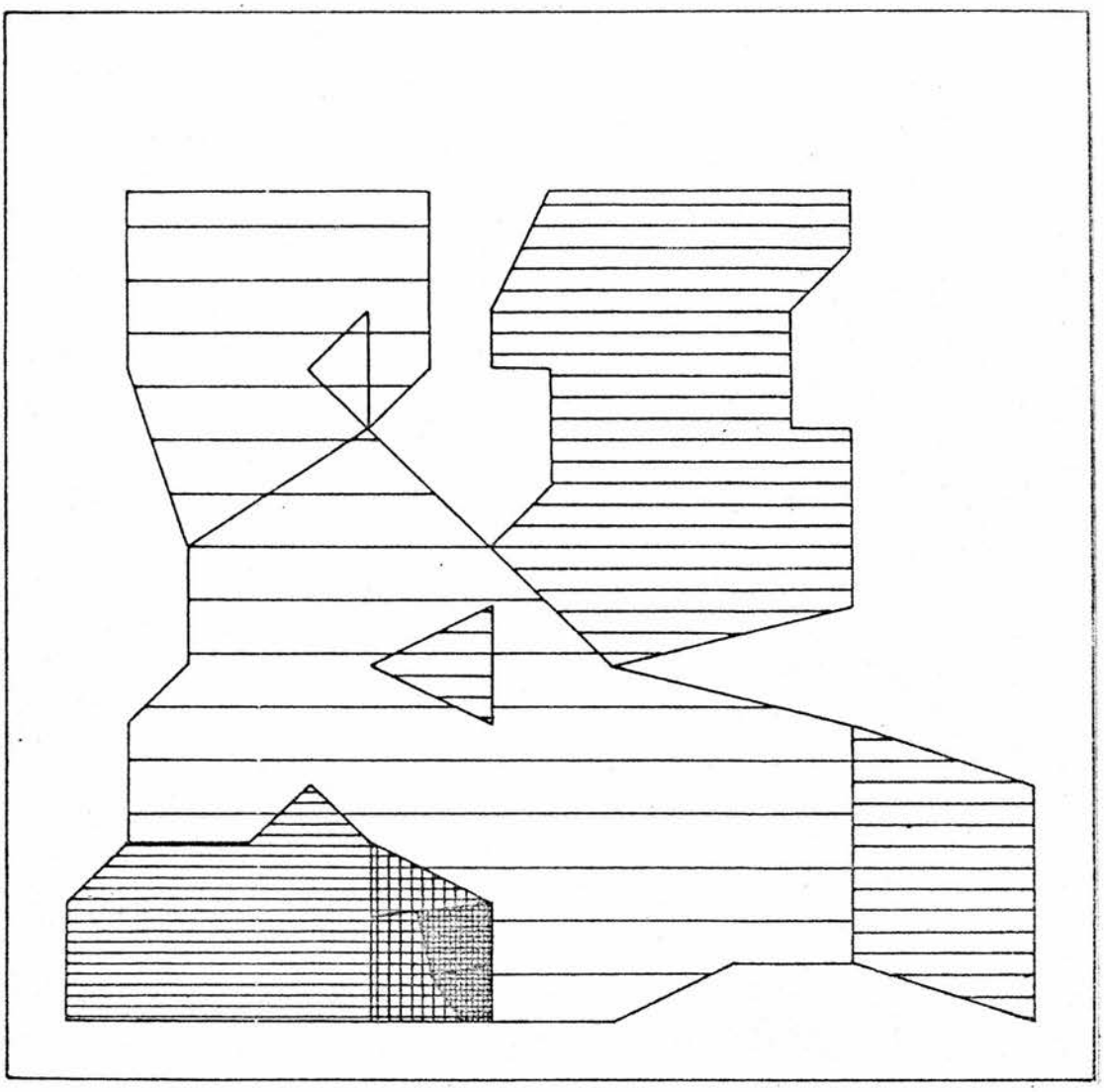
Test Data Set I

Figure 13



Test Data Set II

Figure 14



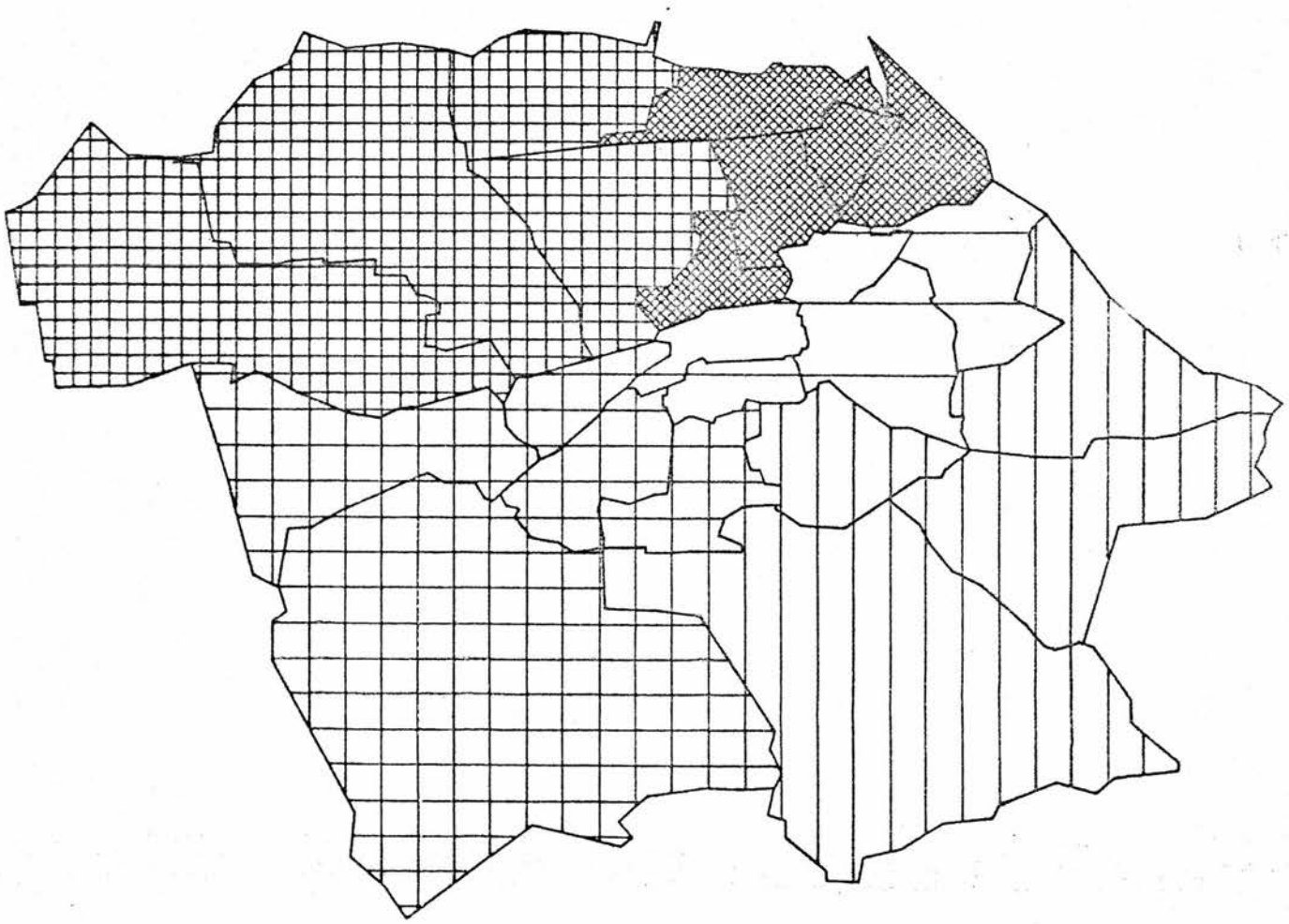
Test Output from the GIMMS System

Figure 15

The form of the polygons was chosen to test out the various geometrical operations described previously, such as calculating the area of the zones, determining their centroids, and experimenting with the form of the shading algorithm. A more complex shading routine than that described above was implemented. This was partly because it included a section for drawing boundary lines, and partly because an attempt was made to ensure that shading lines which were tangent to a boundary line segment were created in the simplest possible form. This proved to be

a valuable exercise in the use of the directional properties of the polygon boundary lines, but it was realised later that graphically it made no difference and the simpler form would have given the same results.

Graphic output showing the successful merging of the two data sets and the application of shading to each of the polygon zones is given in Figure 15. The use of the system to create a map of the City of Edinburgh is given in Figure 16.



Polygon Shading Used in GIMMS to Generate a Map of Edinburgh

Figure 16.

SYSTEM DEVELOPMENT

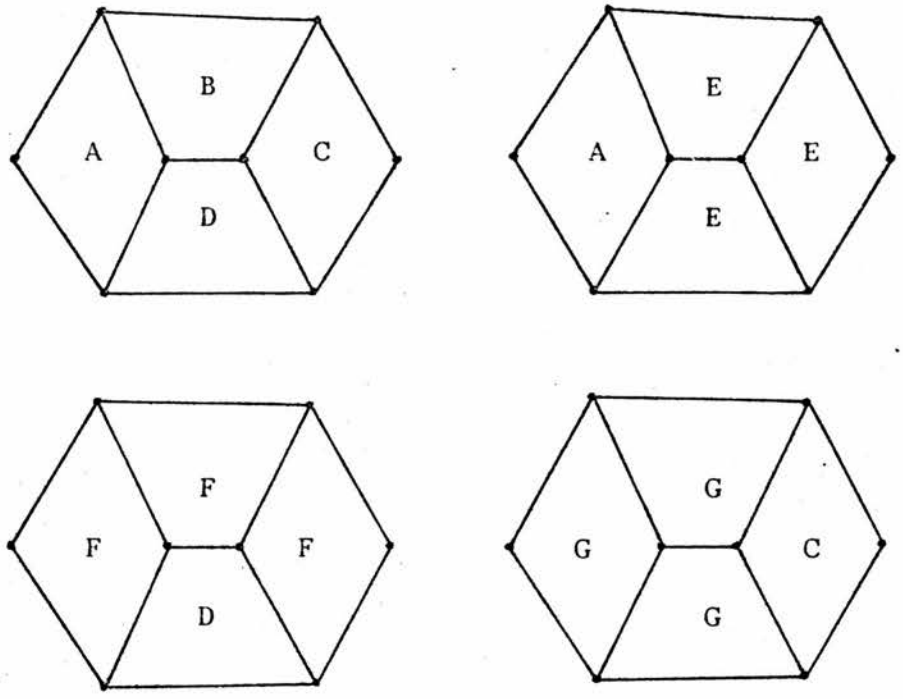
The original work on GIMMS was aimed at the production of polygon maps. Thus the spatial data consisted of descriptions of areas or zones. It was clear from existing practice in cartography that it should also be made possible to associate properties or values with line and point locations. This meant that the next natural step in developing the system was to extend the classification of spatial data to give the three types: areal data, linear data and point data. It was convenient to start with this division for the same reason that spatial data had been distinguished from content data: many of the existing processes for manipulating point data were different from those required to handle areal data.

The property or value data associated with a geometrical description also had to be examined further. The name of a zone was as much a property of the zone as for example the population density of the zone. However, the name of a zone is a different type of data to the population density. Consequently, it was necessary to subdivide 'content data' into the data classes which could be manipulated by similar processes. The starting point for this subdivision was taken as the three levels given by the nominal, ordinal and cardinal scales of measurement.

The way that these ideas were implemented in the GIMMS system can be found described in greater detail in the system manual prepared by T. C. Waugh, who continued with the design and development of the original experimental system, linking into the original structure many further facilities required for the automatic preparation and compilation of maps. The work in this Thesis on spatial models took a slightly different course because interest was directed more towards finding alter-

native ways of carrying out these basic operations, than the provision of a finished product. From the discussion presented in Section I, it is clear that both these approaches are essential, since the use of the system will provide information about what is needed, while alternative choices will eventually provide the best form for them to take.

The starting point for the next study on the geometrical descriptions of zones arose from considering the relationship between associated or property data and the geometrical data. In Figure 17 four mutually exclusive zones A, B, C, and D are defined by a network of boundary lines. If other properties are associated with these zones, so that several neighbouring zones have the same property, it is possible to set up an hierarchical arrangement of names related to the original network.



Aggregation of a Set of Basic Zones

Figure 17

Assuming that each of the zones A, B, C, and D are described by the polygon boundary description used in GIMMS, there are several ways in which the definition of a zone F can be given. The simplest description of F is probably as the collection of sub-zones A, B, C so that the relationship

$$F = A + B + C$$

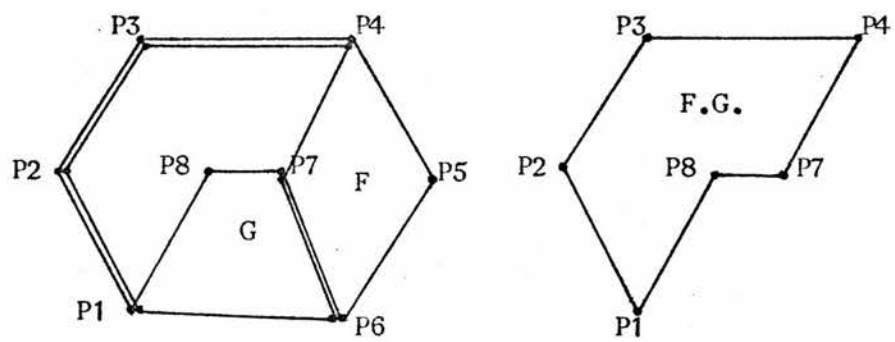
can be written. New composite zones can be defined as the areas with two properties in common for example:

$$F.G = (A+B+C).(A+B+D)$$

$$= A + B$$

The boolean operators are used in these operations in the same way that they were used in the PAX system described in Chapter 5.

An alternative approach to the definition of a composite zone such as F is to store its boundary line. The boolean expression F.G must then be interpreted into a new boundary description. This process is based on the straightforward comparison of components, in this case the co-ordinates of vertices in the two boundary lists, as shown in Figure 18



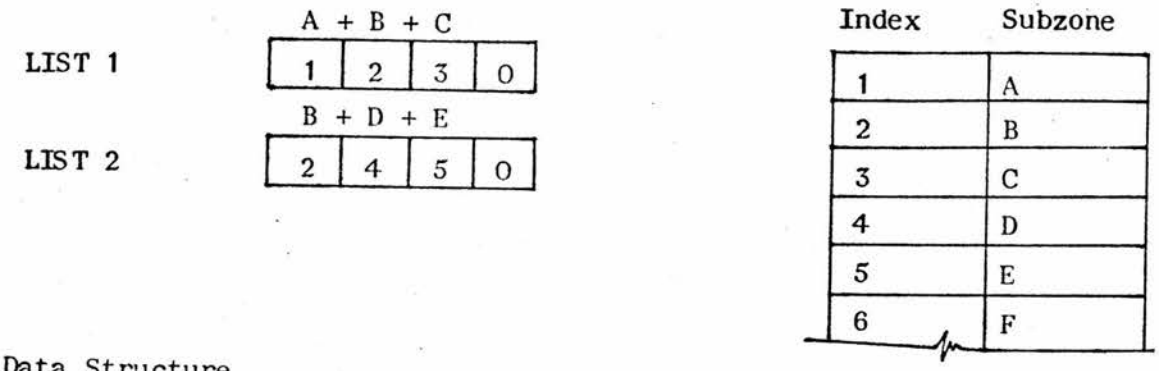
Defining the Intersection of Two Zones Using Polygon Boundaries

Figure 18.

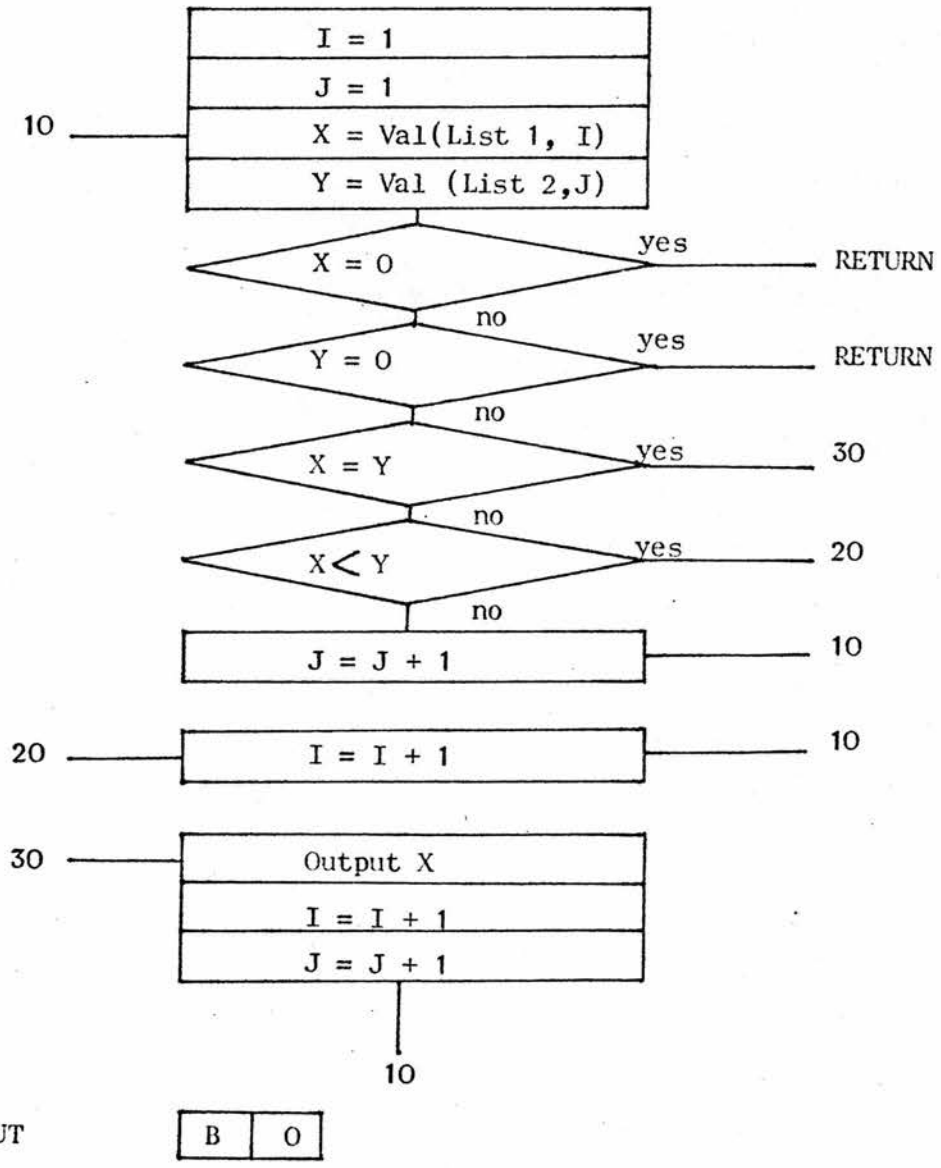
Both these approaches to defining new zones: either as the set operation on sets of subzones, or the creation of a new polygon boundary from the boundaries of two or more other boundaries, can be carried out as sequential operations on ordered lists of components as a form of merging procedure.

As an example, the basic procedure for obtaining the intersection of two zones expressed as an ordered list of subzones is given in Figure 19. The corresponding procedure for merging two polygon boundaries is more complex because of the way in which the lists have to be set up. The first stage is a specialised form of 'sort' which is discussed later, and which links identical points in the two files. The process also depends on the first vertex of each list being the smallest value in the ring of coordinate points: in the diagram this means the point lies furthest in the left direction. Once established this order must be retained by the merging operation in the same way that the order in the subzone pointer list is retained in the set intersection operation shown in Figure 19. What the order permits is an implicit use of the point in polygon test for the first point in each polygon list. The smallest coordinate of the two initial vertices in the two boundary lists can immediately be classified as lying outside the other polygon. It must be noted that the contrary statement cannot be made about the other point.

The process depends on any intersection point between the two polygon boundaries existing as identical coordinates in the two lists. Consider the three cases shown in Figure 20 where no such intersection points occur.



Data Structure



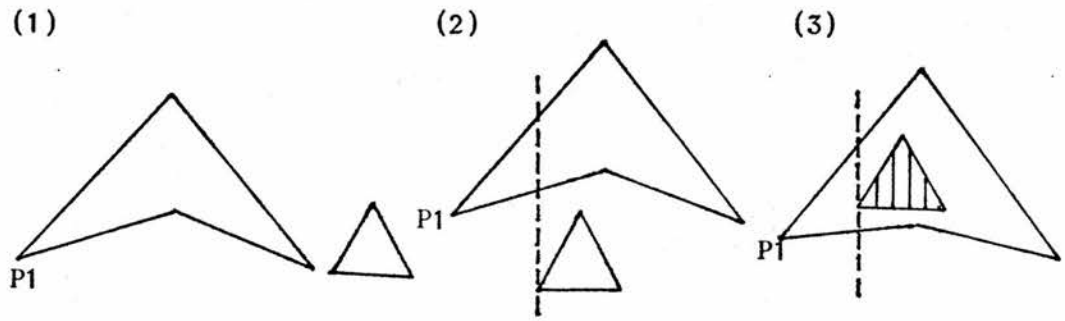
OUTPUT

B	0

Procedure for Intersecting Two Compatible Sets of Subzones[1]

Figure 19

1. See Appendix B. Subroutine Union: ENTRY INTER.



Intersection of Two Polygons Using Boundary Lists

Figure 20.

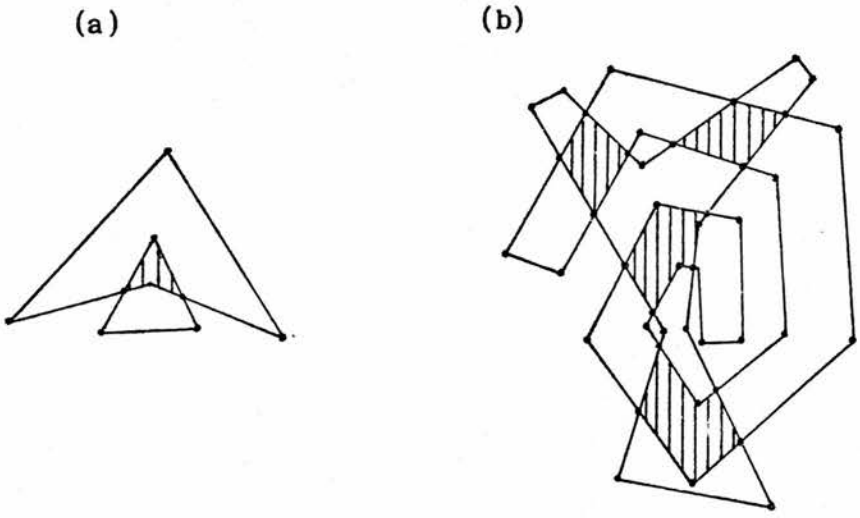
In case 1, P1 is the minimum coordinate and consequently lies outside the second polygon and will not occur in the boundary line for the area of intersection formed by the two polygons. If P1 belongs to List 1, then the rest of the points in this list may be sequentially processed to ensure that they do not match with points in List 2: in other words to show that there are no intersection points. In this case because none of the coordinates in List 1 are linked to identical coordinates in List 2, the points in List 2 can all be discarded. Having processed all the points in List 1 the problem is to decide whether the second polygon lies outside the first in other words case 1 or 2, or whether it occurs within the first polygon as shown in case 3. In the first two cases there will be no region of overlap, in the third the area of overlap will coincide with the inner polygon given by List 2.

In case 1 the second polygon will be discarded because its first co-ordinate lies to the right of all the coordinates in the boundary of the other polygon, which can be established while the first polygon list is being processed. It is therefore necessary to distinguish case 2 from case 3. Using the first coordinate in List 2 it is possible

to carry out a point in polygon test for this point while processing List 1. This can be done by counting the number of times the other boundary line crosses a vertical line through this point. The location of the first point of List 2 - inside or outside polygon 1 - will be established by the time that List 1 has been processed. Where, as in this case there are no boundary intersection points this information will be sufficient to indicate whether the boundary of polygon 2 can be discarded as in case 2 or forms the boundary of the zone of overlap as in case 3.

Where the boundary of two polygons cross as shown in Figure 21 then the original sort will already have linked the occurrence of the points of intersection in each boundary list. Processing must again start with the minimum coordinate and proceed as before, until an intersection point is found. The selection of the next point will then depend on which of the two points following the intersection point in the two boundaries lies 'inside' the other boundary. The common area to the two polygons can be determined by following the 'coinciding' points and the 'inside' points in the sequence in which they are found. The whole of the polygon list containing the starting or minimum point must be processed to ensure that multiple zones of overlap do not exist as shown in Figure 21b.

It became clear that there were several ways in which the definition of new zones could be created from the overlap of existing zones. The final choice had to be left open until more of the system structure had been established. The first process, which used named building blocks, was the one preferred, but the second process appeared simpler to implement



Zones of Overlap where Polygon Boundaries Intersect

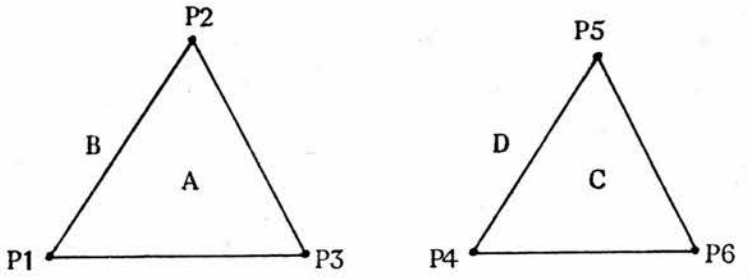
Figure 21

in a small system. A third alternative which was examined was a modification of the second approach. In this case polygon boundaries were not created directly but line segments of the new zone were output as they were located. Line segments created in this way in random order, had to be ordered and concatenated as described in Figure 3 to give new polygon boundary lines. However, since this process is the one adopted for creating contour lines it could well be reasonable to adopt it for this purpose as well.

In all the cases considered so far, it has been assumed that the intersection points where polygon boundaries cross, will already exist in overlapping boundary lists of coordinates. This depends on the way in which the original data were prepared. It assumes that all the possible boundary lines which might at some time be used must be overlayed on a single base map and the resulting minimal zones be digitised as separate polygons. Apart from this being a tedious process, it is

in fact unlikely that the data requirements will be static enough for this to be possible. If one set of boundary lines were changed there would be a major redigitising and editing task to redefine the basic zones in the data-base. A more versatile approach was to define the boundaries of zones with a common property, as independent polygons and then allow the system to determine whether these boundaries intersect the boundaries of other zones, when areas with multiple properties are requested.

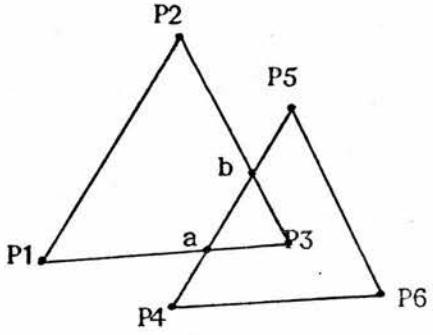
This was the approach adopted by Samuel Arms, and the first attempt to create an overlay routine was based on his technique. This consisted of taking two polygons from two networks and outputting the area of overlap. By carrying out the pair-wise comparison of polygons in the appropriate order, all the new sub-zones could eventually be created. This procedure is similar to the previous boundary overlay process, except that the new boundary intersection points have to be established at the same time. The way that this program was designed is summarised in Figure 22, and the use of the dual naming of line segments used in GIMMS is also shown. The procedure was to take the boundary of one polygon and to divide up the boundary into the sections which lay within polygons of the second network, adding the second polygon name to the first polygon line segment names. When this process had been completed for all polygons the boundaries were sorted into new polygon boundaries based on the new multiple names.



A	B	B	B
P1	P2	P3	P1

C	D	D	D
P4	P5	P6	P4

Original Polygon Definitions



A	BD	BD	BC	BC	BD
P1	P2	b	P3	a	P1

C	BD	AD	BD	BD	BD
P4	a	b	P5	P6	P4

AD	BD
P1	P2
AD	BD
P2	b
AC	BC
b	P3
AC	BC
P3	a
AD	BD
a	P1

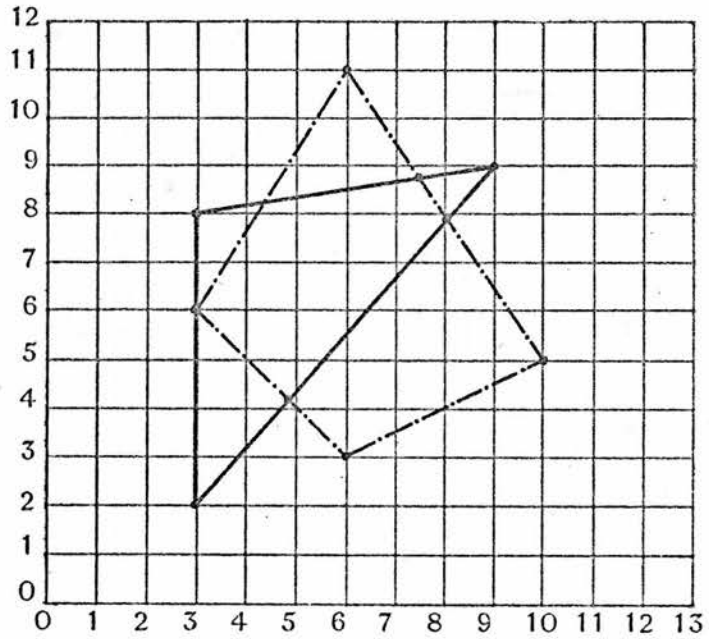
BC	BD
P4	a
AC	AD
a	b
BC	BD
b	P5
BC	BD
P5	P6
BC	BD
P6	P4

Line Segments Created with Multiple Names

Figure 22

SYSTEM DEVELOPMENT

The example shown in Figure 23 gives the test data used with this program and the Tables below show the output. Table 2 shows the output in line segment form while Table 3 shows the data restructured to give new polygon boundaries.



——— Polygon A : Inside - 1, Outside - 2

—·—·— Polygon B : Inside - 1, Outside - 2

Test Data Polygon Overlay Program I

Figure 23

Table 1 Input Data Polygon Overlay Program I

1	2	2	2	
300000	900000	300000	300000	
800000	900000	200000	800000	
1	2	2	2	2
300000	600000	1000000	600000	300000
600000	1100000	500000	300000	600000

Table 2 Line Segment Output File

1	2	1	2
2	2	1	1
300000	800000	300000	600000
433333	822222	433333	822222
1	2	1	2
1	1	2	2
433333	822222	433333	822222
750000	875000	600000	1100000
1	2	1	2
2	2	2	2
750000	875000	600000	1100000
900000	900000	750000	875000
1	2	1	2
2	2	1	1
900000	900000	750000	875000
806250	790625	806250	790625
1	2	1	2
1	1	2	2
806250	790625	806250	790625
484615	415384	1000000	500000
1	2	1	2
2	2	2	2
484615	415384	1000000	500000
300000	200000	600000	300000
1	2	1	2
2	2	2	2
300000	200000	600000	300000
300000	600000	484615	415384
1	2	1	2
2	2	1	1
300000	600000	484615	415384
300000	800000	300000	600000

The first polygon overlay program was constructed so that its output would be compatible with the GIMMS system, either providing the line segment file shown in Table 2 , or the new polygon boundary files shown in Table 3 . Though this exercise was technically successful the resulting program was very slow. The general problem was the same as that encountered when designing the hidden line algorithm in OBLIX. Each line segment in one polygon had to be tested against all the line segments in the second polygon file.

SYSTEM DEVELOPMENT

Table 3 Polygon Boundary Output File

A.B

1001	2001	1002	2001	1002	2001
300000	433333	750000	806250	484615	300000
600000	822222	875000	790625	415384	600000

$\bar{A}$.B

2001	1001	1001	2002	2002	2002	2002
433333	300000	484615	300000	300000	300000	433333
822222	600000	415384	200000	600000	800000	822222

2001	1001	2002	2002
806250	750000	900000	806250
790625	875000	900000	790625

A.$\bar{B}$

1002	2002	2002	1001
433333	600000	750000	433333
822222	1100000	875000	822222

1002	2002	2002	2002	1001
806250	1000000	600000	484615	806250
790625	500000	300000	415384	790625

$\bar{A}$.$\bar{B}$

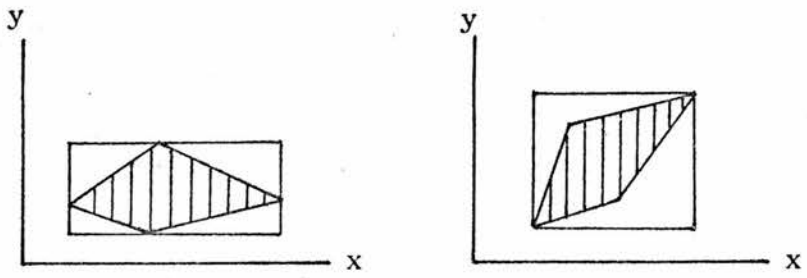
2002	1002	2001	2001	2001	2001	1002	1002
600000	433333	300000	300000	300000	484615	600000	1000000
1100000	822222	800000	600000	200000	415384	300000	500000

1002	2001	2001	1002
806250	900000	750000	600000
790625	900000	875000	1100000

Consequently there was a larger number of tests showing that lines did not intersect than there was successfully locating crossing points. One approach to reducing the computation of the overall process was to reduce these negative tests. This could be done by making the first part of the test some simple process which, as a first priority, would throw out lines which could not cross. If the data could be composed into higher order elements which had global properties suitable for such a test, then the preprocessing required to set up the more complex

data structure might well result in overall reductions in the total time of computation. An example of this approach in the case of polygon boundaries was provided by the creation of envelope boxes. If the extreme coordinates of a complex polygon are found by one pass through the data, then this box can be used to test whether a line segment is in the neighbourhood of the polygon in one operation, instead of the many individual line-pair tests required by a direct approach. Where a particular line segment does cross the polygon boundary then for this line rather more work will have to be done than using the direct method before the intersection point is established, but this will be more than compensated for by the time saved carrying out the other tests. The use of global data properties in the study of hidden line removal algorithms has been called the use of 'scene coherence'. This strategy assumes that the probability of a direct test failing is greater than for it to succeed; if the outcome one way or the other is of equal probability this preprocessing becomes less productive.

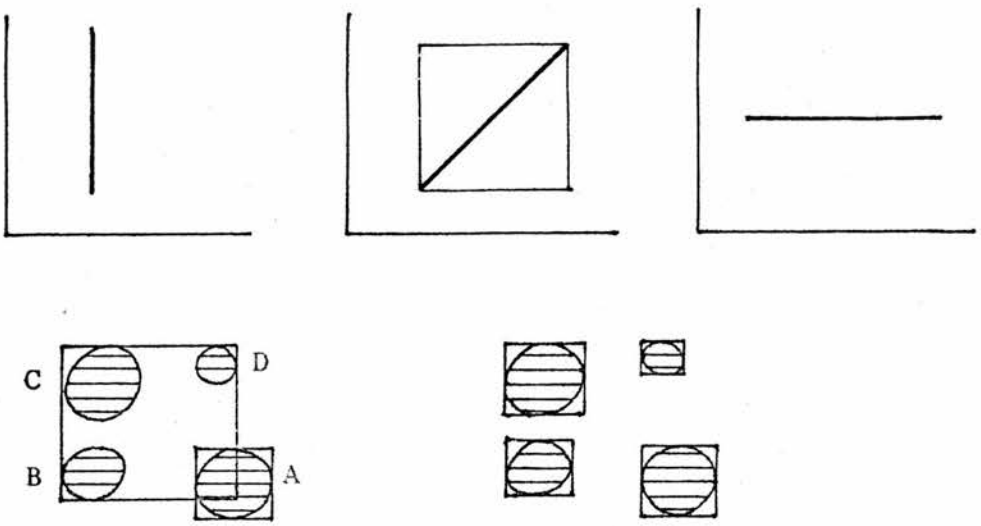
It was from a more careful study of the use of box-envelopes that a different approach to a polygon overlay program was evolved. In a plane the simplest envelope to form was a rectangle. If the rectangle's sides were parallel to the coordinate axes then the rectangle could be summarised by four values: Minimum x, Maximum x, Minimum y, and Maximum y. Such an envelope obviously depends on the relationship between the polygon and the coordinate axes. If the polygon is rotated relative to the axes it is possible to change the area covered by the envelope rectangle, as shown in Figure 24.



Changing Rectangular Envelopes

Figure 24

The first question posed in this examination was what elements should be grouped together within a single envelope. On one hand it was clear that the envelope for a straight line segment would be the same as the end points of the line. The only difference would be the interpretation made of the four numbers. On the other the spatial coverage of the envelope would determine to a great extent how useful it would be in reducing line intersection tests. The coverage of the same length of straight line in different orientations to the axes, range from zero area to a maximum when the line is at forty-five degrees to the axes. Similarly, the effect of grouping polygons into a single envelope is shown in the first example of Figure 25 to create a situation where polygon A will have to be tested against polygons B, C, and D because the two envelopes overlap. In contrast the second example shows that the creation of four separate envelopes would have immediately indicated that no intersections between these polygons were possible. Somewhere between the two extremes of taking separate envelopes for each line segment and having a single envelope which includes all the polygons, there must lie an optimal strategy.

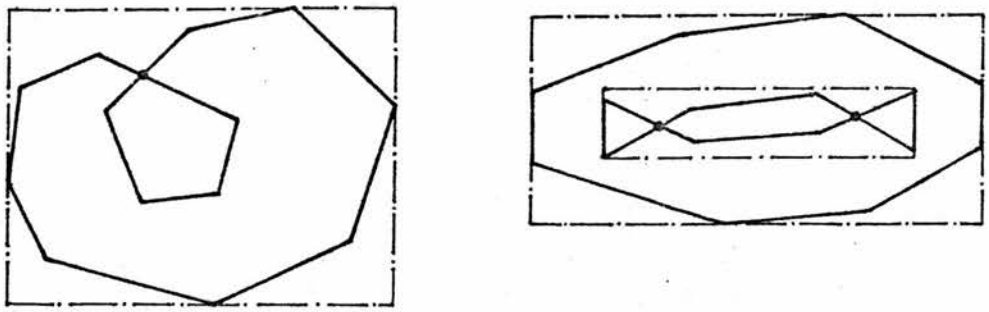


Envelope Coverage

Figure 25.

The starting point for developing such a strategy was found by linking together the results of two separate studies. The first was the solution to the restricted hidden line problem which was used in the OBLIX routine, the second was a study where a more general approach was taken to the hidden line removal problem. In the latter investigation, an attempt to find a simple way of locating the intersection points in the self crossing contour line which separated front facing surfaces from back facing surfaces, showed that it was insufficient to merely form envelopes for closed boundary loops. It can be seen in Figure 26 that the envelopes shown will not contribute in any way to locating the three pairs of crossing line segments.

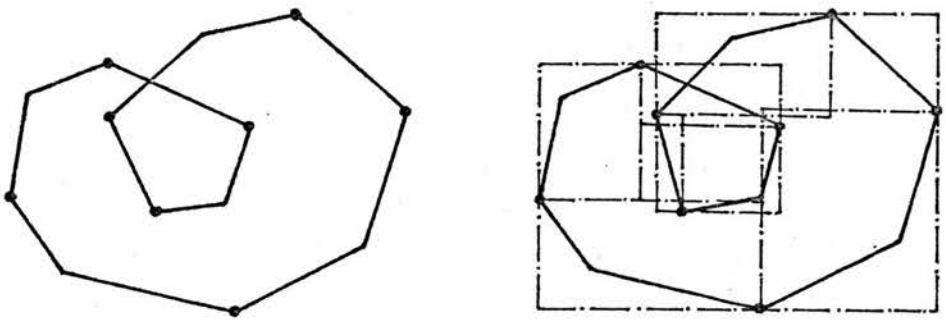
If these line boundaries are divided up into sections so that point coordinates within the sections are either consistently increasing in their x or y values or consistently decreasing, then the envelopes of these line sections can be created from the coordinates of the



Envelopes Created from Self Crossing Hidden Area Contours

Figure 26

stationary points in the line boundary. It can be seen from the diagram in Figure 27 that this gives a small improvement in that not all the line segments in the boundary line would have to be tested against each other, but there are still many tests which visual inspection would suggest were unnecessary.

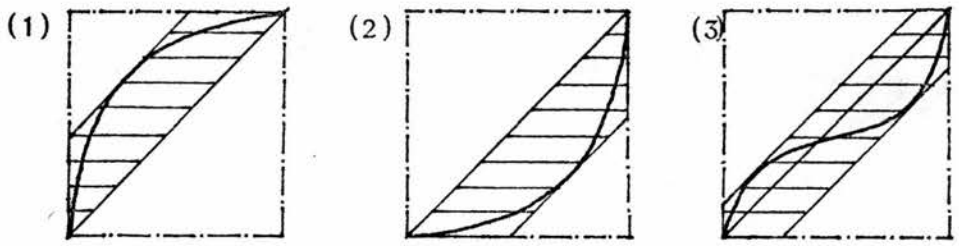


Multiple Envelopes for a Self Crossing Boundary

Figure 27

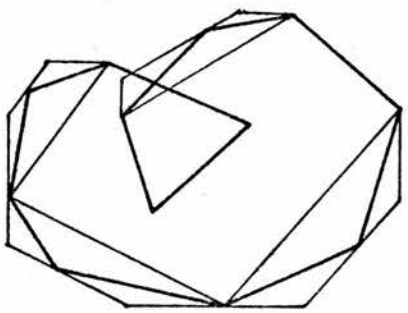
A possible line of improvement came from a modification of the box envelope. If the sections of the boundary lines between stationary

points are examined it becomes clear that about half the area covered by the box envelope cannot contain the line. This is because the change in x or y coordinate values, moving from point to point within the section, must always be either consistently positive or consistently negative. In simple cases shown by (1) and (2) in Figure 28 this means that the line will fall totally to one side or the other of the box envelopes diagonal. Even in case (3) the coverage of the line's envelope can be considerably reduced. If two lines are introduced parallel to the box diagonal passing through the extreme points of the curve then the envelope is reduced to the shaded zones within Figure 28



Reduced Envelopes

Figure 28



Reduced Envelopes Applied to the Self Crossing Contour

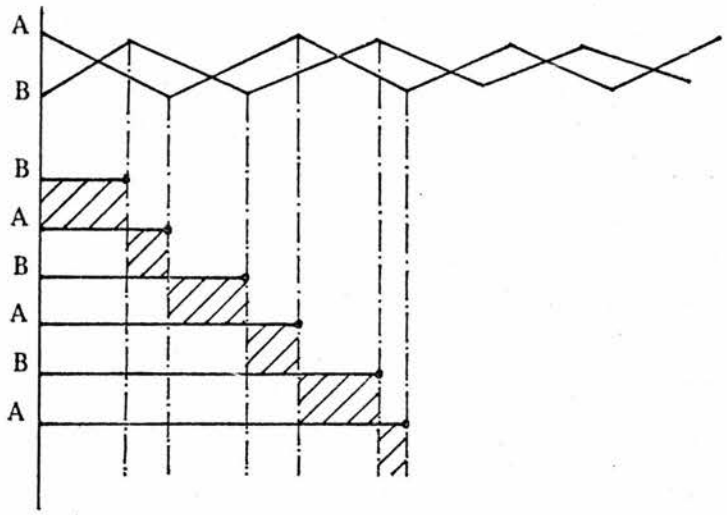
Figure 29

SYSTEM DEVELOPMENT

From Figure 29 it can be seen that the number of line crossing tests will be reduced to two by applying the reduced envelope. However, it is doubtful in this case whether this is an improvement since the minimum number of values which it seemed necessary to store for these more complex envelopes was six numbers, and in each case where they are used in this diagram they merely replace three coordinates. It appeared that better results might be achieved if the line sections had contained a larger number of segments. This approach was not taken any further because a different line of development seemed more promising, though the basic idea was not abandoned completely as a possible area for future development.

The definition of these complex envelopes bore some relationship to the convex hull of a set of points. This construction was useful where it was necessary to find the shortest route through a collection of objects, for example a path between buildings. The convex envelope of a complicated building form is a simpler data structure to work with than the total building description, and at the same time it removes many of these cul-de-sac zones where the path could not be placed but otherwise which would have to be tested to establish this fact.

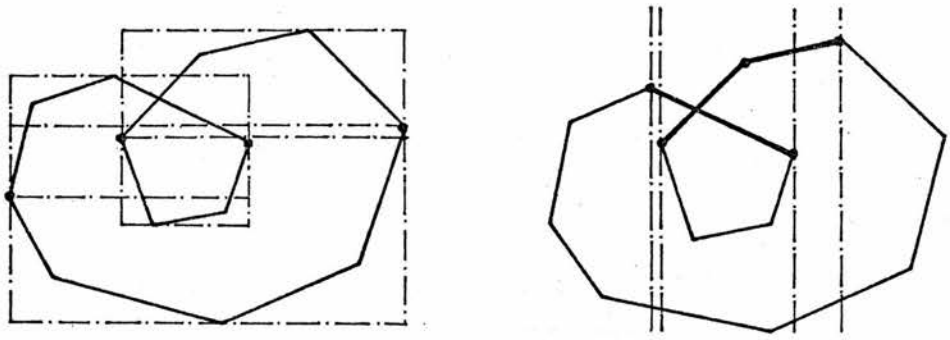
In the OBLIX hidden line routine, many unnecessary line crossing tests were avoided by using the order in the coordinate lists of the two lines being processed, so that line crossing tests were localised within a vertical strip. By incrementing the lagging line coordinate to give the next line segment to test, this vertical strip was kept to a minimum width.



Incrementing Strategy in OBLIX

Figure 30

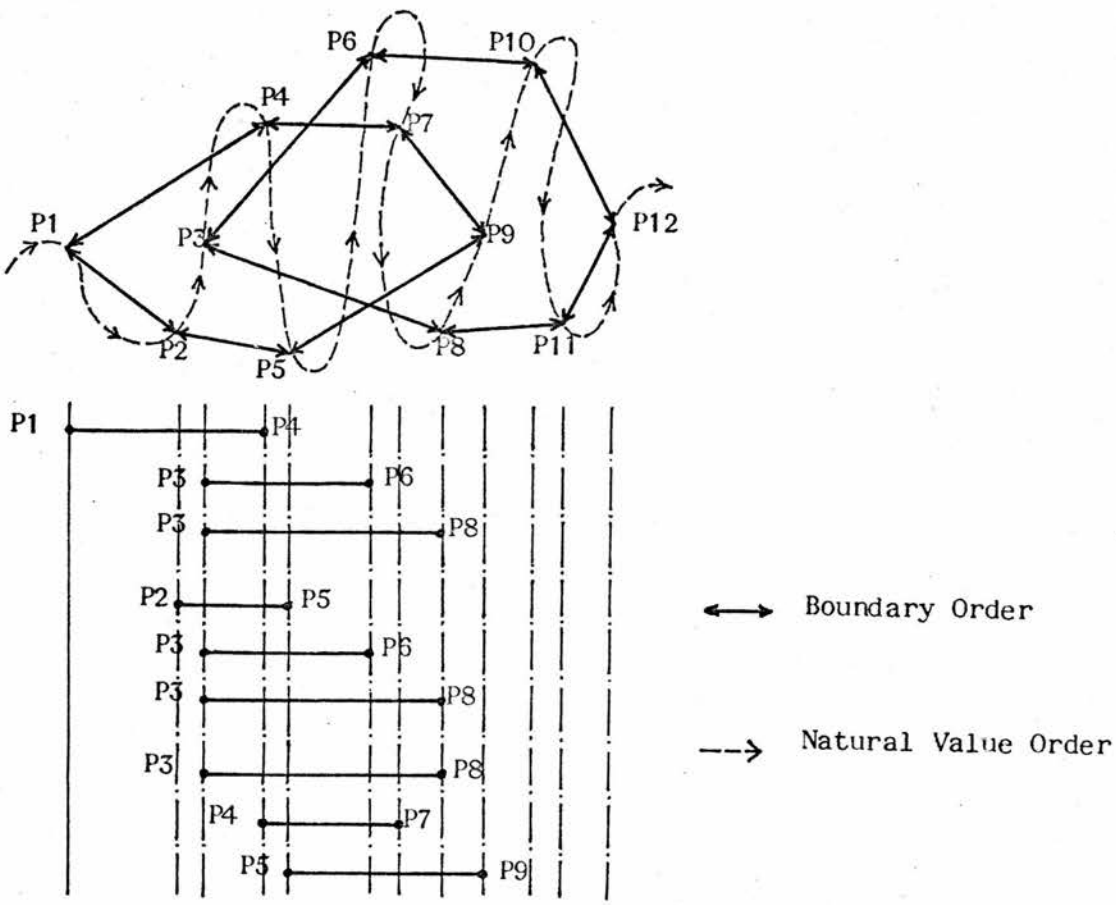
It was realised that this approach could be productively applied to the self crossing contour. To do this only involved finding the 'x' stationary points in the boundary, so that line sections were obtained which were either consistently increasing or decreasing in the x direction. The diagram in Figure 31 shows that this gives four elements which could be compared together as box envelopes, and where these overlap the line crossing tests within these envelopes could be reduced using the OBLIX strategy. It was while this approach was being



OBLIX Strategy Applied to the Self Crossing Contour

Figure 31

investigated that a more general way of implementing this strategy became apparent. If the two polygon boundaries shown in Figure 22 are considered as coordinate lists, then contiguous points in the list will represent line segments in the boundary lines. Where collections of these lines correspond to the kind of envelopes which have just been discussed, this order will also be the natural value order of the coordinates. Consequently, it is possible to create a linked list data structure giving the value order for all the coordinates in a relatively simple way. The result is shown diagrammatically in Figure 32.



Extended OBLIX Strategy

Figure 32.

The natural order of the coordinates is used to give the line segment linkage, the pointer order is used to give the sequence of testing. The first point is P1. This is linked to P2 by the value order pointer. P1 is linked by its natural order to P4 so the first line segment is P1P4. P2 is linked to P1 and P5 as line segments, however both these segments are in the same boundary as P1P4. If P2 is incremented using the value order pointer it gives P3. P3 is linked to P6 and P8 as line segments, so line crossing tests between these two line segments and P1P4 must be carried out. If this process is continued then the following line crossing tests will result:

P1P4 against P3P6 and P3P8

P2P5 against P3P6 and P3P8

P3P6 against P4P7 and P5P9

P3P8 against P4P7 and P5P9

P4P7 against P6P10

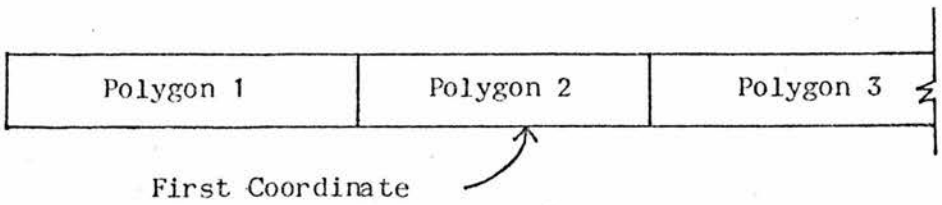
P5P9 against P6P10 and P8P11

P6P10 against P7P9

P7P9 against P8P11

This is a total of 13 tests in contrast with 36 tests if each line were tested against all the line segments in the other polygon.

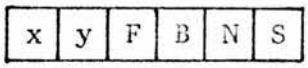
The operation of the program which was created to use this algorithm for polygon overlays can be described in three stages. The first stage was to place all the polygons which needed to be overlayed into a single file with the format shown in Figure 33. Each polygon consists of the boundary points occurring in their natural boundary order, so that points which are neighbours in the boundary are contiguous in



File Format for Polygon Overlay

Figure 33

the file. The points are represented by coordinate records and the format for these records is given in Figure 34. The first process is to create the various pointer linkages between the coordinate records. This structure is the one used in the first version of this program and it may well be possible to stream-line it in the future.



x, y	Vertex Coordinate
F	Pointer to the next vertex in the boundary, unless it is the last vertex in a polygon list.
B	Pointer to the previous vertex in the boundary, unless it is the first vertex in a polygon list.
N	Next coordinate in the Value Order list of Coordinates.
S	Coordinate Set Name.

Format to a Coordinate Record

Figure 34

The output to this first stage for the previous example would be as follows:[1]

1. See Appendix B. Subroutine CHSRT.

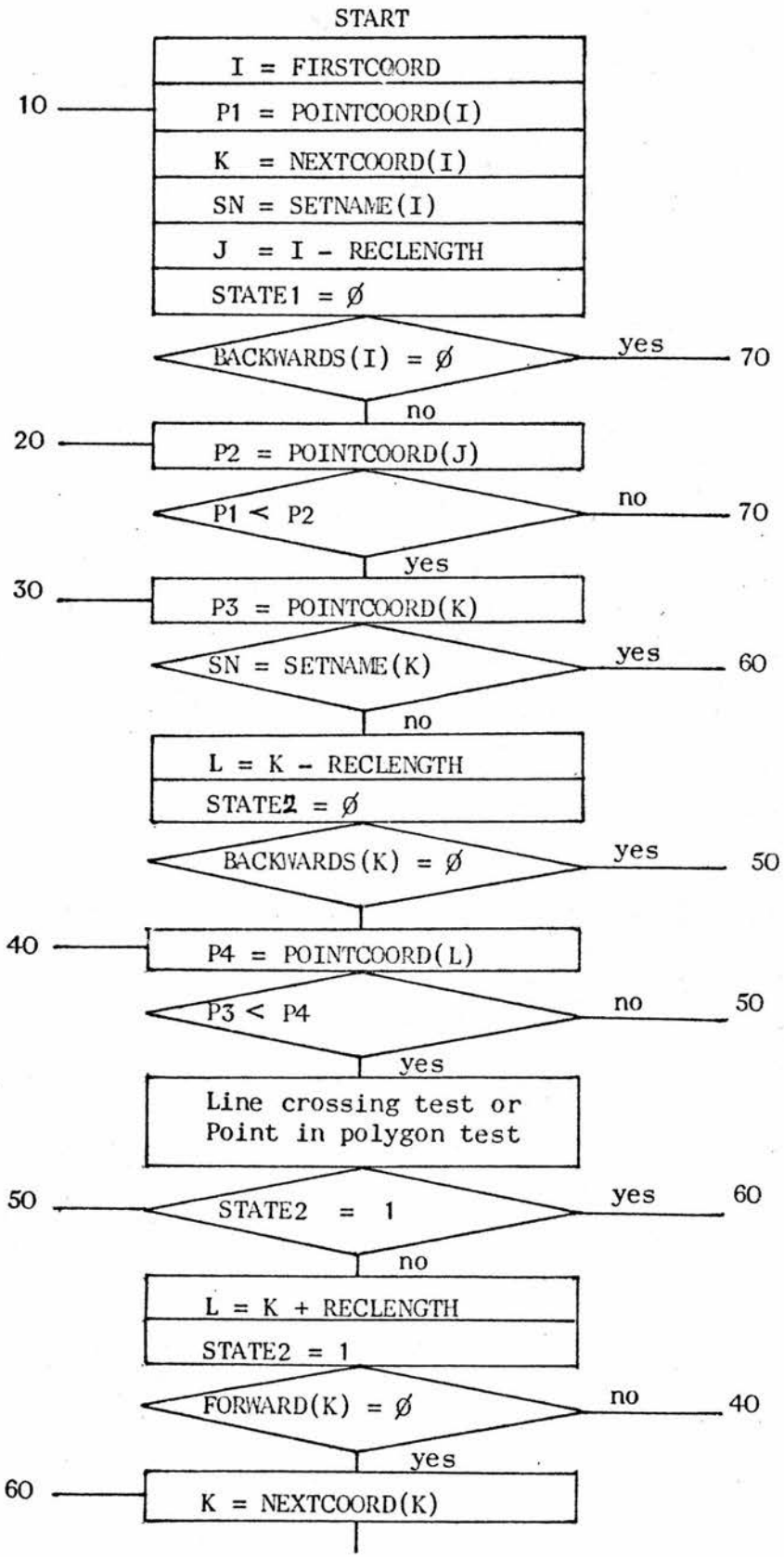
Table 4 Intermediate Polygon Overlay File Example

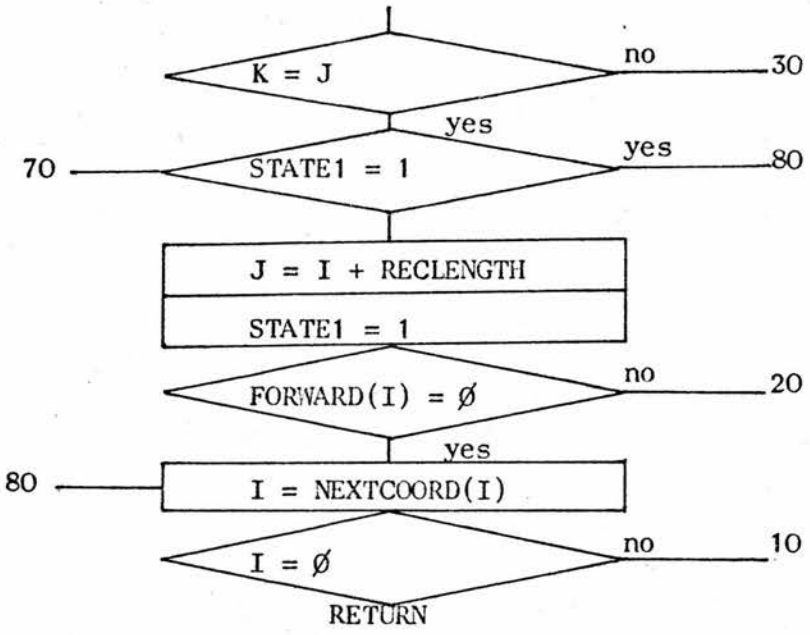
Coordinate Record Index:	x,y	F	B	N	S
1	P1	2	Ø	7	1
2	P4	3	1	5	1
3	P7	4	2	13	1
4	P9	5	3	10	1
5	P5	6	4	9	1
6	P2	7	5	8	1
7	P1	Ø	6	6	1
8	P3	9	Ø	14	2
9	P6	10	8	3	2
10	P10	11	9	12	2
11	P12	12	10	Ø	2
12	P11	13	11	11	2
13	P8	14	12	4	2
14	P3	Ø	13	2	2

The second stage consisted of processing this file in the way described previously. The detailed working of this section of the program is simplest to express as a flow chart. In Figure 35 the one or two character names are local internal variables, the longer mnemonic names refer to items in the data file. These names are:

FIRSTCOORD	Pointer to the smallest coordinate
POINTCOORD (I)	(x,y) from coordinate record
NEXTCOORD (I)	N for record I
FORWARDS (I)	F for record I
BACKWARDS (I)	B for record I
SETNAME (I)	S for record I
RECLENTH	Length of the coordinate record

SYSTEM DEVELOPMENT





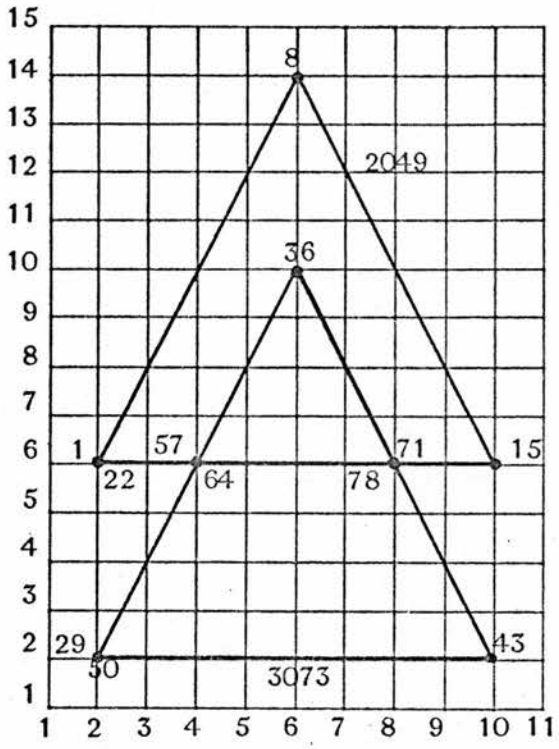
Line Segment Selection Algorithm for the Overlay Operation[1]

Figure 35

This flow chart represents the line segment selection strategy. The actual modification to the input data structure was carried out when two line segments were found to cross within their length. There appeared to be a variety of ways in which this could be done. One of the possibilities was implemented and the output produced for the two overlapping triangles is shown in the file given below them in Figure 36.

It can be seen that the coordinate record format is different from that given in Figure 34. The only new element in the record is 'R', the ring pointer which links identical coordinates. It was found necessary to insert two records for each intersection point, one for each boundary list, and these ring pointers were created to link them together. They also linked together all the identical first and last

1. See Appendix B. Subroutine COMPR.



N	x	y	S	R	B	F

Coordinate Record Format

N: Value Order
List Pointer

S: Set Name

R: Matching Points
Ring Pointer

B: Back Polygon
List Pointer

F: Forward Polygon
List Pointer

Output Stage II, Modified Coordinate Record File

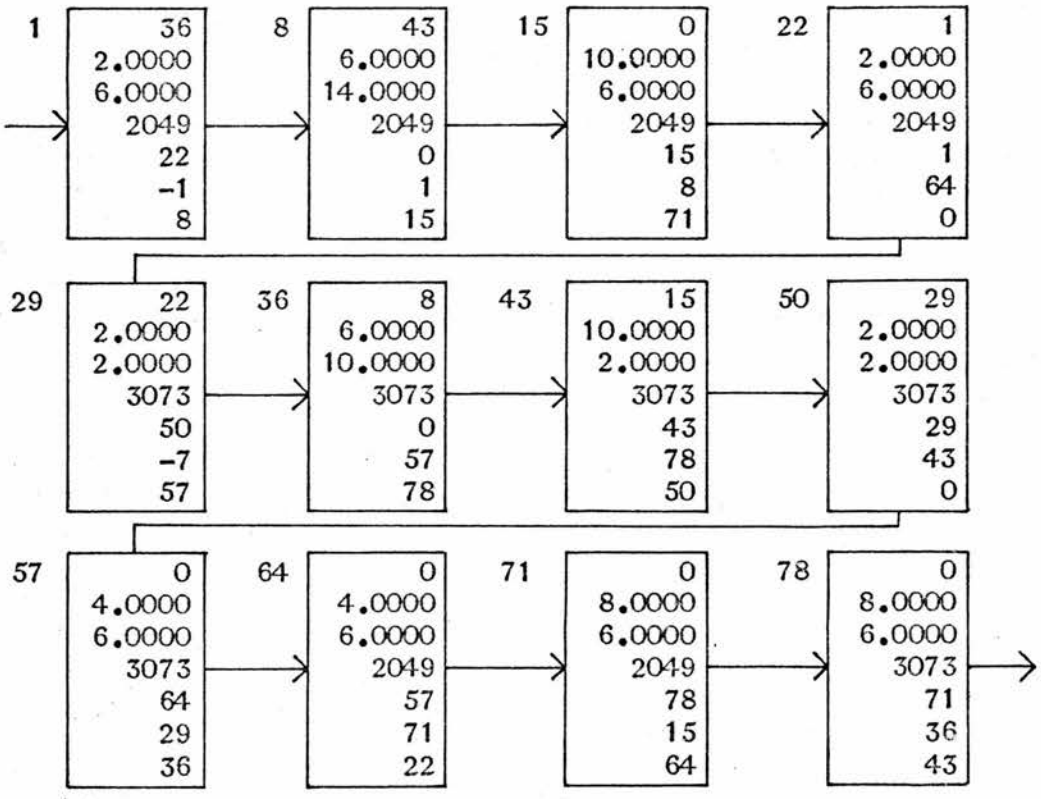


Figure 36

points in boundary lists, and it is interesting that had ring lists been used for boundaries, this would not have been necessary. Though this structure provided redundant information, it made some of the earlier experimental programming easier to carry out. It appeared possible that at a later stage some of the duplication could be reduced and file-space saved as a consequence.

The separation of the line segment selection routine from the line crossing routine which modified the input data resulted in a more versatile program structure. Though originally the aim was to overlay one network of boundaries on another network of boundaries, it was found that by using the appropriate set names a collection of polygons could be overlayed simultaneously. This provided a very great improvement over the previous strategy which was based on the pair-wise comparison of polygons. Only line-segments which have different set names are passed to the line crossing routine, so that if two networks are being overlayed, only two setnames are required. The mutually exclusive zones in a network need not be tested against each other. Where a collection of polygons is being processed, and the relative locations of each polygon in space is not known, then a different set name is required for each polygon.

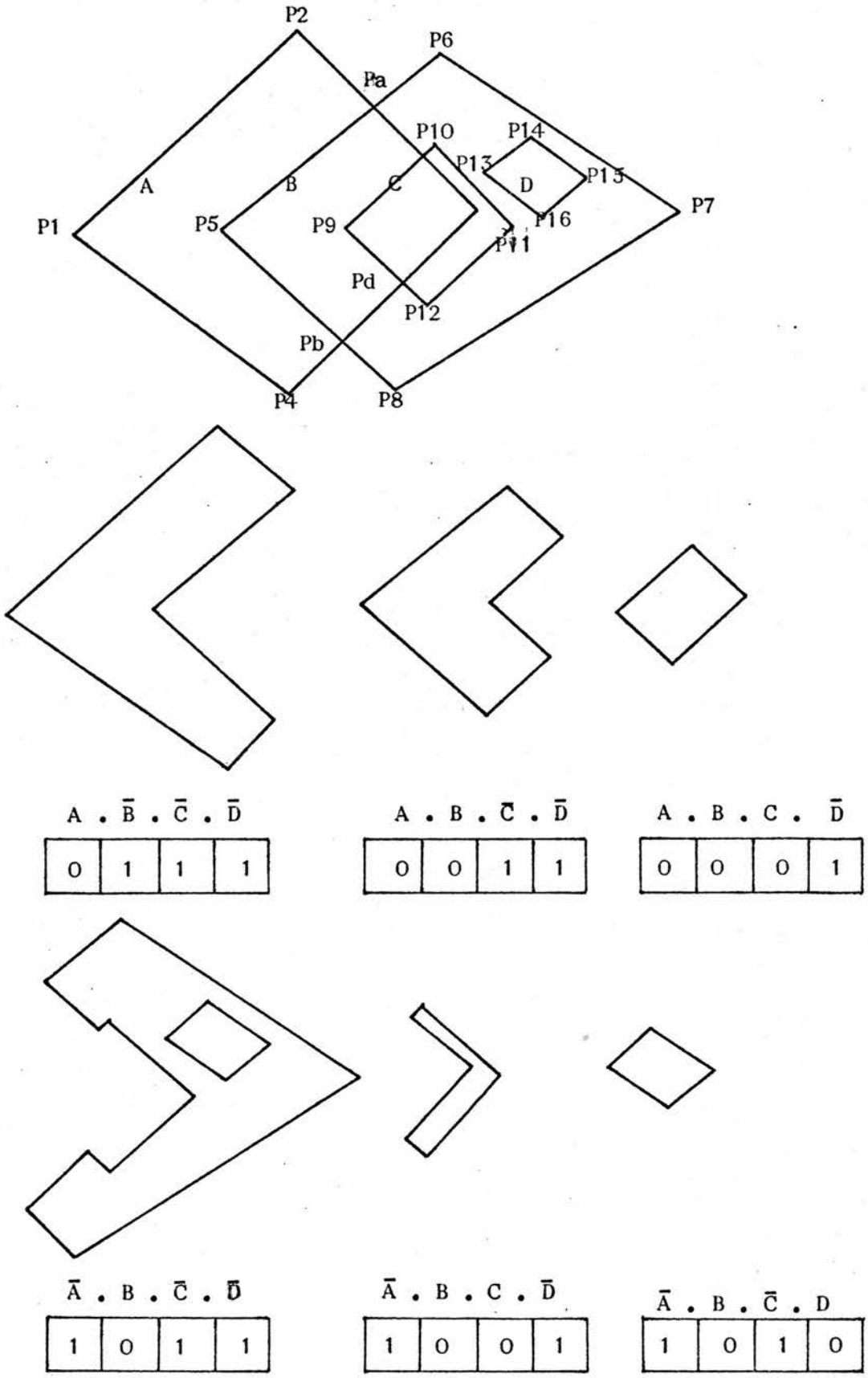
The reason for duplicating the intersection points was to simplify the procedure when a line segment was intersected at one of its vertices. When this occurred, a copy of the vertex was inserted in the list of the second boundary, and the pointer R changed in the original vertex record. This arrangement made it possible to use this program to link line segments from an input procedure such as that

described for GIMMS, and to then produce polygon boundary lists relatively quickly. Since the basic procedure is acting on line segments it is also possible to use the process for overlaying line files on polygon files, which had been an unresolved problem in the original version of GIMMS - lines having to be represented as double sided, or in other words zones of zero area.

The separation of the line segment selection routine from the data manipulation routine led to a further development. If a file of points is treated as a line and instead of working out all the intersection points the vertices of the psuedo line are tested for being below other line segments in the vertical line through them, the result is a multiple-point in polygon-set routine. This gives the ability to overlay a set of points on a set of polygons in one relatively efficient operation.

The final stage of this program depends on the final data structure required. For many operations the intermediate list structure just described can be used as it stands. However, for storage purposes it is convenient to compact this data structure. This can be achieved in a variety of ways. It is possible to create boundary lists for the smallest zones as described in Figure 37 . It is possible to output the boundaries of the originally named zones merely including the new intersection points in the correct position in the lists, and finally it is possible to output sets of boundary line segments with appropriately modified names.

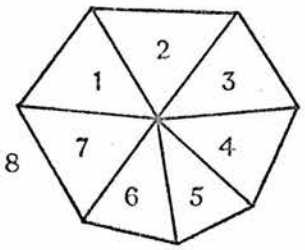
The reverse process to that shown in Figure 37, where boolean arrays are used as the new names for subzones, can be taken to naming zones



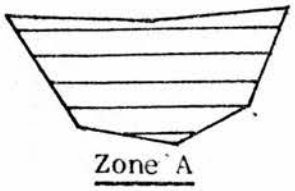
Naming Subzones Created by Polygon Overlay

Figure 37

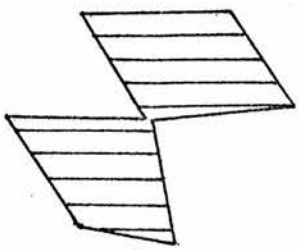
in a network. If the zones in Figure 38 are considered, there are eight zones which would require an eight bit name if the same approach used in Figure 37 is adopted. However, it is already known that these zones are mutually exclusive. Consequently, only eight numbers are needed to distinguish them; in other words, a three bit name. If the zones in Figure 38 are sequentially numbered from 1 to 8, then the binary representation of these numbers can be interpreted as the intersection of three composite zones.



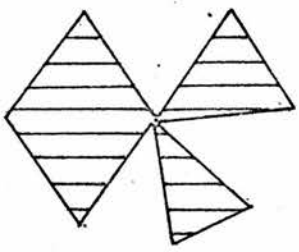
	A	B	C
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	6	0
7	1	1	1
8	0	0	0



Zone A



Zone B



Zone C

Compacting the Names of Mutually Exclusive Zones

Figure 38

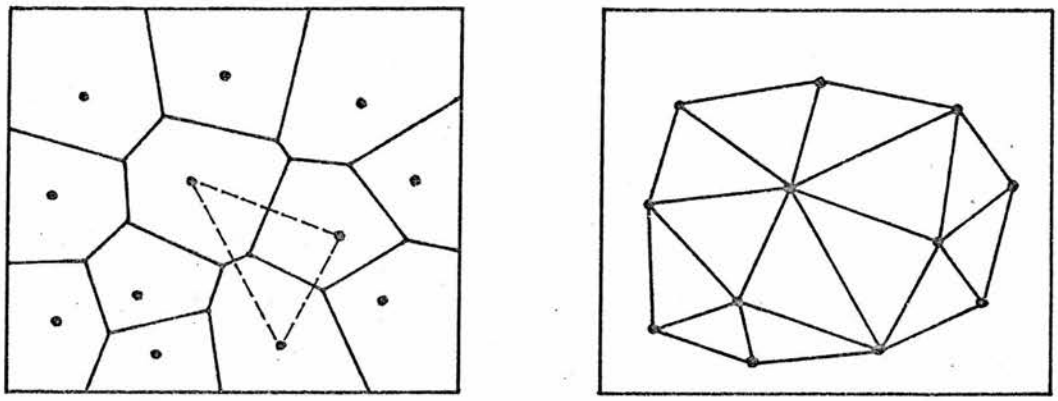
SYSTEM DEVELOPMENT

The conclusion drawn from Figures 37 and 38 , was that a more manageable data base would be created if all the mutually exclusive subzones which were generated by the overlay operation were given sequential numbers as system names. These could then be used as a way of defining a minimum number of zones to store. All internal spatial operations to the system could then be carried out using the sets of subzones and the operation shown in Figure 17. The advantage of creating minimal names is that a 32 bit word can be used to reference 2^{32} separate subzones, while only 32 basic zone files need to be stored.

The next study on zones was the creation of Thieson polygons. As 'nearest neighbour' zones to a set of points in a plane, they naturally follow a discussion of mutually exclusive areas, since they are by definition a collection of non-overlapping regions. The study of these polygons originally came from an attempt to triangulate a collection of data points. It was realised that there were a large number of ways in which a set of points in a plane could be triangulated. A method was required which gave a unique triangulation, or at least one which was repeatable even if the set of points were rotated relative to the coordinate axes. Such a method was provided by first of all creating the boundary network of nearest neighbour polygons and then converting this to the dual network where the points instead of being the centres of polygons became vertices in the network.

The relationship between the polygons and the triangulation are shown in Figure 39. If the polygons are given the same name as the points then if two polygons A and B have a common edge A/B then the two

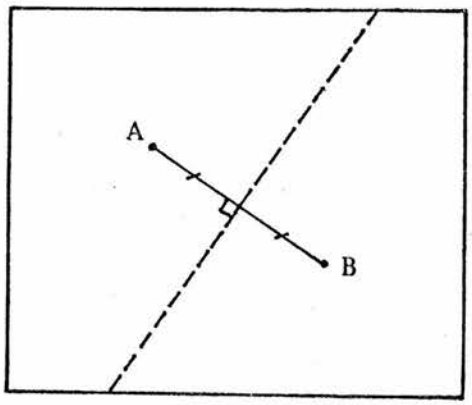
points A and B will have a link in the triangulation network. The only problem occurs when points form the vertices of a regular polygon such as a square. In this case the triangulation will be incomplete.



Nearest Neighbour Polygons and the Triangulation of Points

Figure 39

The problem was starting with a set of points, to construct an algorithm which could create the boundary lines of the nearest neighbour polygons with as little excess calculation as possible. In essence this meant using the known geometrical properties of the boundary network to the best advantage. If a set of two points is considered, shown in Figure 40, then the perpendicular bisector of the line joining



Nearest Neighbour Zones for Two Points

Figure 40

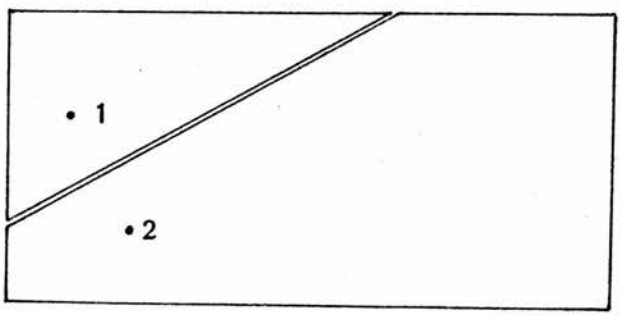
the two points defines the boundary of the two nearest neighbour zones. Any point on this boundary line is equi-distant from both points A and B. If a collection of points A to N are considered, then a point D will create the boundary lines A/D, B/D ...N/D. From the definition of the nearest neighbour polygon some of these boundaries will exclude others. What can be said for the final polygon for D is that any point inside it will lie within all these boundary lines. Consequently, the polygon can be defined as the intersection of all the spaces on the D sides of these lines. An attempt to create these boundaries by calculating all such lines for each point is obviously out of the question. After several attempts to use the known distribution of the points to reduce the calculations necessary, the following strategy was evolved.

Stage I

Sort all the coordinates of the points into order in the same way that the first stage of the overlay algorithm was to order the vertices.

Stage II

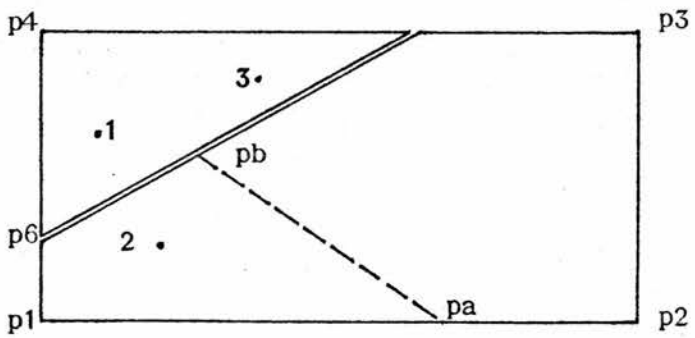
Start with the first two points: 1 and 2, and the existing boundary of the study area. Create the perpendicular bisector of the line 1/2, and use this line to divide the study area into two zones as shown in Figure 41.



First Two Points

Figure 41.

The next operation is to take point 3, and then process each outside boundary line segment in turn. These line segments will have names associating them with the point whose nearest neighbour zone they bound. By taking the referenced point and the new point, a new line which perpendicularly bisects the line linking them is constructed. All the line segments of the polygon which is being processed, are then tested with this line for an intersection point. If all the vertices of this polygon lie either outside or inside the line then there will be no intersection point and the next zone which butts onto the outer boundary line can be processed in the same way. In Figure 42 the situation is shown where an intersection point is found.

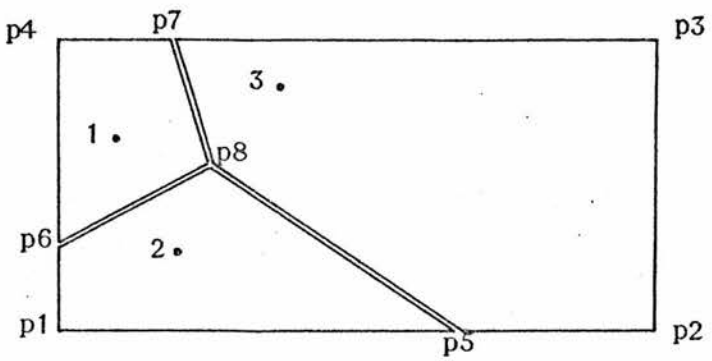


Adding the Third Point

Figure 42.

Once an intersection point has been found it means that the original polygon has to be divided into two sections. The first section belongs to the original point and the second section belongs to the new point. In Figure 42 the first line segment to the new polygon for point 3 is pb.pa. pa is the intersection point with the outside boundary but pb is the intersection point with the line 1/2. This indicates that the next polygon to be tested should be polygon 1. In this case there is in fact no alternative, but generally this use of line names will allow the new

boundary once its first vertex has been found on the outside boundary, to be systematically followed from one neighbouring polygon to the next until it again meets with the outer boundary. Figure 43shows the final result when point 3 has been processed.

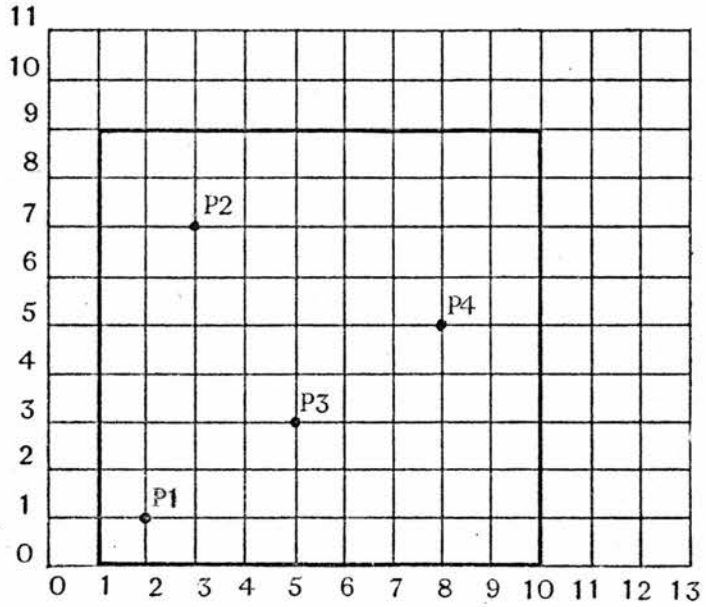


Three Zones

Figure 43

Once the polygon line segments for the new zone have been returned to the outer boundary, it is merely a bookkeeping task to complete the new polygon by inserting the necessary points from the outer boundary. For point 3 in Figure 43 these are P2 and P3. The polygons which were cut by the new boundary also have to be redefined in this process. This overall strategy depends on the points being taken in their coordinate order and the fact that the polygons formed will always be convex. Though some time is spent in creating intermediate results which are discarded in the final result, this process keeps the number of comparisons down to an acceptable level and can be contrasted with the original approach where all the points had to be processed for each new polygon.

The input data used to test this program is shown in Figure44 . It consisted of a study area polygon boundary and a set of four points as shown in the accompanying diagram. The output consisted of the



Test Data for Nearest Neighbour Polygons Program

Figure 44

Polygon for P1 Zone 1
2.1250
4.0625
5
4.8333
0.0000
9
1.1000
0.0000
0
1.1000
4.2333
0
2.1250
4.0625
5

Polygon for P2 Zone 5
5.3750
5.6875
13
2.1250
4.0625
9
1.1000
4.2333
1
1.1000
9.0000
0
6.7000
9.0000
0
5.3750
5.6875
13

Polygon for P3 Zone 9
5.3750
5.6875
5
9.1667
0.0000
13
4.8333
0.0000
0
2.1250
4.0625
1
5.3750
5.6875
5

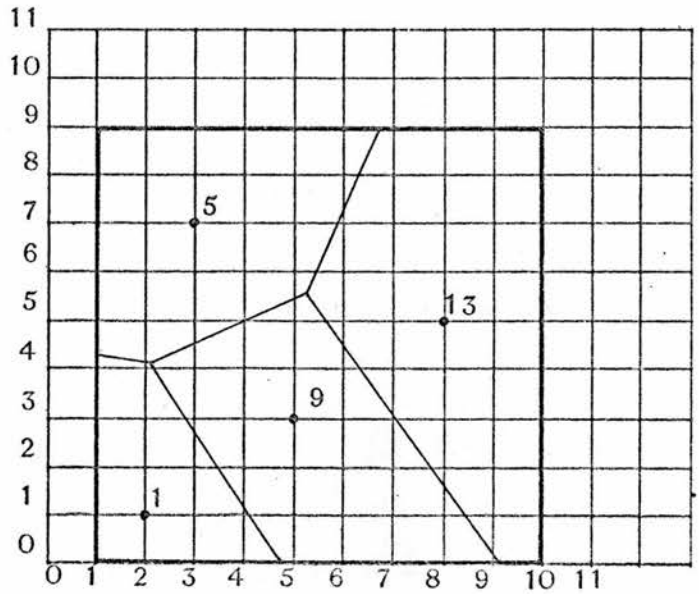
Polygon for P4 Zone 13
6.7000
9.0000
5
10.0000
9.0000
0
10.0000
0.0000
0
9.1667
0.0000
0
5.3750
5.6875
9
6.7000
9.0000
5

Nearest Neighbour Polygon Output Files for Test Data[1]

Figure 45

1. See Appendix B. Subroutine TRIAG.

four polygon boundaries shown in Figure 46. Each of the line segments in these boundaries is given the dual name created from the points it separates.



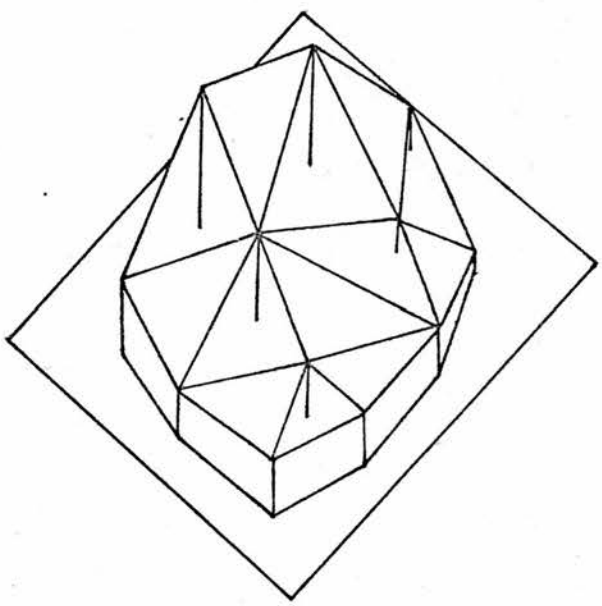
Nearest Neighbour Polygons from Test Data

Figure 46

From the file format shown in Figure 45 it can be seen that the triangulation of points has been achieved at the same time that the nearest neighbour polygons have been created. If all the neighbouring zone names for a particular polygon, say for point 1, are taken as line segment links, the result is a collection of triangles. The only problem would be caused by a pattern of original points which conformed with the vertices of a regular polygon. In this case, similar to the triangulation of the square cells used in contouring the block models, there will be several ways in which triangles can be formed and the choice will be arbitrary.

SYSTEM DEVELOPMENT

One of the applications for this triangulation process occurred in the creation of block models. The block models shown in the previous chapter were all generated from a regular grid of data points. There was no reason why irregularly spaced data points should not have been used. The problem was how to fit a surface through the top of the vertically scaled data values. The simplest solution was to triangulate the data points, because the projected triangles would always give a surface made up from plane facets.



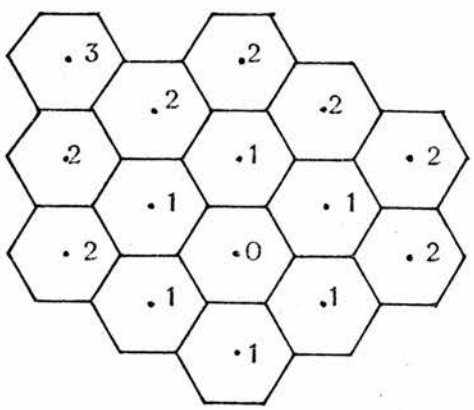
Triangulation Used to Create a Block Model

Figure 47

In Figure 47 a block model is shown generated from the points given in Figure 39. This example shows why this form of drawing is not used. It is virtually impossible to visualise the shape of this surface from the distribution of triangular boundaries. However, the contouring procedure described in the previous chapter for triangular patches can be easily applied to this irregular distribution of triangles, and the resulting

contour representation will give a much more readable drawing. The triangular prisms created by this process make it a simple task to calculate volumes, and the projected triangles can also be used to give surface area.

Once the file in Figure 45 has been created for a set of points, it can be used as the base for a variety of other tasks. The interpolation technique developed by Donald Shepard for SYMAP selects data points in the neighbourhood of the point which is being interpolated. Not only does the neighbouring polygon information associated with a particular zone make this into a direct way of finding these points, but it also ensures that points are selected evenly from all directions. Because of the structure of the polygon network it is possible to define first order neighbours, second order neighbours and so on, while at the same time taking into account the distribution of the points in space.



Localisation of Points Used for Interpolation

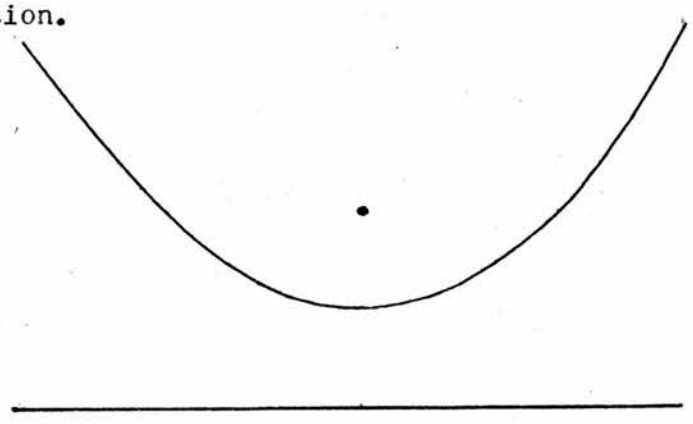
Figure 48

Though this would be a time consuming operation for a single

application, it is a reasonable approach where distributions of data points are used many times, since the network depends solely on the spatial relationships between the points.

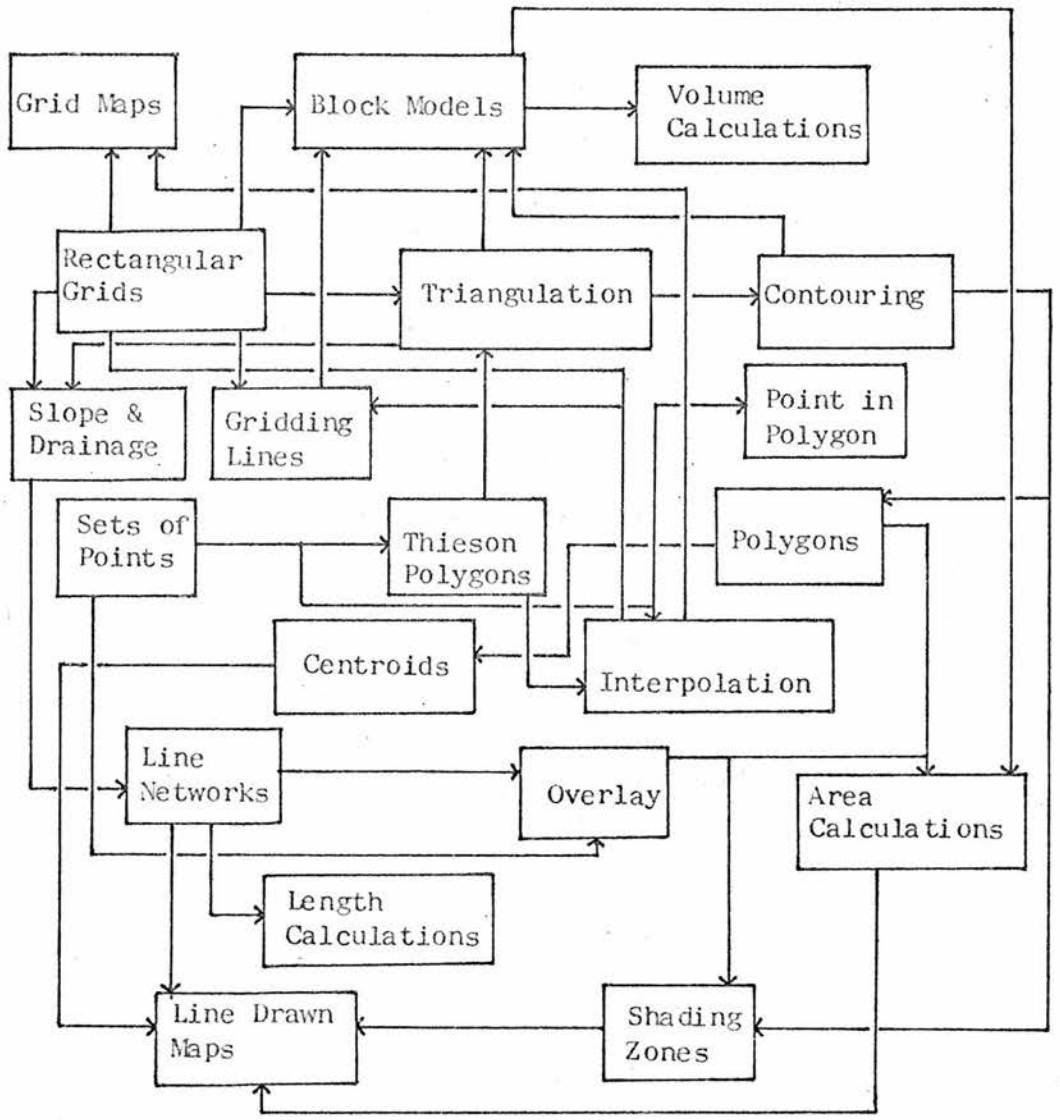
Not only points, but lines can be used to generate nearest neighbour zones. The boundary of the zone between two lines is the line which bisects the angle created where they cross. The zone created by a line and a point has a parabola as a boundary line. The skeleton of shape given in Figure 11 will be seen to be the network of nearest neighbour boundaries generated by the line segments of the original polygon's boundary. An alternative form of representation for a polygon can be constructed by choosing a set of points which, when processed to give nearest neighbour polygons, creates the original polygon network.

It can be seen that most of the processes described in this chapter can be used in combinations to create a wide variety of facilities, and that a general system structure is beginning to emerge. In Figure 50 the principal procedures which have been mentioned are loosely linked together showing the growing complexity in the possible flows of information.



Nearest Neighbour Zone for a Line and a Point

Figure 49



Developing System Structure

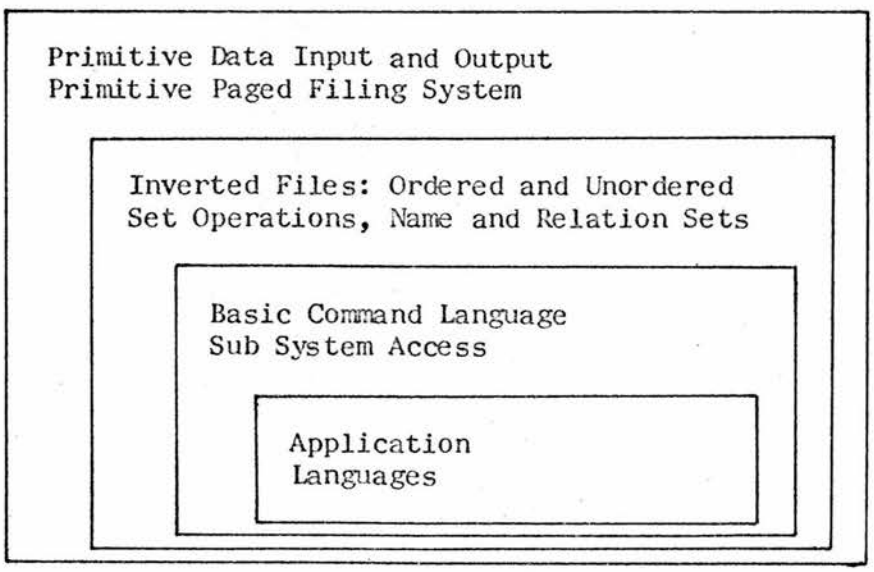
Figure 50

The names used in Figure 50 correspond to the high level description of the operations which have been under investigation. The programs in Appendix B give the collection of lower level procedures it was found necessary to implement in order to carry out these operations.

SYSTEM DEVELOPMENT

The programs in Appendix B are part of an incomplete system which is being developed as a research tool to continue the investigation started in this Chapter. It has been found necessary, several times, in the work so far presented to modify the design of this system, so the description given in the final part of this Chapter merely presents one stage in the work. Because the system was being developed as a research tool, an attempt has been made to keep its overall structure as general as possible. Though it has not been possible for practical reasons to always take this approach, an attempt has been made to close as few avenues for future development as possible. For example, though this is a single user system, in the background there has always been the idea that large data-bases must be accessible to many more than one user.

The routines given in Appendix B can be divided into a series of levels, hierarchically ordered in the schematic way shown in Figure 51.



General System Structure

Figure 51

The outer two levels contain operations which could form the basis for a very much larger system. In Appendix B the Main Program and the Subroutines:

Routine Name	Appendix Page Number
OPSS	2
RRFIL	3
XXFIL	4
NNFIL	5
ENTER(I,J)	54
INCT(I,J,L,M)	56
UINP(I,J,K,L)	57
INC(I,J,K,L,M)	60
COPY(I,J,K,L,M)	62
PSSH(I,J,K)	63
PUSH(I,J,K)	64
CLOSE(I)	65
DELETE(I)	66
INSERT(I,J,K)	68
PAGIN(I,J)	69
READS(I)*	70
PRINTS(I)*	70
SELOUT(N)*	71
NEXT(N)*	71
SPACES(N)*	71
NEWLIN(N)	71
PRINTR(X,I,J)	72
SHIFT(N,I)	73

provide the operations at the first level. Routines marked by an asterisk were written by T. C. Waugh, and were taken from the GIMMS program. Two forms of file are provided at the lowest level. The first has been called 'Open' and contains files which can have more data added to them so that they may grow in size. The second has been called 'Closed' and contains files which are of fixed length; these files can only be modified within their current size. The reason for this distinction was to be able to pack small files into the same page. Open files are given a whole page to themselves. Open files could be implemented without using linked list structures so that data could be accessed directly without having to follow the file entries from the beginning of the file. Files longer than one page even if they were closed files, were entered as open files so that this accessing simplification could be employed and only the page with the required entry need be called into core storage. This paged structure was developed from the experience gained in the GIMMS system. The data files needed in cartography could be very large and it was necessary to restrict the storage space in particular routines. However, the simple paging routines used in GIMMS were found to be inadequate for the procedure described in this Chapter. Though the development work described in the first part of the Chapter was mostly carried out in the automatically paged environment of EMAS (Edinburgh Multi Access System), which could have provided enough virtual memory to make these paged-file programs unnecessary, they proved to be useful on two occasions. Both the Overlay procedure and the Triangulation procedure were initially implemented in a way which caused page thrashing difficulties

and having accessible programs which could be monitored directly simplified the diagnostic problem.

The level above this basic filing, and input-output system, creates inverted files linked to Named entities. Operationally, these files appear to be name sets. For example, it is possible to request the file 'Farmers and Landowners'; where this has not been set up previously it can be constructed by intersecting the inverted files associated with the names 'Farmers' and 'Landowners'. The principal routine at this level is the SUBROUTINE UNION given on page 45 in Appendix B. However, the operation of this routine depends on the order of data values in a file, so included at this level is a series of Sorting and Merging programs to create the appropriate input data for the process. There are two data types defined at this level. The first is the so called unordered set which is implemented as a collection of values in numerical order.

The second is the so called ordered set or n-tuple which consists of values in any numerical order. These ordered sets contain values in a way that gives meaning to the position in which a value is found; in other words, they are a formatted structure. It is within the formatted data files that other data structures can be created, such as linked lists, matrices and so on, where the whole data set can be considered as a single object. The most important ordered element so far encountered is the coordinate record. The overlay procedure was designed to operate on a collection of coordinate records of the end points of boundary line segments from overlapping zones. This

procedure was a complex operation since the coordinates were treated at one stage as separate elements while, in the total procedure, the file of coordinates was treated as a single object.

The level above these set operations is the first level of which the system user is aware. This level provides a command language access to a series of subystems. This facility has not been fully developed, but the logic for its inclusion is that procedures can be implemented in a variety of ways, some more general than others. When handling large data files it is often faster to have a specially designed process for a particular operation which does not fit within a more general set of programs; in such a case it is useful to have the procedure available in an alternative subsystem. New programs tend to be developed for particular cases and then, as the processes involved are understood, they can be made more general. Each of the processes described in the first part of this Chapter was developed in this way, even though it was eventually aimed to include them in one subsystem. As the programs in Appendix B stand, there is only one subsystem linked into the calling programs which is the volume manipulation system described in the following Section.

It was in an attempt to link all the separate operations shown in Figure 50, which could be used at the command language level, into a single system with a coherent user language that led to the work presented in the following Section. The central theme chosen for more detailed analysis was the construction and use of the spatial descriptions of zones.

SECTION III

ZONES IN SPACE

CHAPTER 8

DEFINING SIMPLE VOLUMES

DEFINING SIMPLE VOLUMES

Following the work presented in the previous Chapters, describing two preliminary studies and a system structure which developed from them, this section examines one aspect of such a system. The main topic is the representation of zones in space. The study started by considering volumes, because many of the problems which occur in the use and structure of two dimensional models were expected to be solved by the simplification of techniques used in the higher dimensional space. Added to this there was also the fact that three dimensional models have more direct relationships with 'real space' than the other two.

Appel (1968) breaks down the problem of 'Modelling in Three Dimensions' into four parts: describing the object, storing the description, producing views that are recognisable as the object, and using the computer to generate geometric descriptions based on non-geometric descriptions. The main advantages of using three dimensional models, he describes as - permitting changes in the object description to be based on functional requirements, and permitting a wider range of graphic output forms to be automatically generated, than was possible from two dimensional representations of a body. The main disadvantage being that except for principal views:

> "there is no simple, direct correspondence between the object description and the display. Traditional languages, graphic or verbal, do not enable economic and precise description of three dimensional solids, and there is almost no precedent mathematics".

Appel describes two programs written in FORTRAN which attempt this task: SIGHT and LEGER. The SIGHT program was designed to draw free

standing polygons in space. A solid is described by a series of polygons completely enclosing a volume. The input (object) description is a list of cartesian coordinates that specify the vertex points of the polygons. By using this list in different ways the bounding polygons can be designated transparent, translucent, or opaque. The polygon boundaries are drawn out to create an impression of the boundary planes of material objects. To give a more realistic impression, the lines which would be hidden from the observer are eliminated or dashed where an intervening object is transparent.

The SIGHT system stored polygon descriptions as rings of vertex coordinates and the equations of the segment lines which connect the vertices. Simple to describe, SIGHT was somewhat cumbersome and time consuming to use (Appel, 1968). However, as an experimental program it provided a tool for investigating a range of graphic processes, such as shading line-drawings which, later on, were implemented in the more efficient but more complex system called LEGER.

The restrictions which SIGHT imposed on an object or scene description were the consequence of using straight line edges, plane facets and, the description of surface facets being limited to one external polygon, and one internal polygon (hole). There was no attempt to provide a linking structure for associating points with either surfaces, with lines, or with objects; or conversely for associating surfaces with lines or points. Such a structure would have made volume descriptions more versatile and would have reduced the calculation time for graphic output. A further restriction resulted from a fixed storage allocation for polygon boundaries: 20 lines for external boundaries and 10 for

internal boundaries. Appel comments that this arbitrary choice of storage space proved to be inadequate: a fixed limit of this sort was an unreasonable imposition, and an unrestricted number of line segments should have been the goal for the design of such a data structure.

In preparing a drawing from this description, all the lines to be drawn are divided into short segments, and the line of sight to a point on each segment is tested to determine if the line of sight pierces an opaque surface. Because this basic scheme is, in effect, a point by point process, calculation time increases with resolution or with the minimum size of line segment. The ideal scheme for generating graphic output would enable perfect resolution for any size of drawing to be achieved in the same time period.

The LEGER program, based on the experience gained from the SIGHT program, was built to provide improved facilities for input, storage, and graphic output, along with a procedure for automatic description generation and manipulation. These improvements meant the removal of restrictions on the number of vertices in polygon boundary lists, an improvement in the efficient use of storage, and a considerable reduction in the time taken to create line drawings.

There were two modes of object description provided in LEGER: the micro-description and the macro-description. A micro description of an object gave all the vertices and all the lines and joined them in the polygon boundaries of plane facets. The facets taken together defined the volume. The micro description of two lists of the form shown in Table 1, and Table 2.

DEFINING SIMPLE VOLUMES

Table 1 contains a series of tagged vertices and their spatial co-ordinates. The tags did not need to be in order but the 1000 end tag had to be present to end the file. The vertex list determined the size of the picture. If the coordinates were changed, only the size of the picture changed, not the basic geometrical organisation.

TABLE 1 Data Structure for Vertex Coordinates

Vertex Tag	Coordinates x	y	z
1	0.00	1.00	2.00
2	1.11	8.00	0.00
3	1.22	0.00	7.00
•	••••	••••	••••
•	••••	••••	••••
1000			

TABLE 2 Data Structure Used for Volumes

Picture Element	Picture Structure	Surface	Vertex Tag	Card Number
Surface A	1	1	3	1
	1	1	4	2
	1	1	5	3
	1	1	6	4
	1	1	Ø	5
End Surface	999			
Surface B	1	2	5	6
	•	•	•	•
End Surface	999			
Surface C	1	3	8	•
	•	•	•	•
End Surface	999			
End Object	888			
Surface F	2	7	9	•
	•	•	•	•
	•	•	•	•
End Object	888			
End Input	777			

DEFINING SIMPLE VOLUMES

The geometric organisation of the components of the object descriptions is provided in Table 2 which shows the arrangement of the vertices in lines, of polygon boundaries in plane surfaces and the surfaces which make up each object. The tags for the separate objects and the end marker for each list of linked boundary points - 999 - is given in column 1. The end of each collection of surfaces making up an object is indicated by the marker - 888, and the end of the collection of objects is indicated by the marker - 777. The second column gave the surface tags, the third column the vertex tags, and the final column the number of the input data card. The order of these cards was important because it defined the order in which the list of vertices occurred. In each facet is the boundary line. The zero vertex tag was used to close the boundary by linking back to the first vertex. In preparing the order of these vertices, a consistent sense of rotation round the surface boundaries had to be adopted and internal boundary points round holes had to be ordered in the opposite direction to that used for external boundaries.

The structure of the data in these two lists was used by the main program to generate a storage list in the machine which contained more explicit information about the objects used. This generated information included: the equation of planes, the equation of lines, the surfaces' orientation, internal and external identifications of corners, surface to line correspondence, object to surface correspondence and the general sense of rotation of surface boundaries. From this storage list the program then generated a string of linked 'element-description' blocks. This organisation provided a dense form of data packing, and allowed the 'construction' and 'decomposition' of the picture elements. The

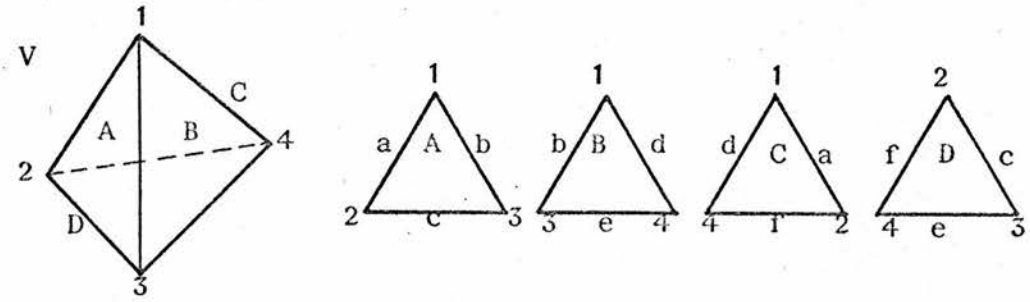
advantage of 'decomposition' was to be able to tell, for example, whether a point lay inside a particular surface. This required the surface to be decomposed into all the line segments in its boundary. The advantage of construction occurred when, for example, it was necessary to determine the solid side of a surface. This required all the faces of the objects making up its closed surface to be brought together.

The second level of object description provided for by LEGER was the macro-description. This consisted of subroutines which were called from the data input stream to copy or to manipulate previously stored micro-descriptions. These subroutines where they created new object descriptions from existing micro-descriptions operated on both the lists described above. Rotation and translation routines only need to operate on List 1 since these operations did not disturb the internal geometrical organisation of the object.

These data structures are complex and not easy to work with, reflecting Appel's comment that there is no direct correspondence between the object's description and the display, or in this case the mental image of the object being studied. The first experiments were an attempt to find a representation which would make it easier to work with this complex mixture of points, lines and facets. Given a faceted object it is possible to describe the object in terms of its collection of boundary faces. Given a plane boundary face it is possible to describe it by its boundary line, and given a boundary line it is possible to break this down into straight line segments each represented by its end coordinates. What was wanted was some way of unifying these

different elements into a single structure which was, as far as possible, self-explanatory.

The first structuring principles studied were provided by Graph Theory. If the vertices and edges which belong to a faceted object are considered, it is possible to create a matrix representation showing which vertices of the object's set of vertices are linked to form edges as shown in Figure 1.



Components of a Tetrahedron

Figure 1

The tetrahedron in Figure 1 is an example of a simple object which can be described in terms of vertices, straight line segments, and plane faces. The hierarchical list of these elements where each is given an identifying tag becomes:

Object	V
Faces	A, B, C, D.
Edges	a, b, c, d, e, f
Vertices	1, 2, 3, 4

The links between the vertices corresponding to edges can be expressed as a graph matrix as shown in Figure 2.

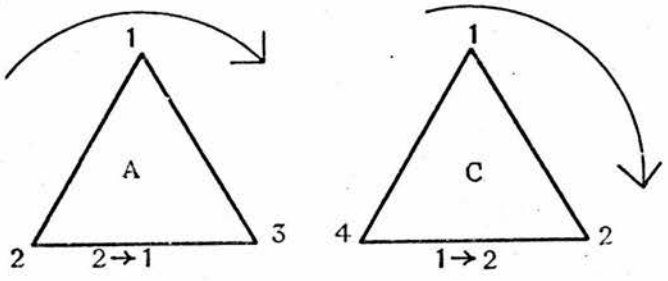
DEFINING SIMPLE VOLUMES

	1	2	3	4
1	0	1	1	1
2	1	0	1	1
3	1	1	0	1
4	1	1	1	0

Graph Matrix for a Tetrahedron

Figure 2

In the matrix in Figure 2 positions V(1,2) and V(2,1) are occupied. This expresses the fact that point 1 is linked to point 2, and also that point 2 is linked to point 1. This is consistent with the two directions of linkage which occur in the two boundaries of A, and C if they are expressed with a consistent sense of rotation about each face.



	1	2	3	4
1	0	0	1	0
2	1	0	0	0
3	0	1	0	0
4	0	0	0	0

	1	2	3	4
1	0	1	0	0
2	0	0	0	1
3	0	0	0	0
4	1	0	0	0

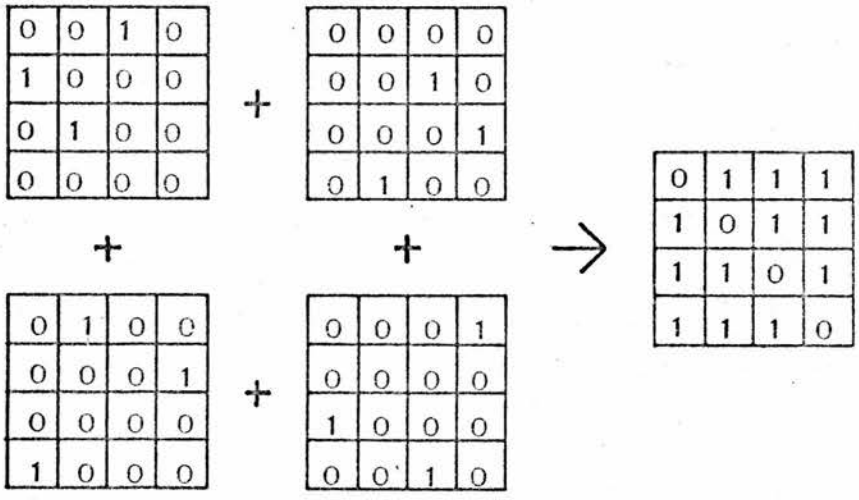
Rotation Convention in Graph Matrices

Figure 3

If the graph matrices for these two faces A and C are prepared

this directional convention for line 1/2 or 2/1 is illustrated in Figure

This matrix form can be manipulated using boolean operations. If the separate facet matrices are added together the result will be the original matrix representing the total volume (Figure 4). These matrices can also be multiplied together. If the vector representing a vertex is multiplied by the linkage matrix, the result is a vector representing the point to which the original point is linked. This operation is illustrated in Figure 5.



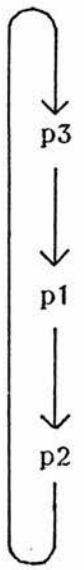
Matrix Addition

Figure 4

The process shown in Figure 5 produces by a series of multiplications a sequence of vertices in the order in which they would occur in a facet boundary list. The boundary of a facet is the simplest form of circuit. A circuit is any closed loop which can be formed by following the links

DEFINING SIMPLE VOLUMES

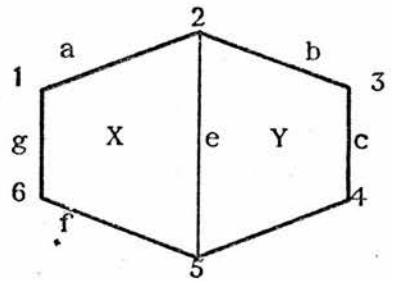
p1 p2 p3 p4		p1 p2 p3 p4		p1 p2 p3 p4
(0, 1, 0, 0)	*	[0, 1, 0, 0; 0, 0, 1, 0; 1, 0, 0, 0; 0, 0, 0, 0]	=	(0, 0, 1, 0)
(0, 0, 1, 1)	*	[0, 1, 0, 0; 0, 0, 1, 0; 1, 0, 0, 0; 0, 0, 0, 0]	=	(1, 0, 0, 0)
(1, 0, 0, 0)	*	[0, 1, 0, 0; 0, 0, 1, 0; 1, 0, 0, 0; 0, 0, 0, 0]	=	(0, 1, 0, 0)



Matrix Multiplication

Figure 5

in a linkage matrix, this definition also includes the boundary of adjacent facets.



		a	b	c	d	e	f	g
X:	C1	1	0	0	0	1	1	1
Y:	C2	0	1	1	1	1	0	0
	C1∘C2	1	1	1	1	0	1	1

Circuits

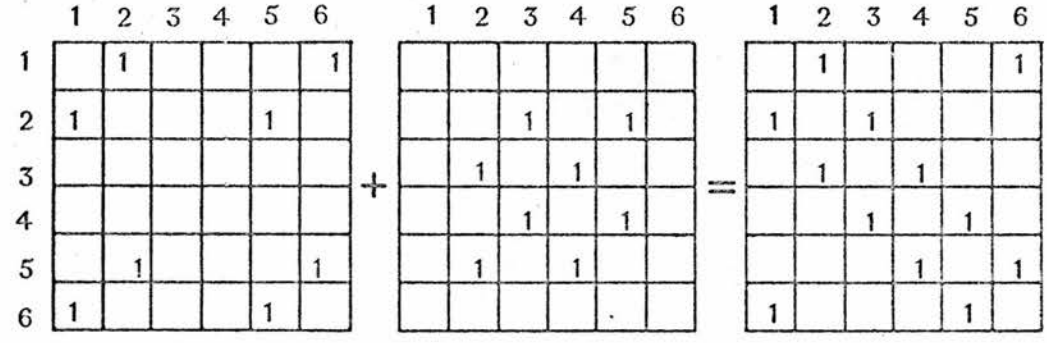
Figure 6

If the two circuits which represent the boundaries of areas X and Y are represented by the vectors C1 and C2 in Figure 6 then the circuit round X and Y taken as a single area can be obtained by the operations (0,1 = 1,

or 0,0 = 0, or 1,1 = 0) applied to each pair of corresponding edges in the two vectors C1 and C2 to give the composite vector C1∘C2.

This operation on circuits can also be expressed as a similar operation carried out on the linkage matrices as shown in Figure 7.

Though these graph matrix operations show one way of operating on a particular description of an object, they do not provide a way of uniting the collection of vertices edges, and facets in a single comprehensible form.

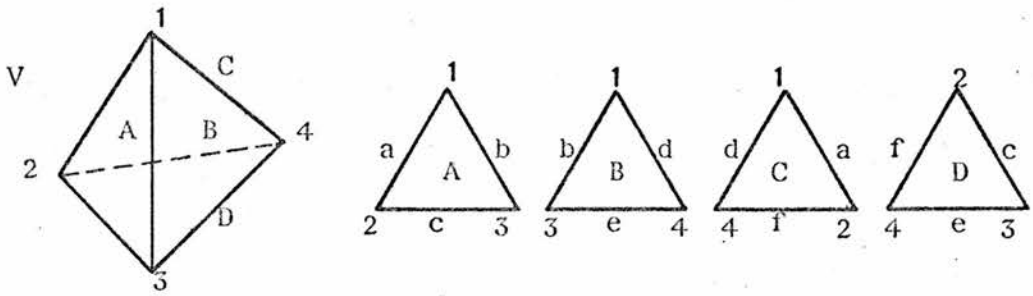


Merging Two Adjacent Zones

Figure 7

However, if the linkage matrix is taken and instead of using the boolean (0,1) to indicate whether a link exists or not, the actual edge tag is substituted for the 1, the result for the tetrahedron becomes that shown in Figure 8.

This includes edge tags and vertex tags. However, it was found that little use could be made of this representation. The symmetry about the main diagonal means that for storage purposes it could be reduced



A)

	1	2	3	4
1	.	a	b	d
2	a	.	c	f
3	b	c	.	e
4	d	f	e	.

B)

	2	3	4
1	a	b	d
2		c	f
3			e

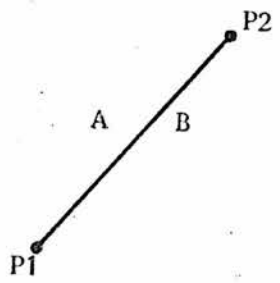
Using Edge Tags in a Linkage Matrix

Figure 8

to the triangular form (B), without losing information.

There is another way in which the edges can be named, which depend on the properties of planar graphs. Each of these volumes is in effect being described by a network of lines on its surface; for simple volumes this by definition makes this network of lines into a planar graph. Such a graph has a dual form. This shows up in there being two ways to name an edge or link in such a graph. So far the edges have been defined by their end points: $p_1 p_2$. In the dual graph the same line will be named by the two facets it divides from each other for example A/B:

Both these naming strategies indicate directional characteristics of an edge. A/B will indicate a link in the opposite direction to the

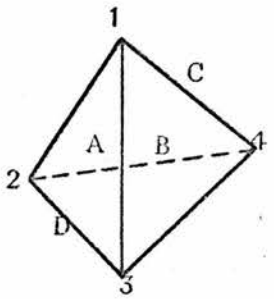


Dual Name for a Line Segment

Figure 9

name B/A. If both these naming strategies are used in the same matrix form, then for the tetrahedron the result can take one of the two forms shown in Figure 10.

It can be seen that the pattern of entries in these matrices correspond for each named vertex, with the pattern obtained in the boolean boundary matrixes. The previous addition of the boolean boundary matrices could be expressed in the way shown in Figure 11.



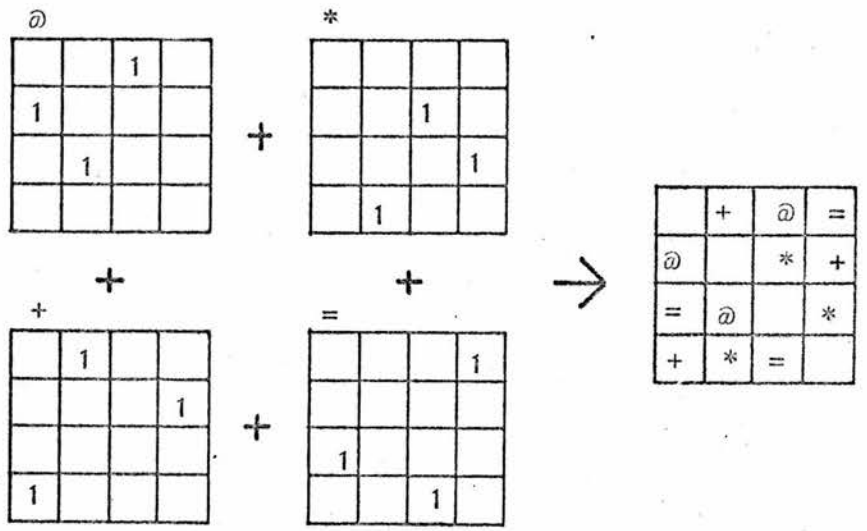
	1	2	3	4
1	.	A	B	C
2	C	.	A	D
4	A	D	.	B
4	B	C	D	.

	A	B	C	D
A	.	3	1	2
B	1	.	4	3
C	2	1	.	4
D	3	4	2	.

Volume Matrices: Point/Area & Area/Point

Figure 10.

DEFINING SIMPLE VOLUMES



Volume Matrices: Addition

Figure 11

This suggests that the same operations which were performed on these boundary matrices could also be performed on the new matrix form. If any row of the point/area matrix in Figure 11 is taken as a vector and multiplied by the original matrix with the rules (A.A = A, or A.B = Ø, or A.(A+B) = A.A + A.B = A) the results will be the transformations shown in Figure 12.

$$(O, A, B, C) * \begin{bmatrix} O, A, B, C \\ C, O, A, D \\ A, D, O, B \\ B, C, D, O \end{bmatrix} = (O, C, A, B)$$

$$(O, C, A, B) * \begin{bmatrix} O, A, . \\ . \quad \circ \quad . \end{bmatrix} = (C+A+B, O, O, O)$$

$$(C+A+B, O, O, O) * \begin{bmatrix} \quad \end{bmatrix} = (O, A, B, C)$$

Volume Matrices: Multiplication

Figure 12.

DEFINING SIMPLE VOLUMES

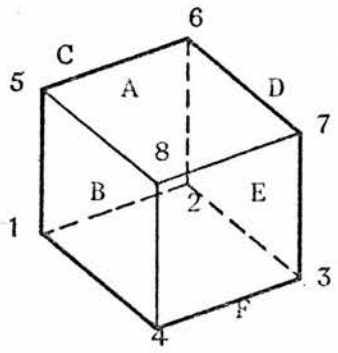
If each of these vectors is taken in the order in which it is produced, the order of the boundary points round the facets named in the original vector can be created as shown in Figure 13.

	P1	P2	P3	P4	A	B	C	D
1)	0	A	B	C	P2	P3	P4	0
					↓	↓	↓	
2)	0	C	A	B	P3	P4	P2	0
					↓	↓	↓	
3)	C+A+B	0	0	0	P1	P1	P1	0
					↓	↓	↓	
4)	0	A	B	C	P2	P3	P4	0

Circuits Produced by Matrix Multiplication

Figure 13

Since this example is based on the tetrahedron which is a simple and regular Figure, the same operation was carried out on two more Figures. Firstly a cube, and secondly a square based pyramid. In the example given above there are several regularities which may be merely a co-incidence, for example vector 2 corresponds with column 1 of the matrix.



	A	B	C	D	E	F
A	•	5	6	7	8	
B	8	•	5		4	1
C	5	1	•	6		2
D	6		2	•	7	3
E	7	8		3	•	4
F		4	1	2	3	•

Volume Matrix: The Cube

Figure 14.

A condensed presentation of the same operations carried out on the

point-area matrix for the cube: the dual to the matrix given above, takes the form shown in Figure 15.

$$\begin{bmatrix} 0, F, 0, B, C, 0, 0, 0 \\ C, 0, F, 0, 0, D, 0, 0 \\ 0, D, 0, F, 0, 0, E, 0 \\ F, 0, E, 0, 0, 0, 0, B \\ B, 0, 0, 0, 0, C, 0, A \\ 0, C, 0, 0, A, 0, D, 0 \\ 0, 0, D, 0, 0, A, 0, E \\ 0, 0, 0, E, B, 0, A, 0 \end{bmatrix}$$

$$(0, F, 0, B, C, 0, 0, 0) * [M] = (0, 0, F, 0, 0, C, 0, B)$$

$$(0, 0, F, 0, 0, C, 0, B) * [M] = (0, C, 0, F, B, 0, 0, 0)$$

$$(0, C, 0, F, B, 0, 0, 0) * [M] = (C+F+B, 0, 0, \ldots\ldots)$$

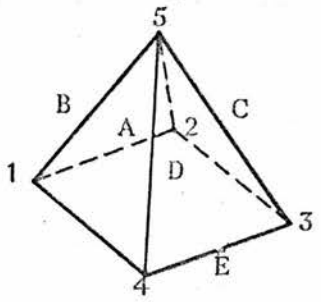
$$(C+F+B, 0, 0, \ldots\ldots) * [M] = (0, F, 0, B, C, 0, 0, 0)$$

Volume Matrix Multiplication for the Cube

Figure 15.

In this example it can be seen that the vector chosen (1) corresponds to row 1 of the volume matrix. It can also be seen that the result of the first multiply does not give a vector (2), which matches with column 1 of the volume matrix. In fact vector 2 does not match with any of the row or column vectors of the volume matrix. However, vector 3, the result of the next multiplication does match with column 1. The consequence of multiplying this vector with the volume matrix follows the same pattern shown in the case of the tetrahedron. It would appear that when this form of correspondence appears, there are only two further multiplications before the original vector is restored, in this case (1) or (b). However, the cube is again a

regular figure in that all its facets have the same number of sides, so for the next figure an example was taken which had a different number of sides to some of its faces:



$$\begin{bmatrix} O, & B, & O, & E, & A \\ E, & O, & C, & O, & B \\ O, & E, & O, & D, & C \\ A, & O, & E, & O, & D \\ B, & C, & D, & A, & O \end{bmatrix}$$

$$(E, O, C, O, B) * [M] = (B, O, O, E, C)$$

$$(B, O, O, E, C) * [M] = (O, B+C, E, O, O)$$

$$(O, B+C, E, O, O) * [M] = (O, E, C, O, B)$$

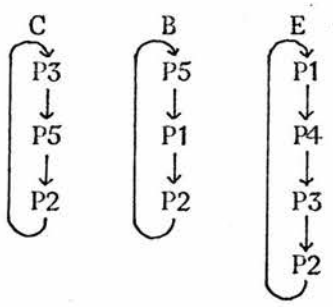
$$(O, E, C, O, B) * [M] = (E+B, O, O, O, C)$$

Matrix Multiplication for a Square Based Pyramid

Figure 16

If the points which have been passed through during this sequence of multiples are laid out in a table, relating them to the facets which they belong to, the result is shown in Figure 17.

	C	B	E
1)	[P3]	[P5]	[P1]
2)	P5	P1	P4
3)	P2	P2	P3
4)	[P3]	[P5]	P2
5)	P5	P1	[P1]

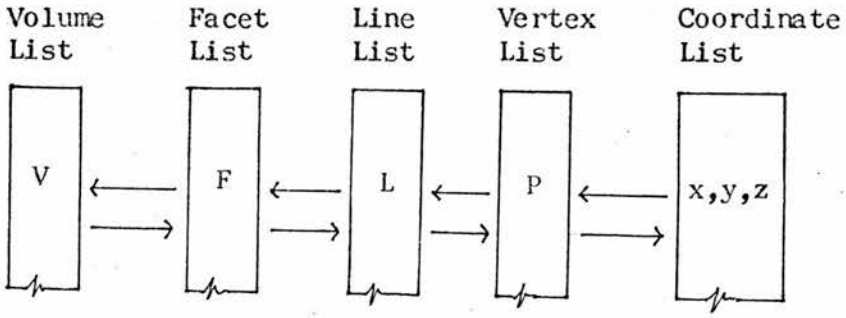


Circuits for a Square Based Pyramid

Figure 17.

It can be seen that the boundary of each of these facets has been closed: in (4) for C and B, and in (5) for E. However, unlike the previous example, vector (5) does not match vector (1). The reason is fairly clear when the nature of the volume is considered: facets C and B have three sides, whereas E has four sides, consequently it will take 13 multiples to transform vector (1) back into itself. For present purposes it is clear that to obtain the boundary links between points in their correct order should only require one more multiple than the number of links in the largest facet boundary. Loops with a smaller number of line segments can be closed when the facet name in the vector returns to the original position held in vector 1.

These volume matrices appear to be one step towards the objective of finding a representation which unites into a single structure the various elements which it may be necessary to name explicitly. If the general form of the data structure discussed by Appel is considered it can be represented by the diagram:



Volume Data Structure

Figure 18.

The argument for a separate vertex and coordinate list is very strong.

However, if the pointers between the separate lists indicated above are laid out for each individual element in any one of these lists the number of links could be very large:

TABLE 3 Pointers in the Volume Data Structure

Elements	Forward Pointers	Backward Pointers
POINTS	Contiguous Lines Contiguous Facets Contiguous Volumes	-
LINES	Contiguous Facets Contiguous Volumes	Constituent Points
FACETS	Contiguous Volumes	Constituent Points Constituent Lines
VOLUMES	-	Constituent Points Constituent Lines Constituent Facets

The aim in developing the matrix form was to attempt to find a way in which the natural order in which elements were stored, could be used to reduce the number of links which must be explicitly stated in the form of pointers. From the matrix it is possible to obtain the end points of a line segment since they are symmetrically placed about the main diagonal of the matrix. If one point is A(i,j) then the other end of the line is A(j,i). Given an area A all the adjacent facets will contain entries in the A row or column of the Area-point matrix. Similarly, the points which make up the boundary of the facet A will be the entries in these rows or columns. The operations discussed above will give these points in the order in which they occur in the boundary line of the area. They will also give the order in which facets occur round any vertex on the surface. The volume tags

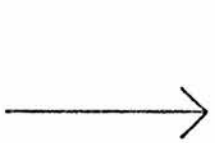
will correspond to the matrix names, the facet and point names will be the column and row identifiers for the matrix, and the pairs of diagonal cells will define the lines.

Having established what appeared to be a reasonable starting point, the next stage was to investigate the operations on volumes, using this representation. Initially, only simple volumes are considered to keep the problem from becoming too complicated. If the matrix form itself is considered, it is merely an indication of the links in a graph. The geometrical properties of the object will depend on the actual coordinates of the points, the form and shape of the lines and the surface facets, so here the use of straight lines and plane facets is not essential. However, it greatly simplifies the problem to assume this, and the study of curved forms are reserved for a later stage.

One of the properties of the matrix representation is the consistent way direction and rotation are treated. This makes it possible to distinguish the inside of a volume from the outside. If the space inside the volume is given the property 'solid' then the space outside is 'not-solid' or void. To change a matrix form from indicating solid inside to representing solid outside corresponding to the reverse direction Appel uses for indicating polygon boundaries of holes simply involves transposing the matrix. So that, for a tetrahedron:

•	*	x	+
+	•	*	o
*	o	•	x
x	+	o	•

SOLID: A



•	+	*	x
*	•	o	+
x	*	•	o
+	o	x	•

VOID: A

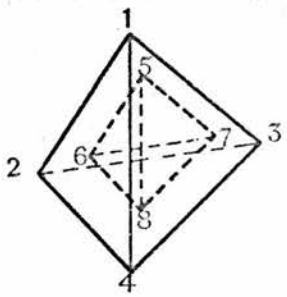
Figure 19

Volume Matrices: Solids and Voids

DEFINING SIMPLE VOLUMES

A void surrounded by solid tends to suggest a second boundary surface, unless the whole universe is regarded as solid. To represent a hollow box there are two possible forms. Either the two matrices for the independent surfaces are left as separate matrices, or they can be merged into a single matrix. This operation seems simple minded but it will be seen that the opposite operations; the decomposition of a large matrix into simpler component matrices may well be a valuable space saving operation.

Given the appropriate geometrical relationships it is possible to merge two or more matrices to give a new matrix describing a new volume. The first example of this process to be considered, was the



A	1	4	3
3	B	1	2
1	2	C	4
4	3	2	D

+

E	7	5	8
5	F	6	7
8	5	G	6
7	6	8	H

→

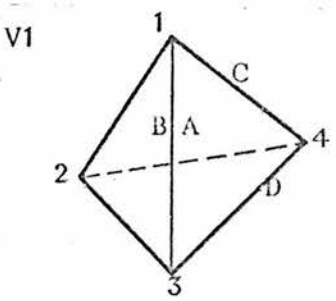
A	1	4	3				
3	B	1	2				
1	2	C	4				
4	3	2	D				
				E	7	5	8
				5	F	6	7
				8	5	G	6
				7	6	8	H

Merging Two Matrices

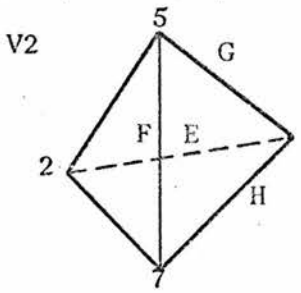
Figure 20.

addition of two tetrahedra. Take the two separate volumes V1 and V2:

DEFINING SIMPLE VOLUMES



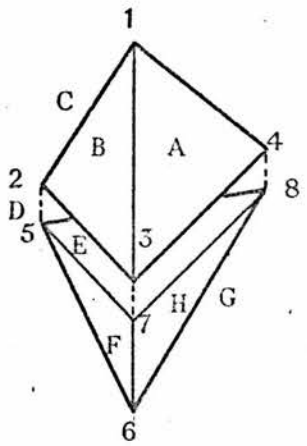
A	1	4	3
3	B	1	2
1	2	C	4
4	3	2	D



E	5	8	7
7	F	5	6
5	6	G	8
8	7	6	H

Merging Two Volumes V1 and V2

Figure 21



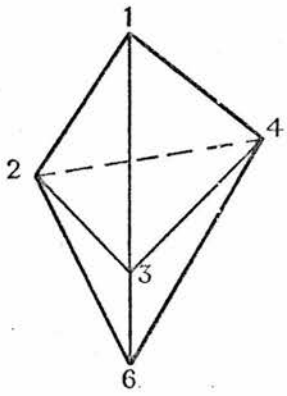
A	1	4	3				
3	B	1	2				
1	2	C	4				
4	3	2	d				
				E	5	8	7
				7	F	5	6
				5	6	G	8
				8	7	6	H

Stage I: Merge the Two Matrices

Figure 22

If the faces E and D are mirror images so that when they are brought together in the correct orientation point 3 matches with point 7, point 2 with point 5 and point 4 with point 8, it is possible to merge

these two volumes so that faces E and D become totally contained and can consequently be removed from the matrix description.



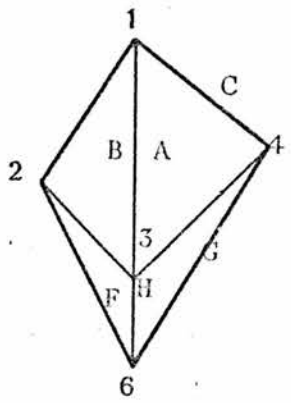
A	1	4	3					3
3	B	1	2			2		
1	2	C	4				4	
4	3	2	D					
				E	2	4	3	
	3			3	F	2	6	
		2		2	6	G	4	
4				4	3	6	H	

Stage II: Remove Internal Faces from Matrix

Figure 23

In merging the two tetrahedra, points 5, 8, 7 no longer exist, becoming 2, 3, 4. In the same process rows and columns representing facets E and D also vanish. However, if they were merely erased then points 2, 3, 4 would only occur twice each, in the resulting matrix, but in the accompanying volume diagram it can be seen that these points are each linked to four other points. In fact these displaced points are relocated in the new matrix by a very simple procedure: consider point 4: it is located in V(A,D) and V(D,C) in the first matrix and in V(D,H) and V(G,E) in the second matrix; these can be merged using matching indices to give V(A,D)+V(E,H)$\longrightarrow$V(A,H) and V(G,E)+V(D,C)$\longrightarrow$V(G,C). The resulting matrix becomes:

DEFINING SIMPLE VOLUMES



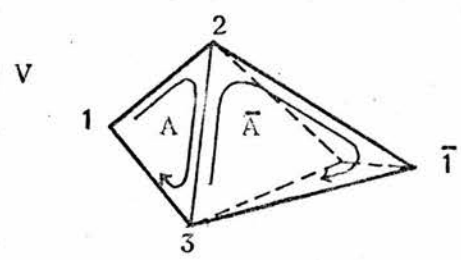
A	1	4			3
3	B	1	2		
1	2	C		4	
	3		F	2	6
		2	6	G	4
4			3	6	H

New Matrix Description for Volume V1+V2

Figure 24.

It is clear that finding two tetrahedra which can be matched in this way is very unlikely. However, by operating in a way that uses tetrahedra as building blocks it is possible to define the second volume as a reflection of the original volume in its base plane or merging plane.

The creation of a reflection volume requires two operations; firstly, the transformation of the point coordinates which may or may not require new tags, and secondly, transposing the original matrix, and the adoption of new facet names. The reason for transposing the matrix can be seen in the diagram:



V_A: 1/2 $\longrightarrow$ $V_{\bar{A}}$: 2/1

Volumes of Reflection

Figure 25

The direction taken round the boundary of corresponding facets is opposite in terms of corresponding points. This building process becomes:

a) Transpose the matrix

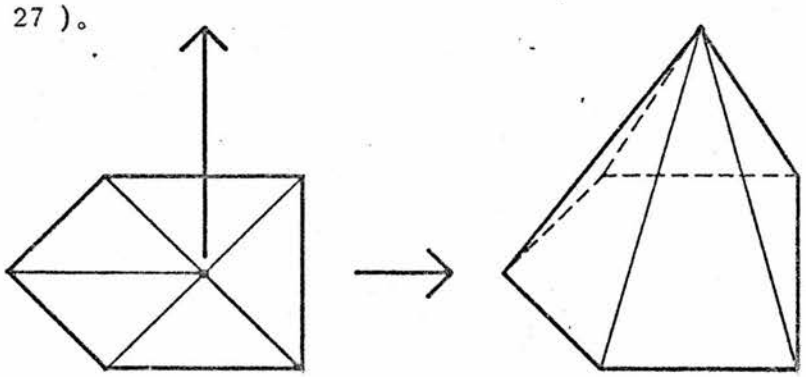
b) transform the coordinates of the points giving new tags where necessary

c) rename the newly formed facets

d) merge the two matrices as described above

If this process is used to construct a more complex object from tetrahedra the resulting surface of this object will naturally be triangulated. In many cases this would appear to be an advantage, in particular where plane facets are being assumed and it is necessary to relocate one point. For a fully triangulated surface the consequences of changing the relevant coordinate are simple.

The volume matrix need not be modified, merely the coordinate in the coordinate list. Where one of the facets that contains the moved point in its boundary, has more than three sides the situation is more complex. Either a new edge has to be added, which means operating on the volume matrix or a curved or bent surface has to be accepted. (Figure 27).

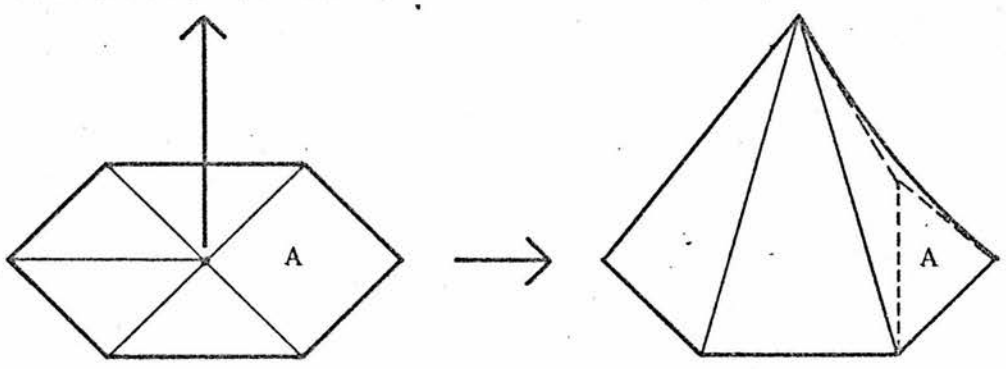


Moving a Vertex Coordinate on a Triangulated Surface

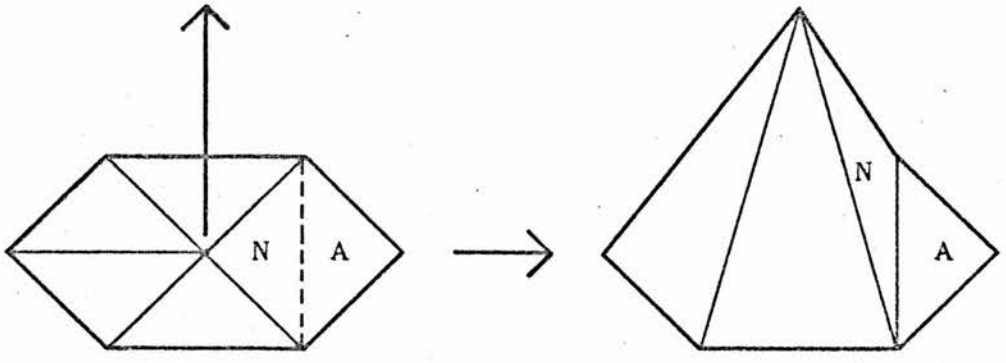
Figure 26

DEFINING SIMPLE VOLUMES

If the second strategy is adopted in order to retain plane facets the operation on the volume matrix requires the extra edge N/A to be created and the new face N.



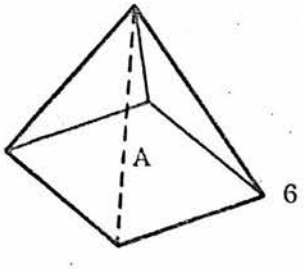
Allow the Surface to Curve



Triangulate the Surface

Moving a Vertex on a Non-triangulated Surface

Figure 27.



A	1	4	3	6
3	B	1	2	
1	2	C		4
6	3		F	2
4		2	6	G

Square Based Pyramid

Figure 28.

DEFINING SIMPLE VOLUMES

Taking a square based pyramid as the starting volume this process can be demonstrated in the following way. If point 6 is to be moved, then a new facet H must be created, as well as a new edge A/H. The point 6 will be in the boundary of H but no longer in the boundary of A. Consequently, locate the occurrences of 6 in the rows and columns of A and relocate them in H in the following way:

$$\underline{6}: \quad V(A,G) \longrightarrow V(H,G)$$

$$\underline{6}: \quad V(F,A) \longrightarrow V(F,H)$$

The other points which need to be included in the boundary of H will be the points at the other end of the lines ending in 6 which occur in A:

$$V(G,A) : 4 \longrightarrow V(G,H)$$

$$V(A,F) : 3 \longrightarrow V(H,F)$$

However, these two points will still occur in A as the line A/H.

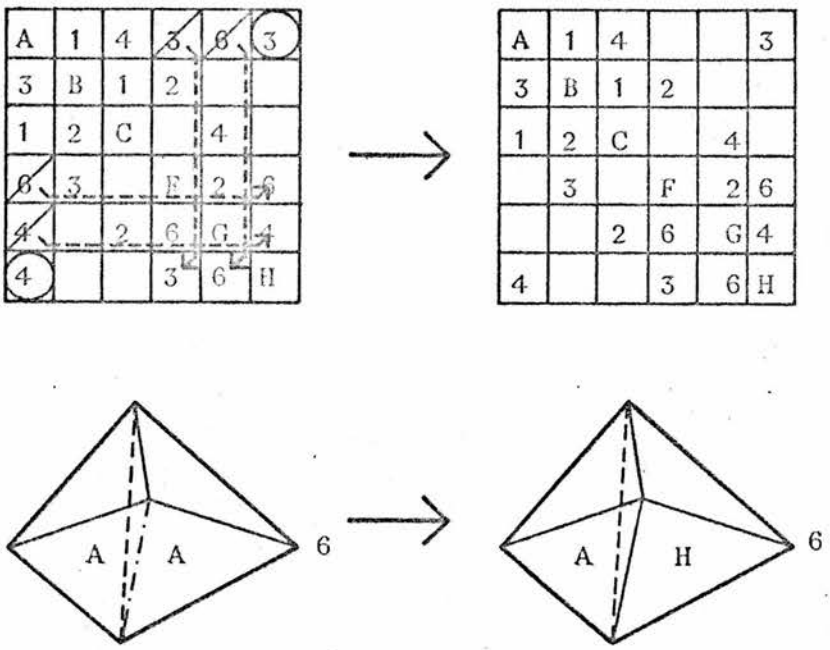
$$V(G,A) : 4 \longrightarrow V(H,A)$$

$$V(A,F) : 3 \longrightarrow V(A,H)$$

Diagrammatically these operations are shown in Figure 29.

The opposite procedure can be carried out in a corresponding manner. Given the final figure from the previous example - two merged tetrahedra. Merging faces A and H means that line A/H will be removed and consequently the points 3 and 4 in V(A,H) and V(H,A). It also means that the face H must be removed. As before, this means relocating the points in the row and column of H after A/H has been discarded. This is done using the indexes of the location of these points in the

DEFINING SIMPLE VOLUMES



Creating a New Facet

Figure 29

matrix:

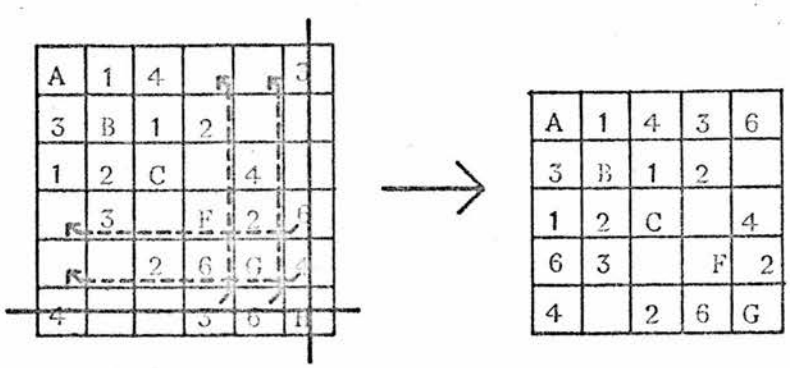
6: V(F,H) ⟶ V(F,A)

4: V(G,H) ⟶ V(G,A)

3: V(H,G) ⟶ V(A,G)

6: V(H,F) ⟶ V(A,F)

Diagrammatically:

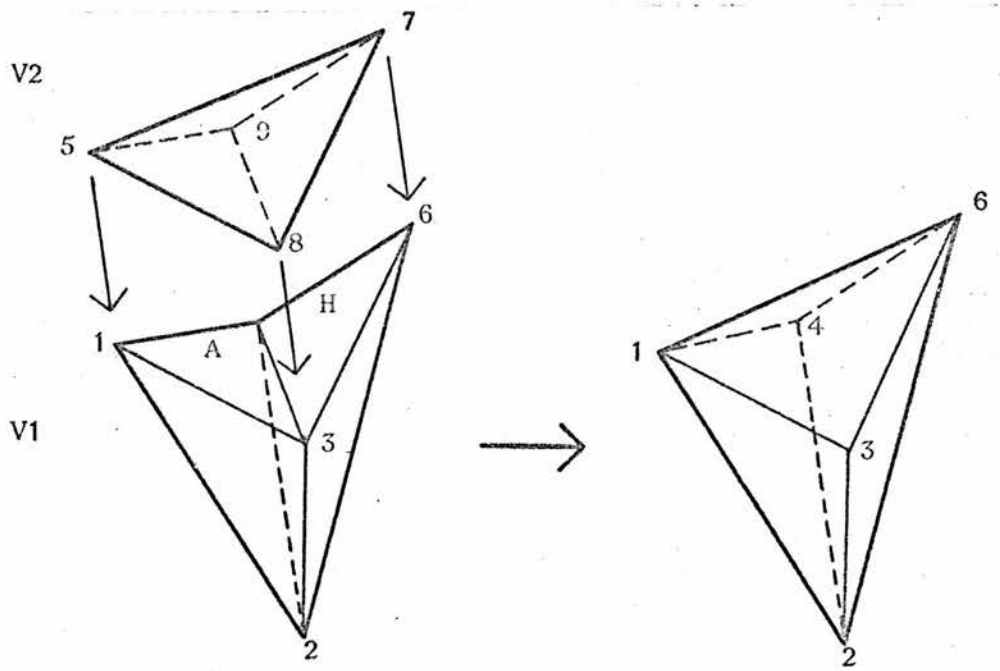


Merging Two Facets

Figure 30

DEFINING SIMPLE VOLUMES

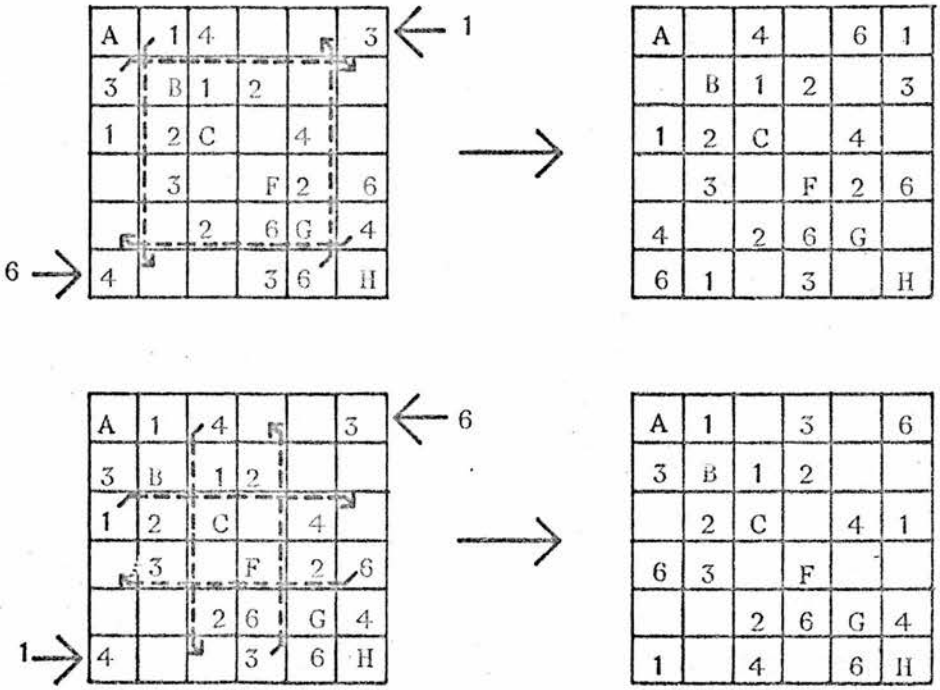
Continuing with the theme of volume construction using tetrahedra as building blocks, the second way in which an appropriate tetrahedron may be merged with an existing volume occurs in the following situation:



Merging Volumes on Two Faces

Figure 31.

As in the previous case, it will be possible to create V2 before merging it with V1. However, there seems to be a more direct approach. If the line 1/6 is included in the volume matrix for V1 instead of the existing line A/H in this case 3/4, then with the correct adjustments within the matrix, faces A and H will cease to be the internal faces lost by the merge but will become the new faces on either side of 1/6. Whether A/H is 1/6 or 6/1 is an arbitrary initial choice. However, once it has been made the rest of the procedure can be carried out automatically and can be expressed diagrammatically by the matrix operations shown in Figure 32.



Matrix Operation: Merging Two Volumes

Figure 32

Expressing the first of these diagrams as a sequence of operations on the contents of matrix cells, gives the result shown in Table 4.

TABLE 4 Index Operations for Merging Two Volumes

Operation Order	Original Contents	New Contents	Secondary Operation
1	V(A,H):3	⟶ 1	
5	V(A,B):1	⟶ ∅	V(H,B) ⟶ 1
	V(A,C):4		
6	V(B,A):3	⟶ ∅	V(B,H) ⟶ 3
	V(C,A):1		
2	V(H,A):4	⟶ 6	
	V(H,F):3		
3	V(H,G):6	⟶ ∅	V(A,G) ⟶ 6
4	V(G,H):4	⟶ ∅	V(G,A) ⟶ 4
	V(F,H):6		

DEFINING SIMPLE VOLUMES

In this sequence the only operation which has not been included, because it does not depend on the value of indexes, occurs between 2 and 3. To locate cells V(A,B) and V(H,G), it is necessary to find which cells in the rows and columns of H and A contain the points 1 or 6.

The inverse of this operation turns out to be the identical process. In this case it would merely be applied to the substitution of the line 1/6 by the line 3/4. However, the location of the points 1,3,4,6 in space would make the volume being added a void, so that in effect this is equivalent to volume subtraction.

Having considered several simple operations on small well behaved volumes, the next question was to consider what the limitations on this form of representation might be. The general properties which had emerged from the operations investigated so far can be summarised in the list:

a) Since a point can only act as one subcomponent of a facet its tag will only occur once in any row or column.
b) In a matrix V, V(i,j) and V(j,i) are the end points of a straight line segment, consequently the order of points in a facet row vector will not be the same as its column vector.
c) Both row and column vectors of a facet will contain the same set of point tags.
d) For straight line segments a point tag will appear at least three times in the whole matrix.
e) The occurrence of a particular point tag can be ordered using its matrix position indexes in the following way:

V(n,m), V(m,1), V(1,k) .....V(b,n)

If the row and column vectors are ordered in the appropriate way this shows up in the patterns:

n	*	
	m	*
*		1

n	*		
	m	*	
		1	*
*			k

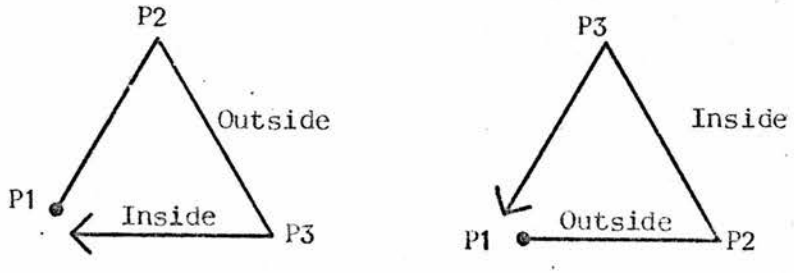
Pattern of Entries for a Vertex

Figure 33

f) The multiplication of a row vector by the matrix will give the next set of points in the respective facet boundaries.

g) The order of rotation about a face is reversed by transposing the matrix, so that where inside/outside is established for a particular matrix, transposing the matrix converts this to outside/inside.

h) The real rotation defined by this order in the matrix is established by the location of the named points in space. For example the same order p_1, p_2, p_3 can represent both outside/inside and inside/outside, as the two triangles demonstrate:



Order of Points and Geometrical Position

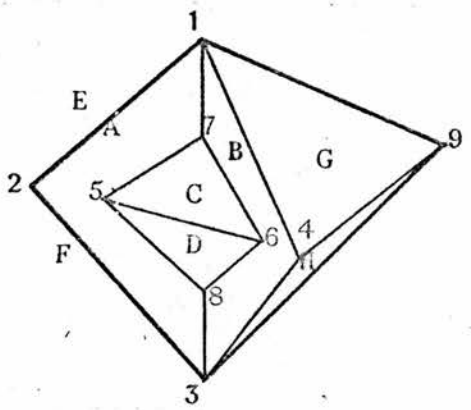
Figure 34

i) The rotation and translation of an object requires its points to be transformed, but the matrix is left unaffected.

j) Single reflections of an object require both that its points be transformed, and that the matrix be transposed.

The first difficulties with this structure appeared when it became clear that the creation of more complex object descriptions could lead to a breakdown in the simple relationship, necessary to use the matrix form: that the two point or two facet composite names for a line segment would only define one line segment.

It had been realised that the use of curved lines approximated by a series of straight line segments would give any number of points; however, such a line would only have two end points. It had also been realised that a closed loop starting and ending with the same point would create difficulties. However it was assumed that the definition of volumes constructed from plane faces would not lead to either of these conditions. It was the construction of the object:

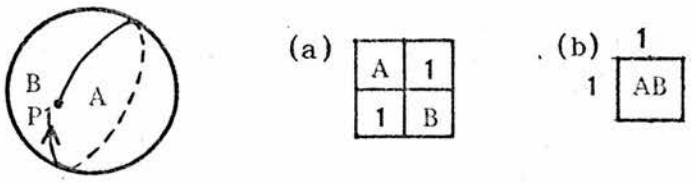


Volume Matrix Line Naming Problem

Figure 35.

which first demonstrated the problem. The line A/B was composed of two separate segments 1/7 and 8/3. What is more, if A and B were planes, these would be segments from the same straight line.

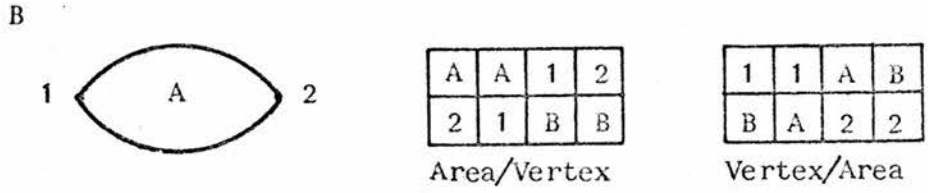
This led to a re-examination of the problem of loops and line arcs to see whether there was not some consistent way of expressing them in this matrix form. If a loop is considered on the surface of a sphere with one starting point the resulting matrix could take the form shown in Figure 36.



Volume Matrix for a Single Edge

Figure 36.

When two lines on a surface, linking two points are considered the result could be expressed in the form shown in Figure 37.



Volume Matrix for Two Edges

Figure 37.

If the same approach used in Figure 37 is applied to the naming problem which occurs in forming a volume matrix for the volume shown in Figure 35 then two columns for A and B would be necessary. If the two columns

A	A	1	8	7	5	2	3		
7	3	B	B	6	8			1	4
5		7		C	6				
8		6		5	D				
1						E	2	9	
2						9	F		3
		4				1		G	9
		3					9	4	H

Volume Matrix Line Naming Problem

Figure 38

for A and B are regarded as anything more than a convenient way of putting two values into one cell, then the matrix ceases to be square. However, if the entries in B/A and A/B are regarded as 7+3 and 1+8 in the sense used in the matrix multiplication already described, then the same multiplication process can be applied to this matrix to order the points round a boundary, and to order the facets round a vertex as shown in Figure 39.

	A	B	C	D	E	F	G	H
1.	(0,	1,	0,	0,	0,	0,	0,	0)
2.	(0,	0,	0,	0,	0,	0,	1,	0)
3.	(0,	0,	0,	0,	1,	0,	0,	0)
4.	(1,	0,	0,	0,	0,	0,	0,	0)
5.	(0,	1,	0,	0,	0,	0,	0,	0)

	A	B	C	D	E	F	G	H	
	0	1+8	7	5	2	3	0	0	B
	7+3	0	6	8	0	0	1	4	↓
	5	7	0	6	0	0	0	0	G
	8	6	5	0	0	0	0	0	↓
	1	0	0	0	0	2	9	0	E
	2	0	0	0	9	0	0	3	↓
	0	4	0	0	1	0	0	9	A
	0	3	0	0	0	9	4	0	↓
									B

Circuit Round Vertex 1

DEFINING SIMPLE VOLUMES

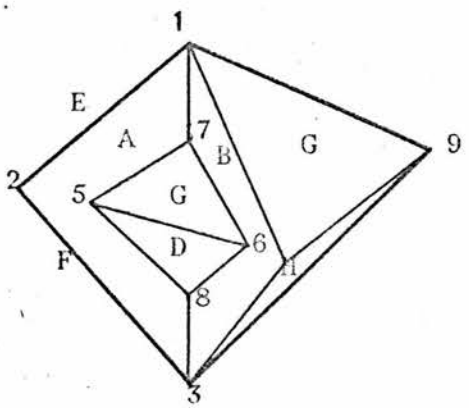
	1	2	3	4	5	6	7	8	9
1.	A	0	0	0	0	0	0	0	0
2.	0	0	0	0	0	0	A	0	0
3.	0	0	0	0	A	0	0	0	0
4.	0	0	0	0	0	0	0	A	0
5.	0	0	A	0	0	0	0	0	0
6.	0	A	0	0	0	0	0	0	0
7.	A	0	0	0	0	0	0	0	0

1	2	3	4	5	6	7	8	9	
0	E	0	B	0	0	A	0	G	1
A	0	F	0	0	0	0	0	E	↓
0	A	0	H	0	0	0	B	F	7
G	0	B	0	0	0	0	0	H	↓
0	0	0	0	0	D	C	A	0	5
0	0	0	0	C	0	B	D	0	↓
B	0	0	0	A	C	0	0	0	8
0	0	A	0	D	B	0	0	0	↓
E	F	H	G	0	0	0	0	0	3
									↓
									2
									↓
									1

Circuit Round Facet A

Figure 39.

Which suggests that this form can still be used, if the storage of more than one value in a cell can be managed in a convenient way. There seem to be several approaches which can be taken to this problem. One of these has already been mentioned, and consists of not creating problem volumes. If the surface of the object is fully triangulated then it is impossible for this difficulty to arise. The price which has to be paid for taking this approach is the extra space required to store all the extra facets. An alternative way is to triangulate merely that section of the surface necessary to resolve the problem:



A	8	7	5	2	3			1
3	B	8	8			1	4	6
5	7	C	6					7
8	6	5	D					
1				E	2	9		
2				9	F		3	
	4			1		G	9	
	3				9	4	H	
7	1	6						K

Naming Problem: Triangulation

Figure 40

DEFINING SIMPLE VOLUMES

This can be done by the following sequence of operations:

a) When a duplicate entry is found store it, in this case A/B:1/7.

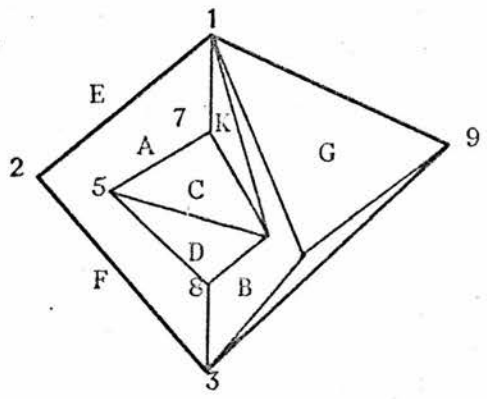
b) Complete the remaining entries.

c) Create a new row and column for K and enter A/K with 1/7.

d) Locate 7 in B, (or 1 in B) : V(C,B):7

e) Perform the Index operations:

$V(C,B):7 \longrightarrow \emptyset$ $\quad V(C,K) \longrightarrow 7$ $\quad V(K,C) \longrightarrow 1$
$V(B,C):6 \longrightarrow \emptyset$ $\quad V(B,K) \longrightarrow 6$ $\quad V(K,C) \longrightarrow 6$



A	8	7	5	2	3			1
3	B		8			1	4	6
5		C	6					7
8	6	5	D					
1				E	2	9		
2				9	F		3	
	4			1		G	9	
	3				9	4	H	
7	1	6						K

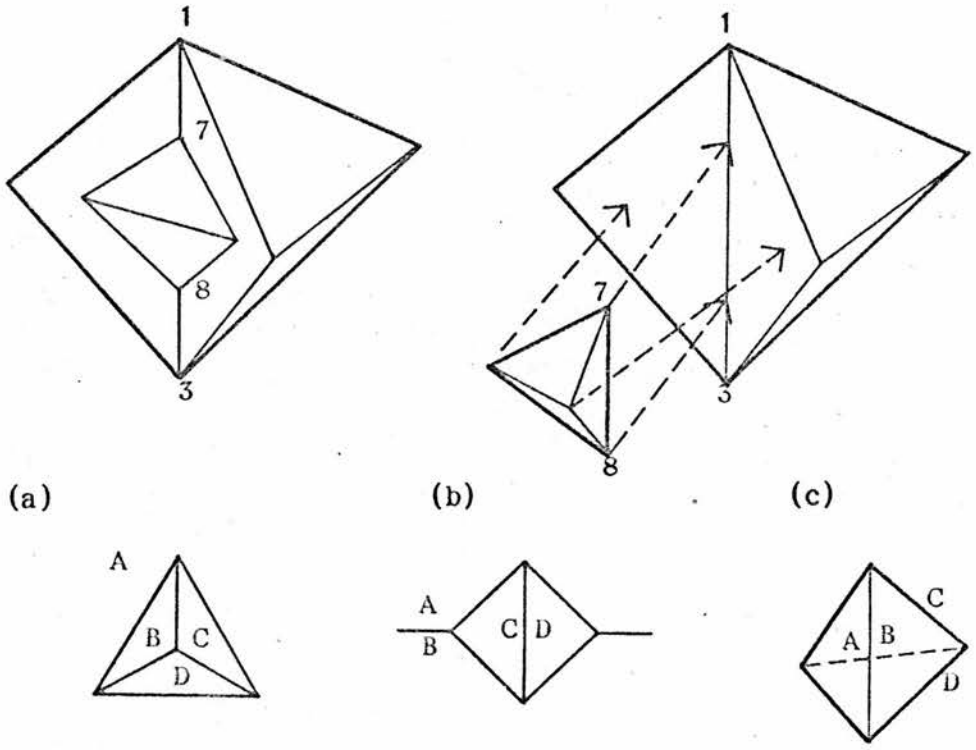
Volume Matrix: Triangulation

Figure 41

When drawing the graph of the edges of a tetrahedron each of the diagrams a, b, c are equally valid forms. In a the base plane is treated as though it were a large closed surface, a section of which can be shown in the diagram outside B, C, and D. Similarly, in b two facets B and A become part of this surface. While c is the conventional projected drawing showing the true geometry of the tetrahedron. It is only when a 'semi-island' of the form b occurs, however complex its internal facets might

be, that the line naming problem arises. The solution is to isolate the section of the edge graph which is linked by only two lines to the rest of the graph.

A third approach was based on the following observations:

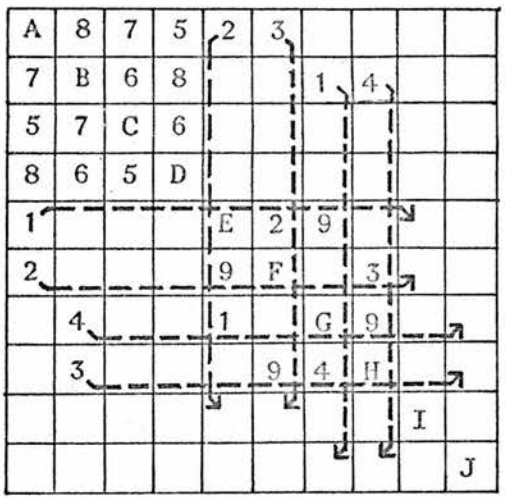


The Naming Problem and Islands

Figure 42.

This can be done very simply when a conflict of the form: A/B:1/7 and A/B:8/3 occurs, by swapping the appropriate points round to give A/B:1/3 and A/B:8/7. Two new faces need to be added to the matrix to represent the back of the newly created tetrahedron: I_a, J_b. If A/B is taken as 8/7 and I_a/J_b as 1/3, then the following operation can be performed:

DEFINING SIMPLE VOLUMES



Volume Matrix: The Naming Problem and Islands

Figure 43

a) Locate points 1 and 3 in rows or columns of A and B**

b) By using indexes carry out the operations:

**	V(A,F): 3 ⟶ Ø	V(I,F) ⟶ 3
	V(F,A): 2 ⟶ Ø	V(F,I) ⟶ 2
**	V(E,A): 1 ⟶ Ø	V(E,I) ⟶ 1
	V(A,E): 2 ⟶ Ø	V(I,E) ⟶ 2
**	V(B,G): 1 ⟶ Ø	V(J,G) ⟶ 1
	V(G,B): 4 ⟶ Ø	V(G,J) ⟶ 4
**	V(H,B): 3 ⟶ Ø	V(H,J) ⟶ 3
	V(B,H): 4 ⟶ Ø	V(J,H) ⟶ 4

The operations marked by a double asterisk require a searching procedure to find the cell with the given value. Once this has been done, the indexes of this cell are used to carry out the unmarked operations.

A	8	7	5						
7	B	6	8						
5	7	C	6						
8	6	5	D						
				E	2	9		1	
				9	F		3	2	
				1		G	9		4
					9	4	H		3
				2	3			I	1
						1	4	3	J

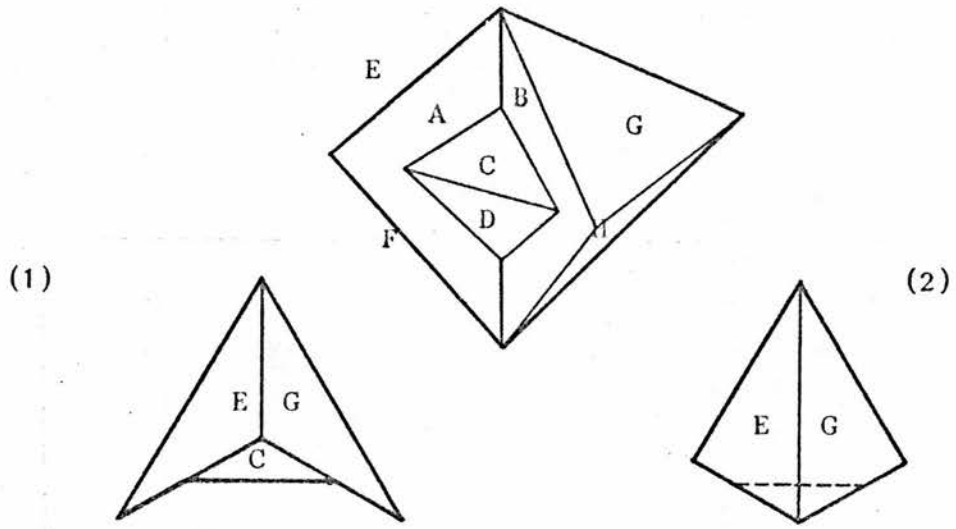
Volume Matrix Partition: Naming Problem

Figure 44.

The solution to the naming problem illustrated in Figure 43 results in a volume with two separate surfaces, which is indicated by the way the matrix can be partitioned. Further investigation of this solution is presented in the next chapter which discusses volumes which require multiple surface descriptions.

VOLUMES WITH MULTIPLE SURFACES

In this Chapter the matrix structure which was examined in the previous Chapter as a way of representing simple volumes is applied to more complex examples. The starting point for this investigation was the 'naming problem', and the way it could be removed by partitioning the original matrix. Though this process was attractive at first sight, there were a variety of difficulties which emerged which depended on the geometry of the surfaces of the volume. Even though plane faces were assumed to simplify the overall problem, it was found that the partitioned matrix could be defining one of two physical objects. These are shown in Figure 1, the two views from above in diagram 1 and 2 illustrating the differences.

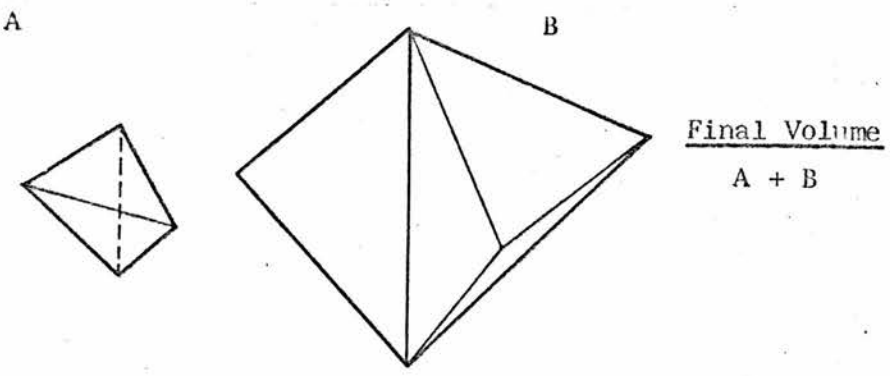


Two Interpretations of a Partitioned Matrix

Figure 1

Without including the coordinates of vertices in the volume description it is not possible to decide whether the second volume is, in effect, being subtracted from the first volume or being added to it. The partitioned matrix for diagram 1 required the two solid volumes in Figure 2.

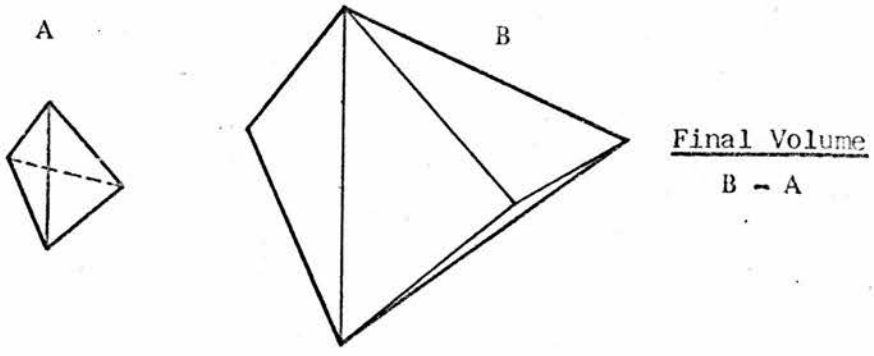
VOLUMES WITH MULTIPLE SURFACES



First Interpretation

Figure 2

The corresponding matrices for diagram 2 would represent the solid and void shown in Figure 3.



Second Interpretation

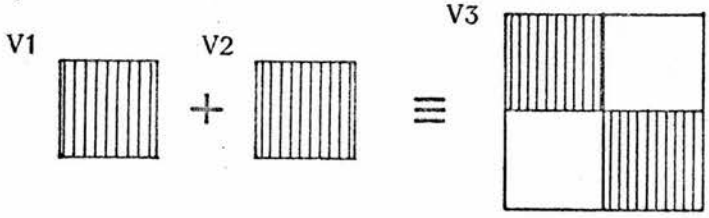
Figure 3.

Whether the change which partitioning the matrix produced in the volume description was important, seemed to depend on the way in which the data structure was to be used. If the volume content of the object was wanted, then it did not matter which form of description was used. On the other hand, it was not clear how to treat the second interpretation

in Figure 3 when preparing output drawings. Hidden line removal would require a specially designed procedure to remove front facing surfaces from volumes which were voids. This problem is discussed in greater detail in the Chapter on Graphic Models because the idea of volume subtraction was too important to discard. It was necessary to have voids to conveniently define a hollow box.

If the operations discussed in the previous Chapter are quickly reviewed, it is possible to summarise them in the following way.

When describing a hollow box, it was stated that its two surfaces could be expressed in one of the two forms shown in Figure 4.



Matrix Concatenation

Figure 4

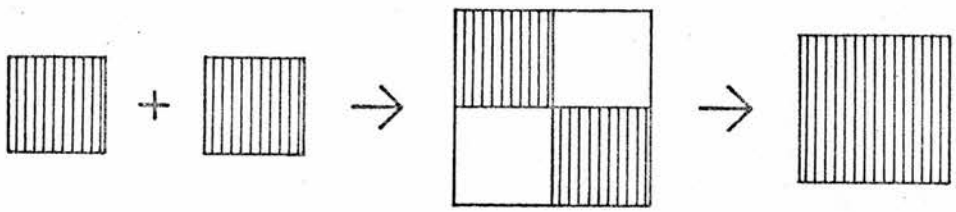
This is the simplest example of a hole. An island can be expressed in the same way, except that the two surfaces are not nested. Simple holes and islands are illustrated by the two dimensional diagrams in Figure 5

Holes and Islands

Figure 5

VOLUMES WITH MULTIPLE SURFACES

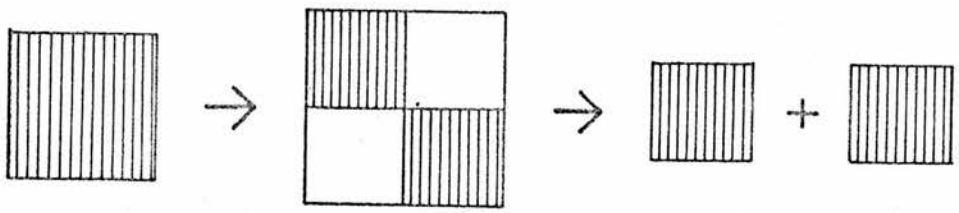
The composite matrix corresponding to either of these two situations can be taken as the definition of a single object. In the previous Chapter it was on a matrix of the form V3 in Figure 4 that simplifying operations were carried out to create a new matrix of simpler or more appropriate form. The whole sequence of these constructional operations are summarised in Figure 6.



Matrix Reducation

Figure 6.

The creation of the most appropriate form for the matrix representation of a volume was found to create the line-naming problem, and this could be solved by decomposing such a matrix by the reverse sequence shown in Figure 7.



Matrix Partition

Figure 7.

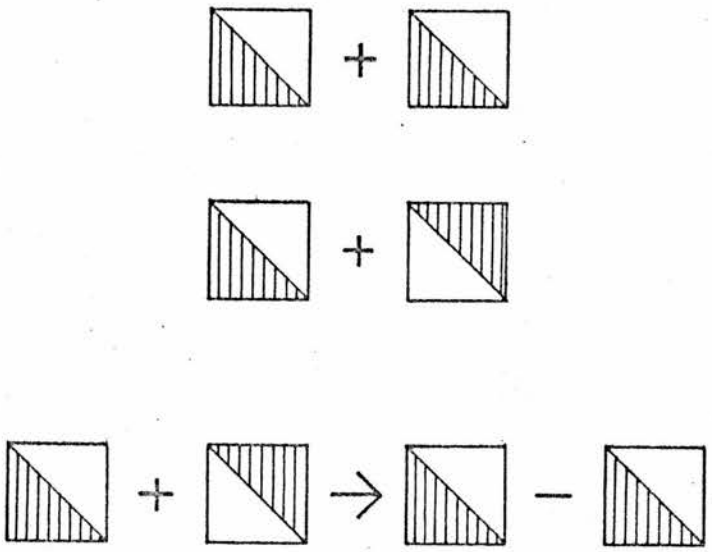
VOLUMES WITH MULTIPLE SURFACES

Returning to the volume shown in Figure 1 , it is attributing 'substance' to these volume descriptions which provides the distinction between the forms 'island' and 'hole'. If a particular matrix is called solid, in other words the directional property of 'inside' shown by these matrices has been associated with the property solid, then the transpose matrix, where this property is reversed, represents a void.

If the operations between matrices representing solids and voids are illustrated in the way shown in Figure 8 , where the solids have the lower triangle shaded, and voids the upper triangle shaded, then an equivalence between the addition of a solid and the subtraction of a void can be made.

ISLAND

HOLE



The Addition and Subtraction of Matrices

Figure 8.

The interpretation of this addition and subtraction if it is extended to include the addition of two solids which are defined by nested surfaces, must be made in terms of boolean algebra, since the only

reasonable result is a new solid.

For two nested surfaces: where surface 1 is outside surface 2

SOLID 1 + SOLID 2 ⟶ SOLID 1

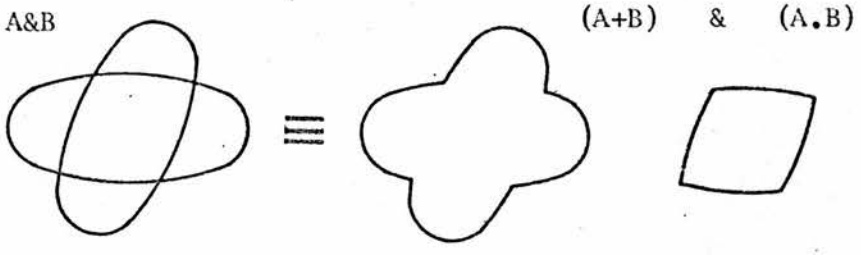
SOLID 1 + VOID 2 ⟶ SOLID 1 - SOLID 2

and for two independent surfaces:

SOLID 1 + SOLID 2 ⟶ SOLID 1 + SOLID 2

SOLID 1 + VOID 2 ⟶ SOLID 1

The implicit assumption which has been made so far in these operations is that the two surfaces do not cut each other in space. The problem which occurs when they do can be seen in the diagram in Figure 9.



Double Counting

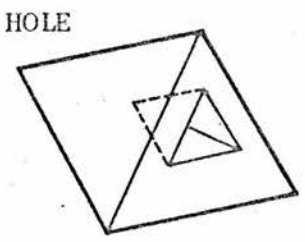
Figure 9.

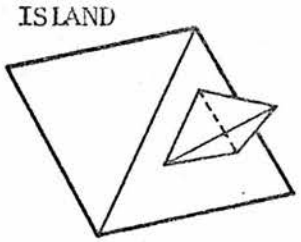
Including the two surfaces of A and B in the same matrix would imply that any operation which can be performed on A or B separately is equally valid when carried out on the composite matrix of A+B. One of the operations which can be carried out on A is to calculate its volume content. If the same process were to be applied to the composite matrix of A+B the answer would include the volume of A.B twice.

VOLUMES WITH MULTIPLE SURFACES

This demands an investigation of the ways in which composite matrices can be created for volumes whose surfaces come into contact or cross.

The simplest case occurs where two volumes have co-planar faces. If this does not occur through construction it may be necessary to cope with it happening as a result of moving a volume from one place to another. Where the vertices and edges do not touch or cross the possible results are shown in Figure 10.





(a)

A	1	4	3				
3	B	1	2				
1	2	C	4				
4	3	2	D				
				$\bar{A}$	5	8	7
				7	E	5	6
				5	6	F	8
				8	7	6	G

(b)

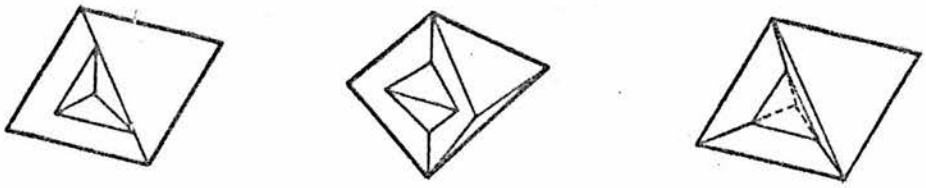
A	1	4	3	5	8	7
3	B	1	2			
1	2	C	4			
4	3	2	D			
7				E	5	6
5				6	F	8
8				7	6	G

Two Volumes with a Common Face in Contact

Figure 10.

Form 'a' is preferred to form 'b' to avoid the line-naming problem. The situation which occurs where two facets of one volume are in contact with two facets of a second volume has already been demonstrated. Here again the partitioned matrix is preferred to avoid the line-naming problem.

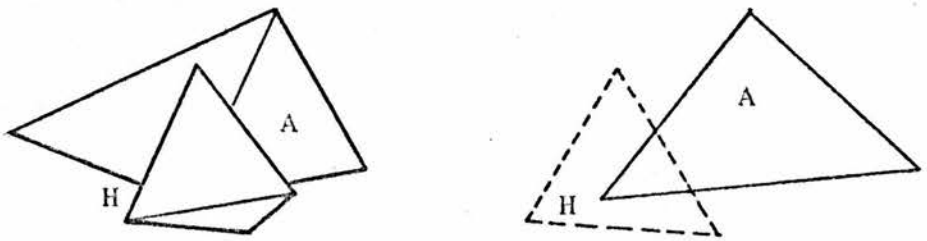
VOLUMES WITH MULTIPLE SURFACES



One, Two, and Three Face Contact

Figure 11

Even when two objects only have one plane in contact, if no constraint is placed on the shape or positioning of the two volumes there are many possible geometrical relationships. Consider the simple cases shown in Figure 12.

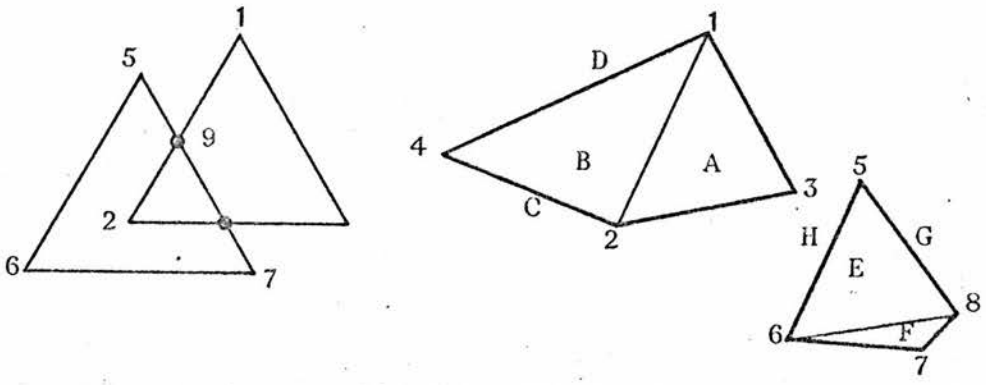


Volumes with a Common Face and Overlapping Edges

Figure 12.

If the description of this object is left as two separate matrices, then there will be eight triangular faces. However, if an attempt is made to form the composite matrix then the shaded area on surface A and the corresponding area on surface H first of all has to be defined and secondly removed. This involves locating the points P9 and P10 (Figure 13).

VOLUMES WITH MULTIPLE SURFACES

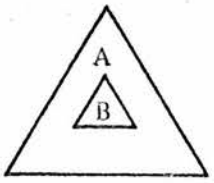


Points of Intersection on Edges

Figure 13.

In the ensuing manipulations of the volume data structure the line segments 9/2, 2/10 will need to be changed from A/B and A/C to H/B and H/C and the new segment 9/10 created as H/A. The subsequent book-keeping will also have to set up the line segments 5/9, 10/7, 9/1, and 10/3. In simple cases this can be done using similar index operations to those described above. When more complex situations occur it would seem to be a better approach to leave objects with a common plane or common planes as separate objects.

A summary of the operations which will need to be performed if the creation of the composite matrix is required for volumes with one or more surfaces in contact, can be given in the following way:



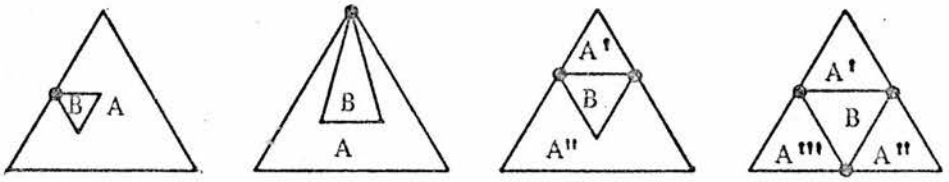
Complete Islands and Holes

Figure 14.

VOLUMES WITH MULTIPLE SURFACES

Complete Islands and Holes

If face A and B are merged, where A is greater than B, then all lines called B/* must be re-entered as A/*, and the facet B must be removed.

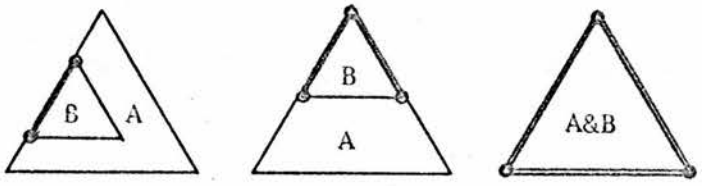


Tangent Points

Figure 15.

Tangent Points

If faces A and B are merged, where A is greater than B, then the tangent point or points must be entered into the lines A/*. Facet B must be removed and all lines B/* replaced as A/*.



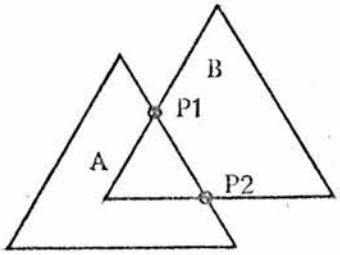
Tangent Edges

Figure 16.

Tangent Edges

If faces A and B are merged, where A is greater than or equal to B, then B can be removed. The points of the boundary of B which lie on the boundary of A where they do not already exist in the description of A

must be added to A. The tangent line segments with labels A/* and B/@ must be renamed */@. The non-tangent sections of the boundary of B, named B/* must be renamed A/*.



Edge Intersections

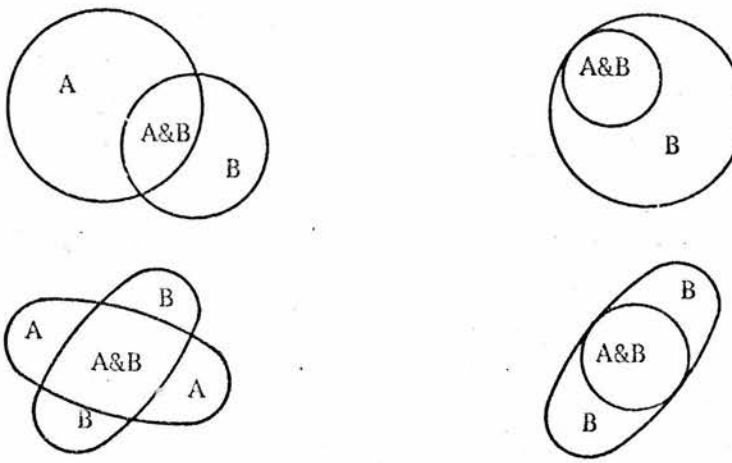
Figure 17.

Edge Intersections

Where faces A and B are merged, the number of intersection points in the boundary of A or B will always be a multiple of 2. This is discounting the normal problems associated with intersection counts, when a line segment cuts a line arc at one of its vertex points. Neither face A nor face B will be totally removed. The intersection points P1 and P2 must be found and entered into the boundaries of both A and B. Overlapped line segments with the labels A/* and their corresponding segments labelled B/@ must be renamed */@, while line segments labelled A/* and B/* which bound areas of A and B which have been lost in the merge must be renamed B/* and A/* respectively.

Multiple Tangents and Intersections

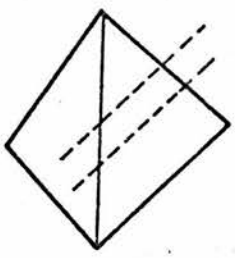
Where there are more than two intersection points or more than one tangent then it is possible for the merging faces to be divided into isolated patches. This may well require new names to be defined.



Multiple Tangents and Intersections

Figure 18

It is not necessary for line-naming problems to occur but, on the other hand there is nothing to rule out the situation shown in Figure 19 from arising.



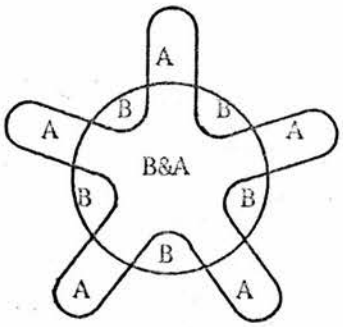
Intersections: Naming Problem

Figure 19

The number of new names will depend on the particular case. The number of subdivisions created by intersections and tangents can be given by the expression

$$n + r$$

where n is the number of intersection points and r is the number of tangent points. Since two of these facets existed initially, the number of new facets which have to be created can be given by the expression : n+r-2, if faces with holes in them are not considered.

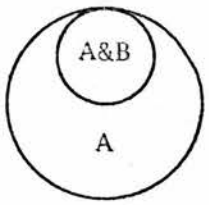


10 Intersection Points

5 A Facets

5 B Facets

8 New Facets



1 Tangent

1 A Facet

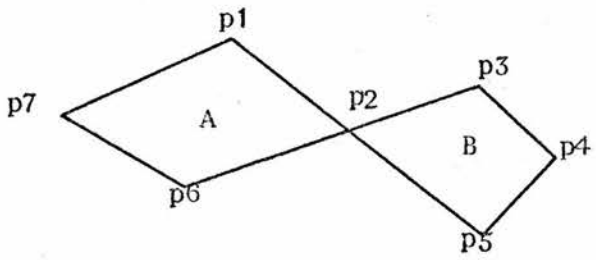
-1 New Facet

Relationship between Tangents, Intersections and New Facets

Figure 20

Other situations which may be affected by the desire to use component matrices where possible over composite matrix forms have to be considered. It is possible for two volumes to be in contact in a variety of ways without actually intersecting. It is necessary to review these in a systematic way to ensure that none of them creates problems. The corresponding situation when processing polygons occurs where two loops

have a point in common. Where polygon boundaries are created automatically as the output from other processes: as a stream of points, it is possible for this kind of polygon to be created in two ways:



Polygon Form 1

p1,p2,p3,p4,p5,p2,p6,p7,p1

Polygon Form 2

p2,p3,p4,p5,p2 B

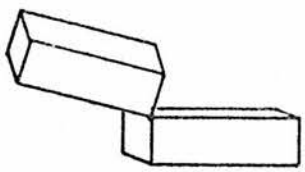
p2,p6,p7,p1,p2 A

Polygons with Point Contact

Figure 21.

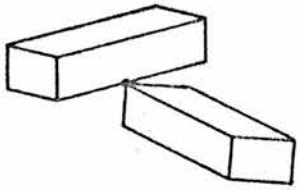
The various forms of contact between volumes can be illustrated in the following way:

(1)



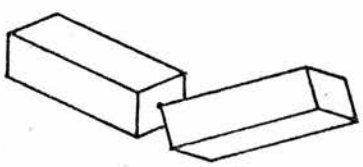
Vertex Vertex Contact

(2)



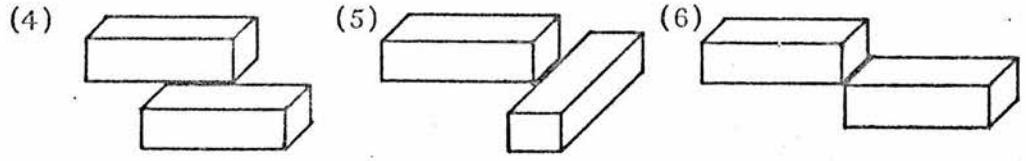
Vertex Edge Contact

(3)

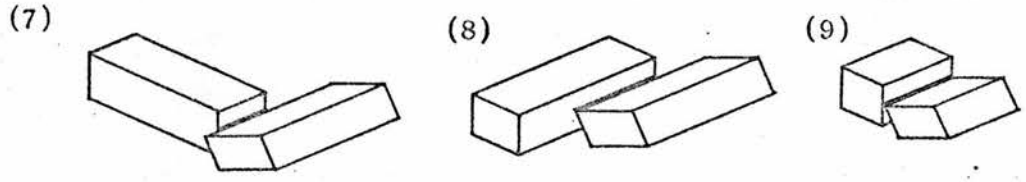


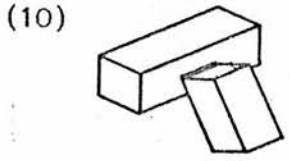
Vertex Face Contact

VOLUMES WITH MULTIPLE SURFACES

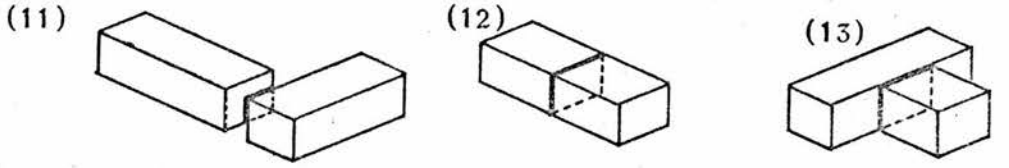


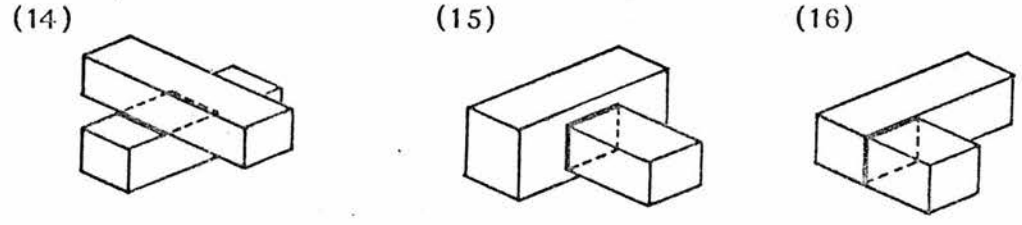
Edge Edge Contact





Edge Face Contact





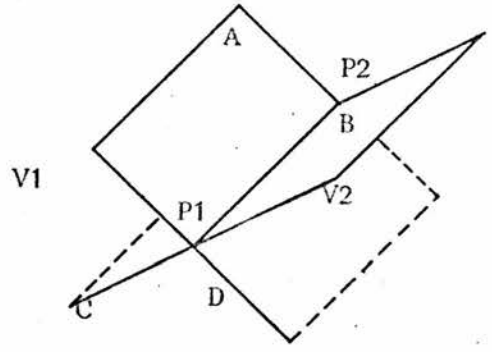
Face Face Contact

Different Volume Volume Contact Conditions

Figure 22.

Since closed loops with a single point are not represented in the matrix

form, the three conditions of vertex contact will not create difficulties by themselves. The edge contacts may or may not create difficulties depending on how they are created. Since an edge is represented in the dual way A/B and p_1/p_2 there are two possible ways in which an edge from one volume may affect the description of another volume. The first is by unnecessary points appearing in the matrix. The second is more important in that greater care is needed to make sure the procedure's bookkeeping is correct. In this case the edge names may be formed from facets from both volumes.



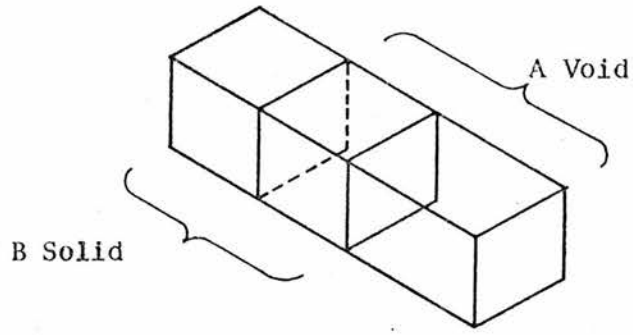
Edge Naming where Volumes are in Edge Contact

Figure 23.

In the case shown in the diagram in Figure 23 line p_1/p_2 can be labelled as A/C and B/D as correctly as A/B and C/D. This result could well arise from all the cases from 4 to 10, though in 8 and 10 it is more readily detected and removed. The facet/facet contact has already been discussed. Case 15 is the island or hole condition, 13, 16 and 12 the tangent situations where 12 shows the merging of identical faces. 11 and 14 show the two and four boundary intersection conditions.

All these situations consider solids with external contact. Where

VOLUMES WITH MULTIPLE SURFACES

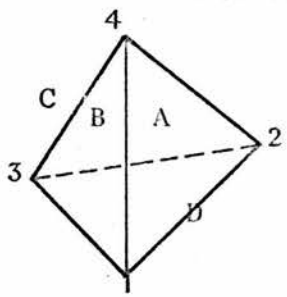


Interaction of Solid and Void

Figure 24.

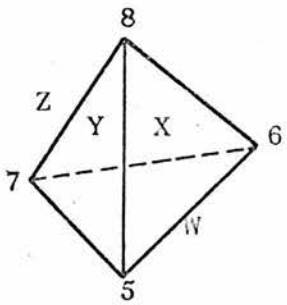
internal contact and solid void combinations are considered it leads into the problem of crossing surfaces and of volume intersection. It is likely that edge and face coincidence will be just as important in this case, particularly in the kind of situation shown in Figure 24.

V1



A	1	4	2
4	B	3	1
2	4	C	3
1	3	2	D

V2



W	5	7	6
6	X	5	8
5	8	y	7
7	6	8	Z

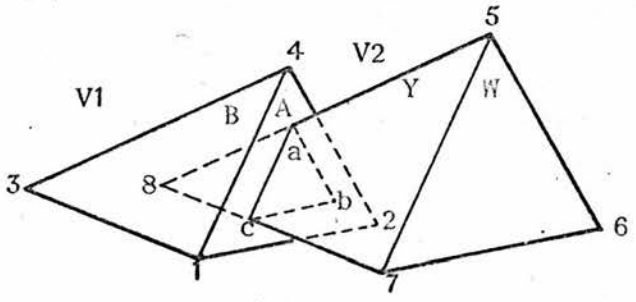
Two Tetrahedra: Volume Intersection

Figure 25.

VOLUMES WITH MULTIPLE SURFACES

Taking a simple case to start with, consider the intersection of the tetrahedra V1 and V2 shown in Figure 25. If volume V2 penetrates volume V1 so that point 8 lies inside V1, and line 8/5 cuts facet A at a, line 8/6 cuts facet A at b, and line 8/7 cuts facet A at c, then it should be possible to obtain the union or intersection of these two volumes from this information using matrix manipulations: The first objective is the union of V1 and V2 which is illustrated in Figure 26.

Having formed the composite matrix V1 + V2, the first stage is to



The Union of V1 and V2

Figure 26.

remove point 8, since it will no longer be a surface point. Where 8 occurs in this matrix, the appropriate new point from a, b, c will have to be inserted to define the lines a/5, b/6, and c/7 as Y/X, X/Z, and Z/Y respectively. After this the remaining operations should depend on bookkeeping operations to give the matrix a correct structure.

The bookkeeping operation depends on the matrix property that the indexes of a point can be ordered in a sequence of the form @/*, */&, &/@. Thus the operation

$$V(Y,X) \longrightarrow a$$

A	1	4	2				
4	B	3	1				
2	4	C	3				
1	3	2	D				
				W	5	7	6
				6	X	5	b
				5	a	Y	7
				7	6	c	Z

Replacing the Lost Vertex '8'.

Figure 27.

where Y/X has cut plane A means that the following operations are also necessary:

$V(X,A) \longrightarrow a$ and $V(A,Y) \longrightarrow a$

similarly:

$V(X,Z) \longrightarrow b$

where X/Z has intersected plane A

$V(Z,A) \longrightarrow b$

$V(A,X) \longrightarrow b$

and:

$V(Z,Y) \longrightarrow c$

where Z/Y has cut plane A

$V(Y,A) \longrightarrow c$

$V(A,Z) \longrightarrow c$

Since in this case all the new entries V(*,@) have matching entries V(@,*) the final matrix becomes:

VOLUMES WITH MULTIPLE SURFACES

A	1	4	2		b	a	c
4	B	3	1				
2	4	C	3				
1	3	2	D				
				W	5	7	6
a				6	X	5	b
c				5	a	Y	7
b				7	6	c	Z

The Complete Matrix for V1 + V2

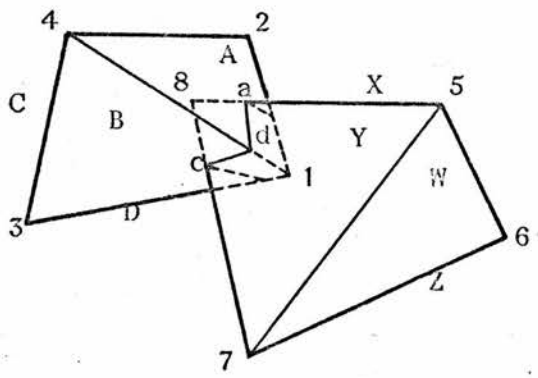
Figure 28.

In the previous example only lines from one of the volume descriptions were cut by the other volume. If the relationship between V1 and V2 in space had been that shown in Figure 29, then following the previous procedure:

$V(Y,X) \longrightarrow a$

line Y/Z intersects plane A: $V(X,A) \longrightarrow a$, $V(A,Y) \longrightarrow a$

$V(X,Z) \longrightarrow b$



Alternative Spatial Relationship of V1 and V2

Figure 29

CHAPTER 9

VOLUMES WITH MULTIPLE SURFACES

VOLUMES WITH MULTIPLE SURFACES

line X/Z intersects plane D: $V(Z,D) \longrightarrow b$, $V(D,X) \longrightarrow b$

$V(Z,Y) \longrightarrow c$

line Z/Y intersects plane B: $V(Y,B) \longrightarrow c$, $V(B,Z) \longrightarrow c$

Once this has been done the matrix entries: V(A,X), V(Y,A), V(D,Z), V(X,D), V(B,Y) and V(Z,B) have not been accounted for. If these entries are ordered using the indices, the result is:

V(A,X), V(X,D)	which requires	V(D,A)
V(D,Z), V(Z,B)	which requires	V(B,D)
V(B,Y), V(Y,A)	which requires	V(A,B)

V(D,A), V(B,D), V(A,B) already contain the point 1. This point lies inside V2. Apparently, this fact has been deduced without having to test the lines of V2 against the facets of V1. The actual points which will replace 1 still have to be calculated, but it would appear to be a matter of finding the intersection of D/A with X, D/B with Y, and B/A with Z. If these points are named f, e, and d respectively, then the composite matrix for the union of V1 and V2 will be that shown in Figure 30.

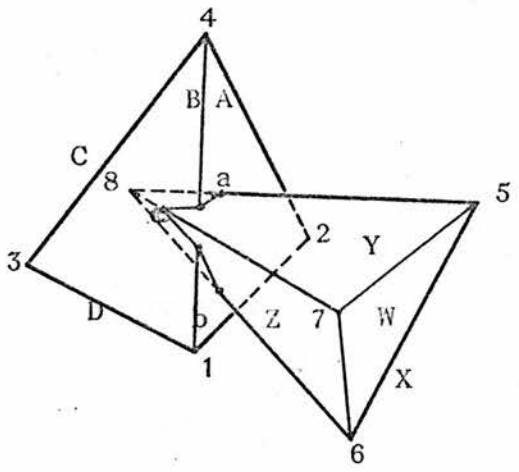
A	d	4	2		f	a	
4	B	3	e			d	c
2	4	C	3				
f	3	2	D		b		e
				W	5	7	6
a			f	6	X	5	b
d	c			5	a	Y	7
	e		b	7	6	c	Z

Volume Matrix for V1 + V2, shown in Figure 29.

Figure 30.

VOLUMES WITH MULTIPLE SURFACES

If the volumes are arranged in the following way it can be seen that a line-naming problem will arise.



Third Relationship of V1 and V2

Figure 31.

If the lines of V1 are tested against the facets of V2 then the problem will become immediately apparent. However, if V2 is tested against V1 it will initially appear to be a similar case to the previous example:

V(Y,X)⟶a

line X/Y intersects plane A: V(X,A)⟶a, V(A,Y)⟶a

V(X,Z)⟶b

line X/Z intersects plane A: V(Z,A)⟶b, V(A,X)⟶b

V(Z,Y)⟶c

line Z/Y intersects plane B: V(Y,B)⟶c, V(B,Z)⟶c

However, when sorting the unmatched cells:

V(B,Y), V(Y,A) which requires V(A,B)

V(A,Z), V(Z,B) which requires B(B,A)

This would mean that both end points of A/B would be overwritten if the

same approach were adopted as before.

Until it was shown to be insufficient by itself, this situation was used to indicate the existence of the naming problem. On one hand, neither V(A,B) nor V(B,A) could be overwritten without making the matrix incorrect. On the other, the naming problem and this situation could both be solved by creating the new face names Ia and Jb. To achieve this, all previous entries into A or B had to be transferred to Ia and Jb respectively. If tests for the naming problems had been carried out before any new entries into the matrix were made, it would have saved having to transfer these entries when new face names were found to be necessary. If the two new points required are called d and e, the composite matrix becomes:

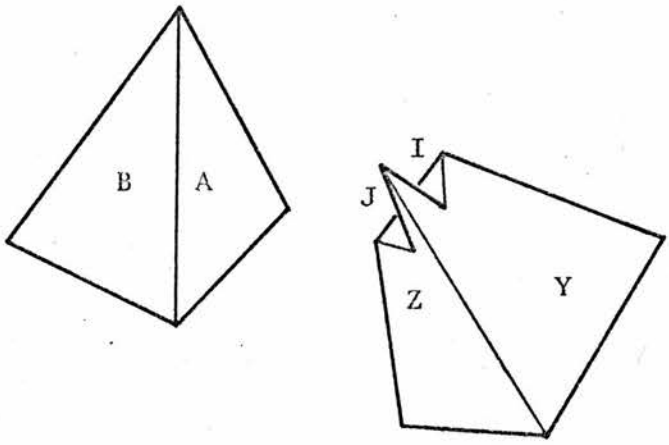
A	1	4	2						
4	B	3	1						
2	4	C	3						
1	3	2	D						
				W	5	7	6		
				6	X	5	b	a	
				5	a	Y	7	d	c
				7	6	c	Z	b	e
					b	a	e	I	d
						d	c	e	J

The Naming Problem: V1 + V2 as a Partitioned Matrix

Figure 32.

This partitioned matrix represents the two separate objects shown in the

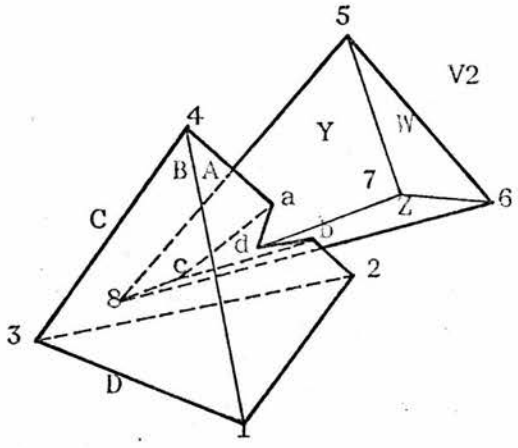
diagram. When the naming problem occurs simultaneously for both objects



Volumes which Correspond to the Partitioned Matrix

Figure 33.

the problem is treated in a similar way. An example of the way this situation can come about is shown by the two volumes:



A Double Naming Problem for V1 + V2

Figure 34.

If V2 is tested against V1 it will be found that line Y/Z, 7/8 cuts face A in d, and face C in c; so creating a naming problem for Y/Z. If new faces M:Y and N:Z are defined, then:

$V(M,N) \longrightarrow d$

M/N cuts face A at d: $V(N,A) \longrightarrow d$, $V(A,M) \longrightarrow d$

$V(N,M) \longrightarrow c$

N/M cuts face C at c: $V(M,C) \longrightarrow c$, $V(C,N) \longrightarrow c$

This gives the unassigned cells:

V(M,A), V(A,N), V(N,C), and V(C,M)

which when they are ordered define the new points:

V(A,N), V(N,C) .... V(C,A): point a

V(C,M), V(M,A) .... V(A,C): point b

This identifies another naming problem so, setting up the new facet names - I:A and J:C - the new composite matrix becomes:

A	1	4	2								
4	B	3	1								
2	4	C	3								
1	3	2	D								
				W	5	7	6				
				6	X	5	8				
				5	8	Y	7				
				7	6	8	Z				
								N	c	d	a
								d	M	b	c
								a	d	I	b
								c	b	a	J

V1

V2

$\overline{V1 \cdot V2}$

Partitioned Matrix for the Volumes with a Double-Naming Problem

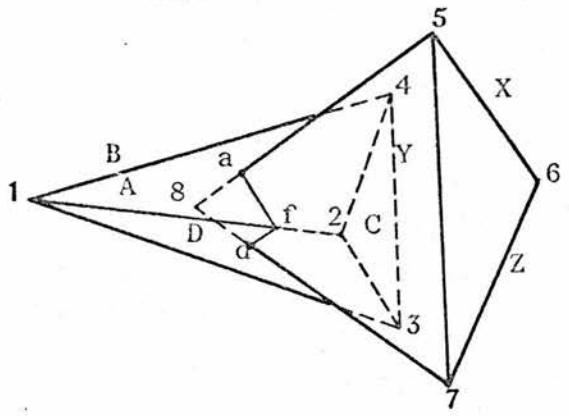
Figure 35.

It is interesting in this example that the composite matrix has been partitioned into three sections, each representing a tetrahedron. The original two volumes are present in the first two submatrices, and the

third matrix is the complement of the intersection of these two. This is an approach which could be used to define the union of any volumes, and is a subject which is returned to at a later stage.

These experiments have assumed that the intersection points on the edges of the second volume are effectively defined once the intersection points for edges of the first volume have been found. This has been true for the examples chosen but is not always going to be true. Another problem to be resolved is to find out what happens to internal points of the second volume, when there is a large number of them. The following example was considered to determine whether these points could also be removed by simple matrix bookkeeping procedures.

Consider the volumes V1 and V2 orientated in the following way:



The Creation of an Included Facet

Figure 36.

If V2 is tested against V1 it will be found that line 5/8 cuts A at a, 6/8 cuts B at b, and 7/8 cuts D at d.

$$V(Y,X)\longrightarrow a,\ V(X,A)\longrightarrow a,\ V(A,Y)\longrightarrow a$$

$$V(Z,Y)\longrightarrow d,\ V(Y,D)\longrightarrow d,\ V(D,Z)\longrightarrow d$$

$$V(X,Z)\longrightarrow b,\ V(Z,B)\longrightarrow b,\ V(B,X)\longrightarrow b$$

VOLUMES WITH MULTIPLE SURFACES

Re-ordering the diagonal cells to define the remaining points:

V(A,X), V(X,B)	requires	V(B,A)	: e
V(D,Y), V(Y,A)	requires	V(A,D)	: f
V(B,Z), V(Z,D)	requires	V(D,B)	: g

Since there are no naming problems the matrix is transformed by these entries in the following way:

A	1	4	2				
4	B	3	1				
2	4	C	3				
1	3	2	D				
				W	5	7	6
				6	X	5	8
				5	8	Y	7
				7	6	8	Z

→

A	1	4	f		e	a	
e	B	3	1		b		g
2	4	C	3				
1	g	2	D			f	d
				W	5	7	6
a	e			6	X	5	b
f			d	5	a	Y	7
	b		g	7	6	d	Z

Stage I, Insert New Points

Figure 37.

It can be seen that e, f, g have replaced points 2, 3, 4 in the V1 submatrix, quite correctly when the diagram is examined. However, these points are still represented in the final matrix. These remaining entries need to be removed. If the column positions of the substituted points in V1 are used to create a vector of these points, then the matrix multiply will locate all the remaining occurrences of these points.

What this is doing is to follow the facets in the order in which they occur round these points. It can consequently be used to systematically remove all the entries of these points in other facet descriptions.

$$(4,\ 3,\ 0,\ 2)\ .\ \begin{bmatrix} 0, & 1, & 4, & f \\ e, & 0, & 3, & 1 \\ 2, & 4, & 0, & 3 \\ 1, & g, & 2, & 0 \end{bmatrix} = (0,\ 0,\ 4+3+2,\ 0) \quad (1)$$

$$= (2,\ 4,\ 0,\ 3) \quad (2)$$

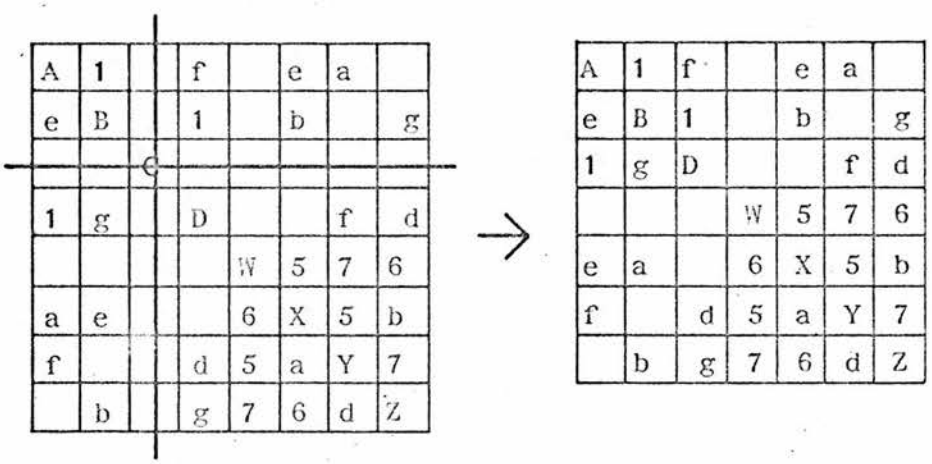
$$= \text{Starting Position}$$

$$\begin{bmatrix} 0, & 1, & 4, & 4 \\ e, & 0, & 3, & 1 \\ 2, & 4, & 0, & 3 \\ 1, & g, & 2, & 0 \end{bmatrix} \qquad \begin{bmatrix} 0, & 1, & 0, & 4 \\ 0, & 0, & 0, & 1 \\ 0, & 0, & 0, & 0 \\ 1, & g, & 0, & 0 \end{bmatrix}$$

Stage II, Removing Internal Facets

Figure 38

What is not shown in this example, but which is fairly obviously going to be true is that all the facet rows and columns which have points deleted from them by this process, but which are not the rows containing the original substituted points, can also be deleted. This fact can be used to simplify this garbage collection procedure. Once the first multiplication has been completed it simply becomes a matter of systematically deleting rows and columns. This gives the final matrix for V1+V2 shown in Figure 39.

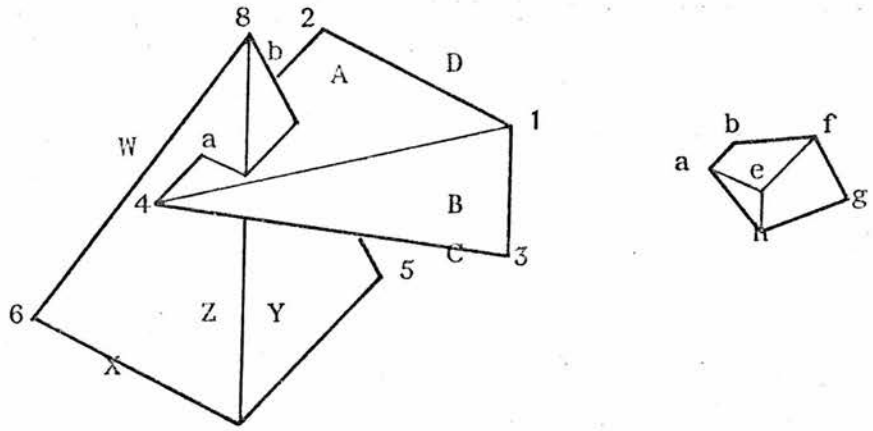


Final Matrix with All Internal Facets Removed

Figure 39.

This example is not the only case where the simple procedure adopted for the earlier examples must be extended. If the two tetrahedra V1 and V2

are related to each other as shown in Figure 40 though it is possible



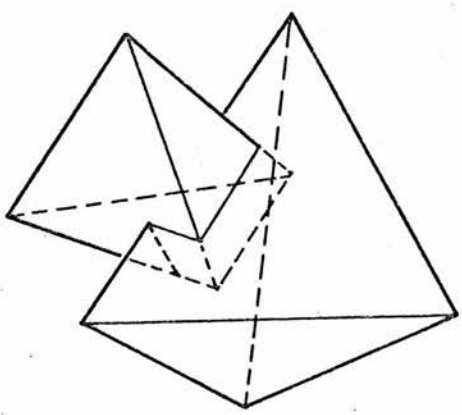
Volume Relationship which Creates Difficulties

Figure 40

to carry out the first stage as before, the second stage breaks down. If V1 is tested against V2, then it will be found that line 2/4, C/A, cuts face Z at a and also face W at b. This indicates a naming problem, so, defining the new face names IC: C and IA: A, these points can be entered in the usual way:

$$V(IC,IA) \longrightarrow a, \quad V(IA,Z) \longrightarrow a, \quad V(Z,IC) \longrightarrow a.$$

$$V(IA,IC) \longrightarrow b, \quad V(IC,W) \longrightarrow b, \quad V(W,IA) \longrightarrow b.$$



The Alternative Volume Relationship

Figure 41.

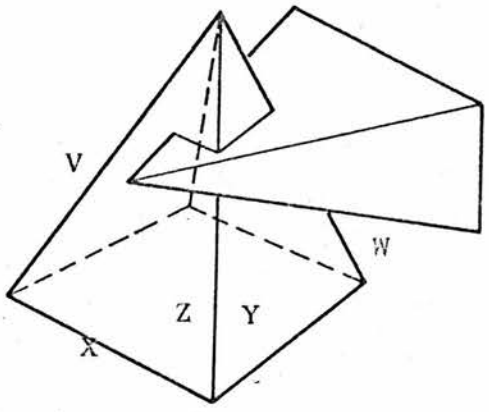
However, the next stage would give:

V(W,IC), V(IC,Z) .... V(Z,W)

V(Z,IA, V(IA,W) .... V(W,Z)

This suggests that there is a naming problem and that line Z/W is cut by V1. Though in this case there will be a naming problem it cannot be assumed that Z/W is cut or that there is a naming problem from the information so far obtained by testing V1 against V2 because the volumes could be in the relationship shown in Figure 41.

It can be seen in this diagram that line Z/W is not cut by volume V1. Moreover, had V2 been a square based pyramid in the position shown in Figure 42 then line ZW would not exist. It is still possible to apply



Further Difficulties

Figure 42.

the same approach adopted in the previous examples to this extent: all the edges of the facets Z and W must now be tested against volume V1. Consequently, the information gained in the first set of tests will still reduce the number of tests but V2 must be tested against V1. In this

case the tests will show that: Z/Y cuts facet A at e and facet C at h, and Y/W cuts A at f and C at g. This identifies two naming problems, so renaming: Z:IZ, Y:IY, W:IW:

$V(IZ,IY) \longrightarrow h$, $V(IY,IC) \longrightarrow h$, $V(IC,IZ) \longrightarrow h$

$V(IY,IZ) \longrightarrow e$, $V(IZ,IA) \longrightarrow e$, $V(IA,IY) \longrightarrow e$

$V(IY,IW) \longrightarrow g$, $V(IW,IC) \longrightarrow g$, $V(IC,IY) \longrightarrow g$

$V(IW,IY) \longrightarrow f$, $V(IY,IA) \longrightarrow f$, $V(IA,IW) \longrightarrow f$

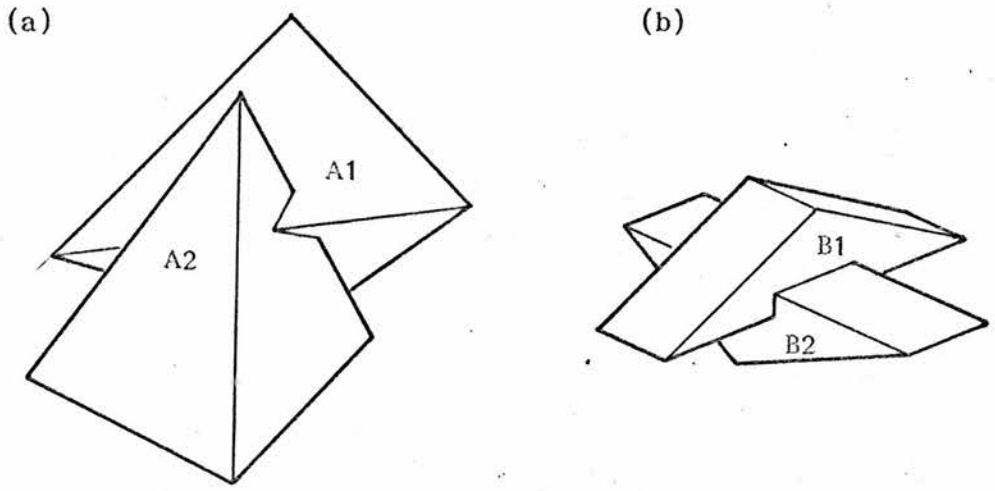
A	1	4	2									
4	B	3	1									
2	4	C	3									
1	3	2	D									
				W	5	7	6					
				6	X	5	8					
				5	8	Y	7					
				7	6	8	Z					
								IA	b	f	e	a
								a	IC	b	g	h
								b	g	IW	f	
								f	h	g	IX	e
								e	a		h	IY

Complete Matrix

Figure 43.

which gives the final correct form to the matrix shown in Figure 43. This example reopens several problems which had apparently been resolved. In particular, the renaming operation is more complex than was first thought. The issues are high-lighted by the comparison of the two composite volumes (a) and (b):

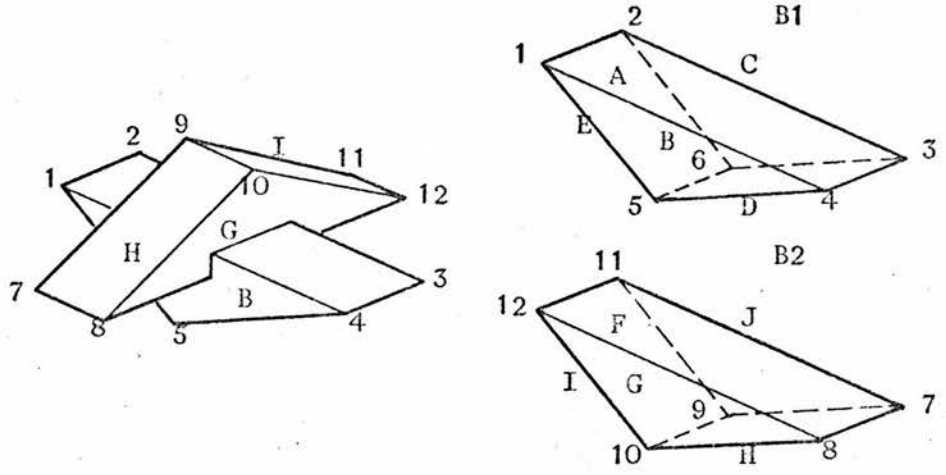
VOLUMES WITH MULTIPLE SURFACES



Variation to the Naming Problem

Figure 44.

In the case of (a) there was no alternative to the creation of the complement of the intersection of volumes A1 and A2; whereas in (b) there are two approaches to the way in which the renaming can be carried out.

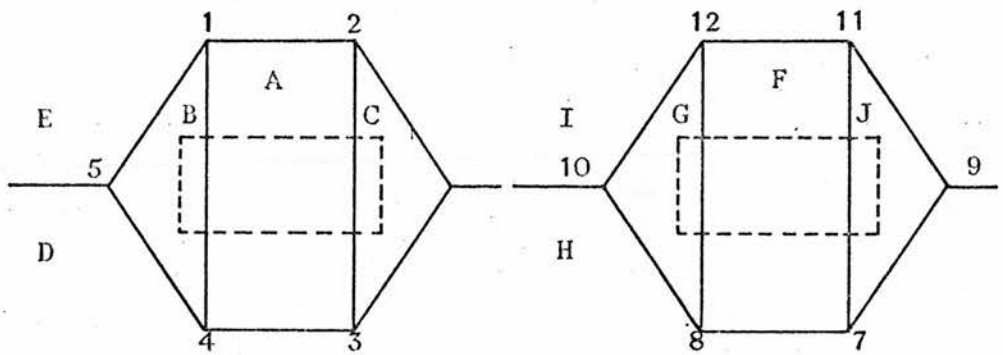


Volume with Alternative Solutions to the Naming Problem

Figure 45.

VOLUMES WITH MULTIPLE SURFACES

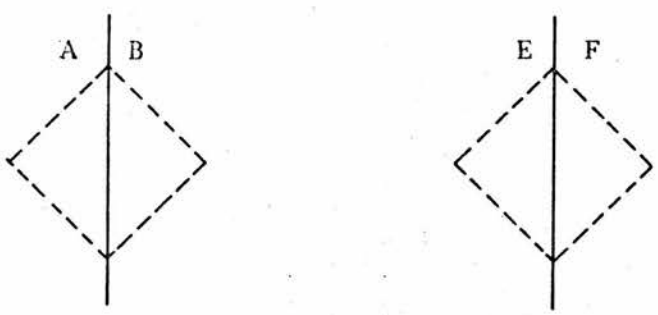
It can be seen from the diagram that lines: 1/4, 2/3, 8/12 and 7/11 will each be cut to give naming problems. This can be solved by the approach taken in a previous example: the creation and subtraction of the volume of intersection of B1 and B2. However, if the exploded view of these volume surfaces is examined a second approach becomes apparent.



Exploded View

Figure 46.

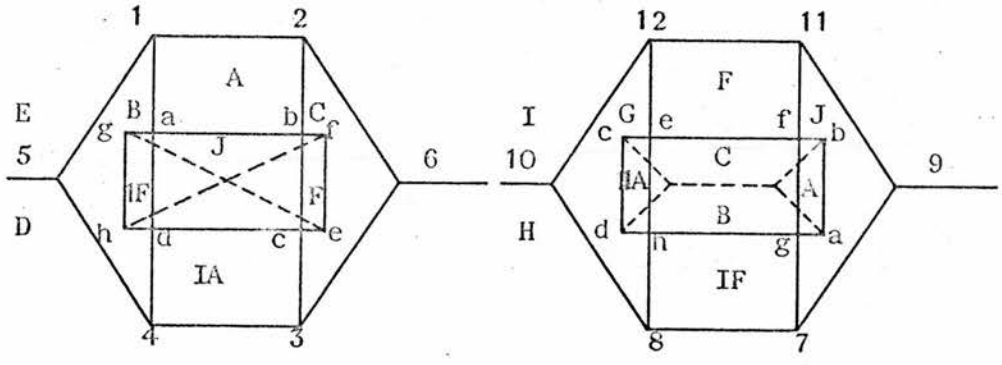
The naming problem arises for the lines B/A, A/C, G/F, and F/J. It can also be seen that this has come about because facets A and F have been divided into two separate zones. In the previous case, the situation is summarised by Figure 47 but not all the facets creating the naming



Simple Naming Problem

Figure 47

problem in Figure 44 are subdivided in this way. This means that in the current case, instead of having to create all the new facets necessary to define the intersection of B1 and B2: IA, IB, IC, IG, IF, IJ; merely the extra zones of F and A need new names. Not only does this resolve the naming problem, but also, it permits the composite matrix of B1+B2 to be created in a non partitioned form:



Alternative Renaming

Figure 48.

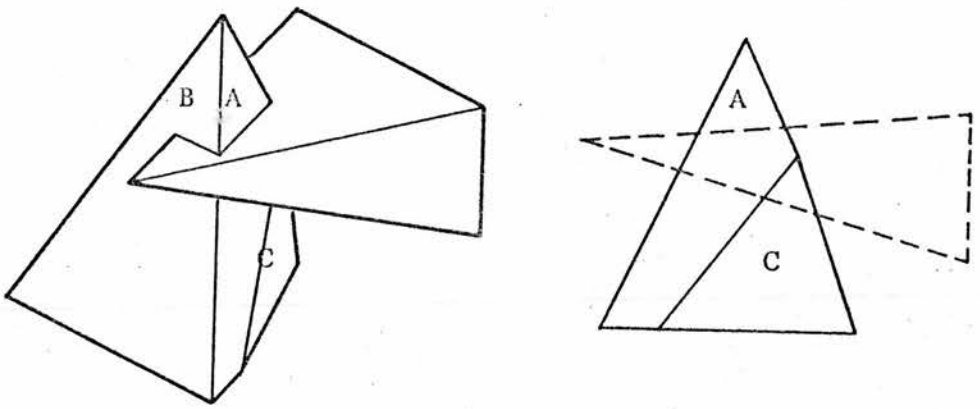
A	a	2		1							b
1	B		4	5	d	g		h			a
b		C	6	2	3		e	c			f
	5	3	D	6	4						
2	1	6	5	E							
	4	c	3		IA			d			
	h					IF		8	7		9
		f					F	e		12	11
	d	e			c	h	12	G	8	10	
						8		10	H	9	7
							11	12	10	I	9
a	g	b				7	f		9	11	J

Volume Matrix using Alternative Naming Strategy

Figure 49.

VOLUMES WITH MULTIPLE SURFACES

In this case neither the partitioned nor the unpartitioned form of matrix seems to be intrinsically better or worse than the other. However, when the interaction of the two volume descriptions: V1 and V2 in Figure 50 are considered, it is clear that the creation of the



Volume Demanding the Alternative Approach

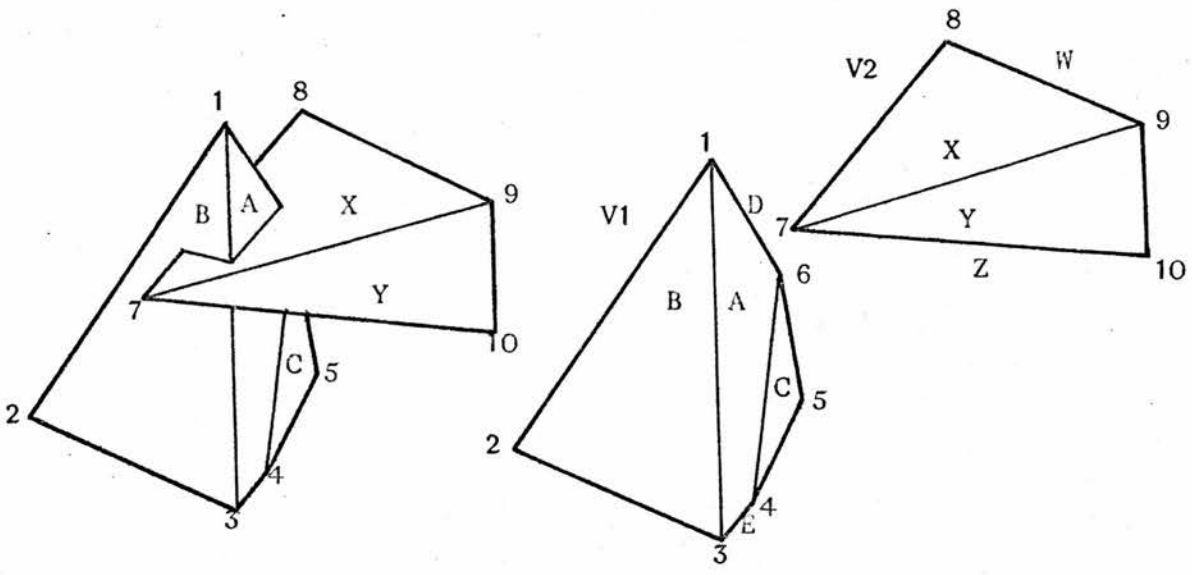
Figure 50

volume of intersection will not occur merely by renaming A and B as a consequence of line A/B being cut twice. In this case, the second approach to renaming must be adopted if the composite matrix for V1+V2 is to be formed. The question which this poses is how to determine, automatically, which of the names, A or B, should be duplicated. If all the intersection points of the edges of V1 are listed:

B/A:a, B/A:b, A/D:c, A/C:d, C/D:e

then it can be seen that edges of A are cut four times at different points. Based on the previous discussion of tangent/intersection counts, this indicates that it is facet A which will be subdivided into two separate zones. If the partitioned matrix form is desired for V1+V2 it can be obtained in the following way:

VOLUMES WITH MULTIPLE SURFACES



Implementing the Alternative Approach

Figure 51.

line X/Z:7/8 cuts facet B at f, and facet D at g:

V(Z,X)⟶f, V(X,B)⟶f, V(B,Z)⟶f.

V(X,Z)⟶g, V(Z,D)⟶g, V(D,X)⟶g.

line B/A cuts facet X at a, and facet Z at b:

V(A,B)⟶a, V(B,X)⟶a, V(X,A)⟶a.

V(B,A)⟶b, V(A,Z)⟶b, V(Z,B)⟶b.

line D/A cuts facet X at c:

V(D,A)⟶c, V(A,X)⟶c, V(X,D)⟶c.

line D/C cuts facet Z at e:

V(C,D)⟶e, V(D,Z)⟶e, V(Z,C)⟶e.

It can be seen that, in this case, not only the boundary lines of B and D have to be tested against facets X and Z of V2, but also the boundary lines of other facets which are adjacent to boundary lines of B and D which cut X or Z. In this case this means facets A and C. The only intersection point in this case is where line A/C cuts Z at d:

V(A,C)⟶d, V(C,Z)⟶d, V(Z,A)⟶d.

This comparison will also indicate that point 6 is an internal point, thus:

V(A,D):6, V(C,A):6, V(D,C):6

which gives the matrix for V1.V2 shown in Figure 52.

V3 = V1.V2

A	a	d	6	c	b
b	B			a	f
6		C	e		d
c		6	D	g	e
a	f		c	x	g
d	b	e	g	f	Z

V1+V2

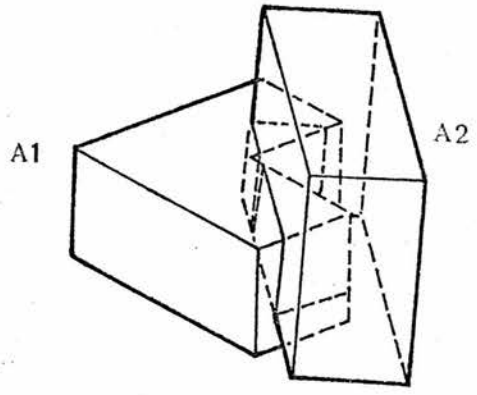
V1		
	V2	
		$\overline{V3}$

$\overline{V3} = \overline{V1.V2}$

A Fully Partitioned Matrix to solve the Naming Problem

Figure 52.

If the two original matrices for V1 and V2 are taken with the transpose of this matrix the result is the partitioned form of V1+V2. However, without restrictions this approach is just as likely to create naming problems, as the volume A1.A2 demonstrates:

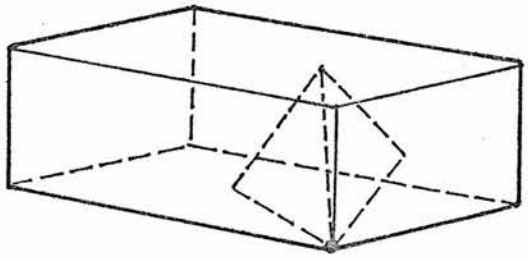


Naming Problem created by Intersection and Subtraction

Figure 53.

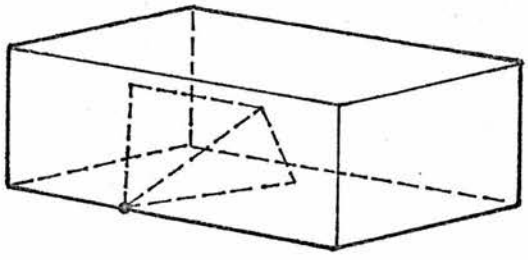
VOLUMES WITH MULTIPLE SURFACES

The next problem is to consider the internal contact of volume surfaces in the Solid/Solid situation, where the two surfaces do not form a volume of intersection. The same basic relationships already illustrated for the solid/solid external contact can be found in the solid/void surface contacts. These situations are more difficult to draw but the comparison is interesting:



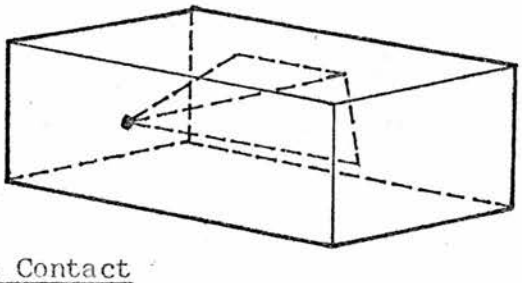
(1)

Vertex/Vertex Contact



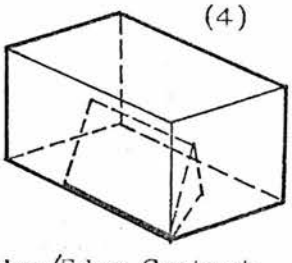
(2)

Vertex/Edge Contact

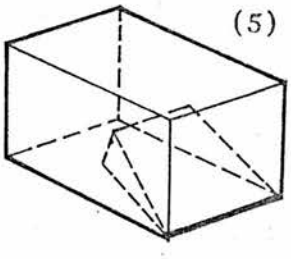


(3)

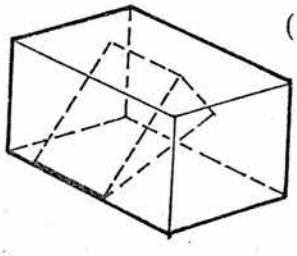
Vertex/Facet Contact



(4)



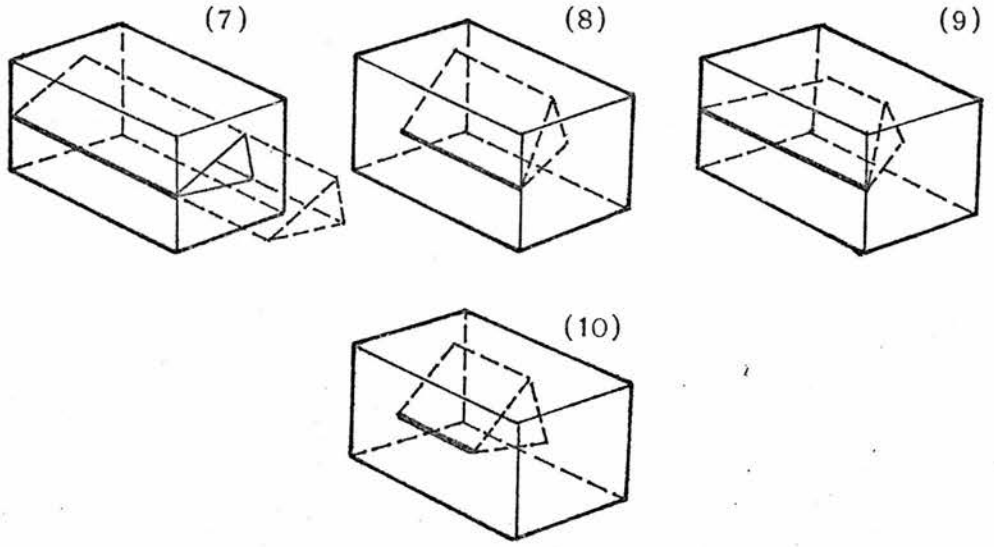
(5)



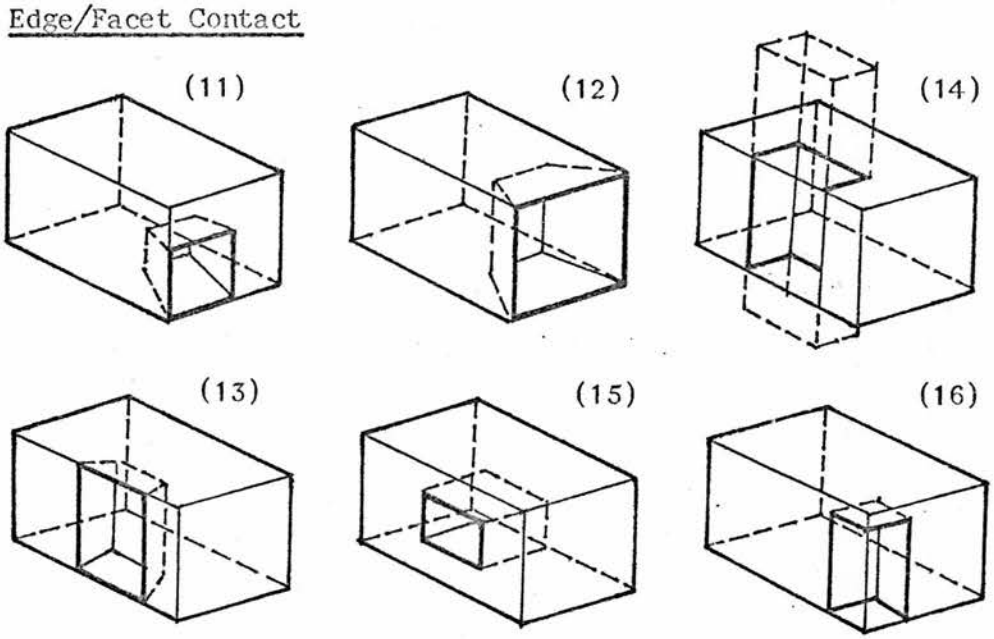
(6)

Edge/Edge Contact

VOLUMES WITH MULTIPLE SURFACES



Edge/Facet Contact



Facet/Facet Contact

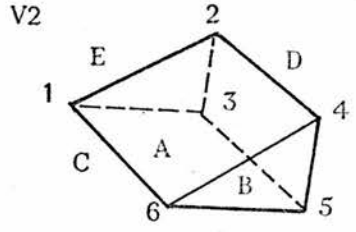
Different Volume Void Contact Conditions

Figure 54

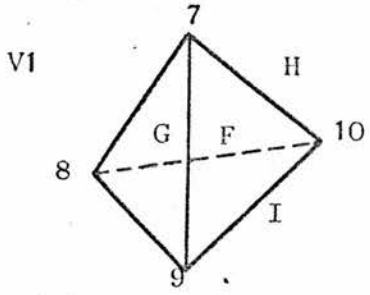
As nearly as possible the corresponding situations to the solid/solid contact diagrams have been illustrated. This has been indicated by the numbering of the figures. It is difficult to be certain that this

is an exhaustive list since it depends on what constitutes a significant variation. If it is merely the effect on the individual surface facets that is relevant then 7 and 9 become equivalent as do 13 and 14, and 16 and 11. However, particularly in the case of 7 it is necessary to establish what happens to the vertex of the triangular opening if the line on the surface is excluded from the solid volume-matrix. Again, it can be seen that vertex contacts can be ignored, but that edge contacts may cause facet/facet line names to be created incorrectly.

It is not possible to know in what form the edge contact conditions will appear, without some knowledge of the process which led to their creation. What can be examined at this stage, is how the different conditions which have been illustrated will affect the matrix structure. Taking case 5 to start with as the simplest form of edge contact in that it does not immediately create naming problems, the following figures correspond to the matrices in Figure 55.



A	4	6	2	1
6	B	5	4	
1	6	C	5	3
4	5	3	D	2
2		1	3	E



F	9	7	10
7	G	8	9
10	7	H	8
9	8	10	I

Solid and Void

Figure 55.

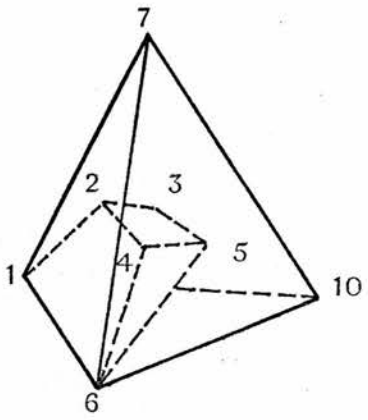
VOLUMES WITH MULTIPLE SURFACES

If volume (V1-V2) is constructed so that point 1 corresponds to point 8 .and point 6 to point 9 the result could be that shown in Figure 56.

$\overline{V2}$

A	6	1	4	2				
4	B	6	5					
6	5	C	3	1				
2	4	5	D	3				
1		3	2	E				
					F	6	7	10
					7	G	1	6
					10	7	H	1
					6	1	10	I

V1



Solid and Void Partitioned Matrix

Figure 56.

However, this partitioned matrix could appear in the form shown in Figure 57.

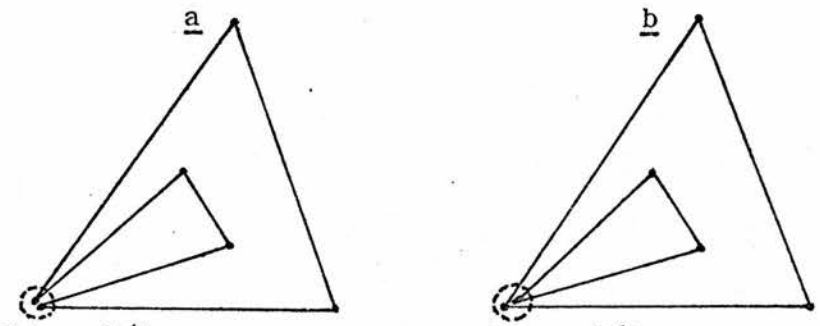
A	6		4	2		1		
4	B	6	5					
	5	C	3	1				6
2	4	5	D	3				
1		3	2	E				
					F	6	7	10
6					7	G	1	
					10	7	H	1
		1			6		10	I

Volume Reduction

Figure 57

Since this matrix is correctly formed, it is reasonable to assume that

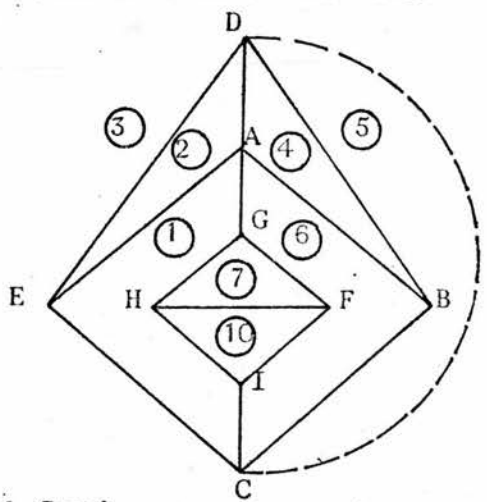
it will not create any problems in any of the matrix operations so far discussed. However, if it is the policy to avoid the forms of representation which lead to a cross section of the form (a) as opposed to (b) it will be necessary to find some way of testing a matrix to find out whether this situation exists.



Alternative Cross Sectional Boundaries

Figure 58

If the dual matrix of the volume is set up, in other words the vertex/area as opposed to the area/vertex matrix, it can be seen that this situation is similar to the naming problem discussed earlier. If the dual graph is drawn, this case can be seen to correspond exactly to the volume which created the first naming difficulties:

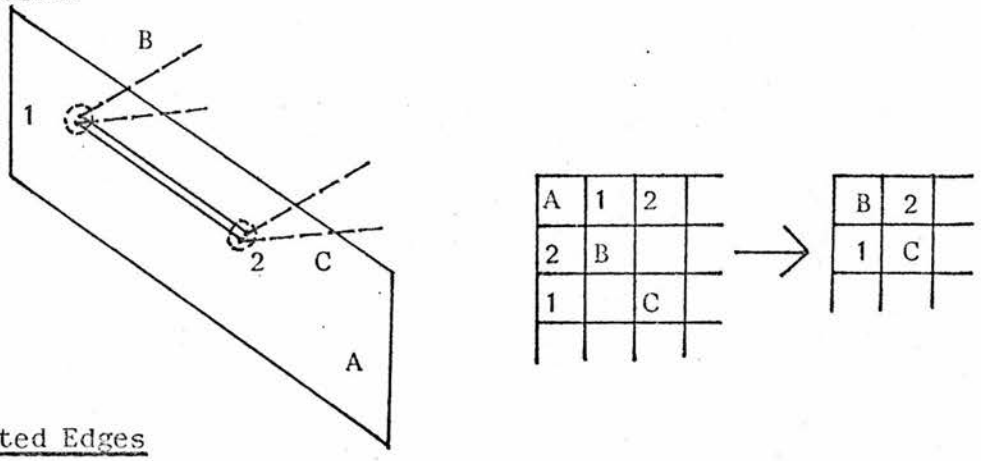


The Dual Graph

Figure 59.

VOLUMES WITH MULTIPLE SURFACES

Most of the remaining edge conditions create naming problems in the area/point matrix. Resolving these problems by creating new facet names will produce the required partitioned matrix for two separate volumes. Condition 10, however, is similar to 5 in that the area/point matrix will be correctly formed and the point/area matrix will be needed to indicate the double presence of the line 1/2 in the situation:



Isolated Edges

Figure 60.

The most efficient way of resolving these conflicts in practice will depend, to a great extent, on the way these matrices are implemented in a system. These matrix manipulations pose a series of practical problems which are unrelated to the logic of their use. The first issue which must be faced is the way in which the overall matrix escalates in size as the number of facets increase. This and other issues which are associated with storing and using these matrices are discussed in the next Chapter.

CHAPTER 10

VOLUME MATRIX STORAGE

VOLUME MATRIX STORAGE

In this Chapter the practical problems associated with storing and accessing volume matrices are considered. The original motivation which led to the development of the matrix data structure for representing volumes was to provide a system which was easier to work with than the complex list and ring structures which had been encountered in much of the existing work in this area. Since most of the examples which were examined in the initial stages of this work were relatively small, the matrix could be implemented as a two-dimensional array in any high level language. It soon became apparent, as more complex volumes were investigated that the size of the matrices was going to escalate to a wasteful if not unmanageable size if this direct approach was retained. The larger the matrix, it appeared, the more of its cells would be empty.

Eulers formula relates the number of faces, edges and vertices which occur in the closed faceted surface of a volume in the following way:

$$n - m + r = 2$$

where n represents the number of facets

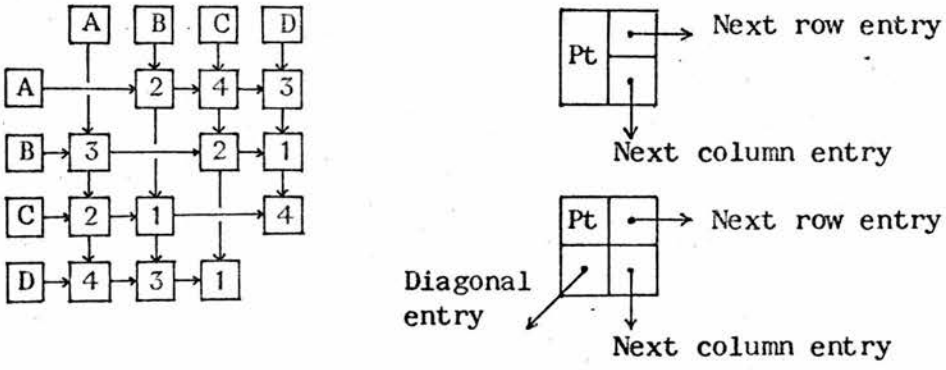
m represents the number of edges

and r represents the number of vertices.

If the Area/Vertex matrix is created then the number of cells in the matrix will be n^2, conversely the number of cells in the Vertex/Area matrix will be r^2. The number of matrix cells used by entries defining edges will be $2.m$. Consequently, the number of empty cells in each of the two matrix forms will be $n^2 - n - 2.m$, and $r^2 - r - 2.m$ respectively. It can be seen that as the matrix size gets larger so the squared term

will dominate and the entries will be more and more sparsely distributed.

One approach to this problem is to use one of the standard techniques for representing sparse matrices in a more compact form (Knuth,1968). One of these techniques is to represent the matrix as a linked list structure. If the matrix for a tetrahedron is expressed in this form then its structure would be:



Volume Matrix as a Linked List Data Structure

Figure 1.

In this case a 4x4 matrix with 12 entries is replaced by a linked list structure which requires at least 36 storage places. If element (a) is used then accessing edge A/B involves following the links along row A until column B is found and vice versa, following the links along row B until column A is found. If element (b) is used then line pairs are expressed explicitly in the data structure, but for this simple example it raises the storage space required to at least 48 words of memory. This approach has effectively lost the original advantage of direct access by calculating addresses from the indexes.

One advantage of this form of representation is that the distinction

between two separate matrices and a single composite matrix becomes less important. If the two volume matrices A and B are mixed up in one matrix the result would be

A	2	4	3
3	B	2	1
2	1	C	4
4	3	1	D

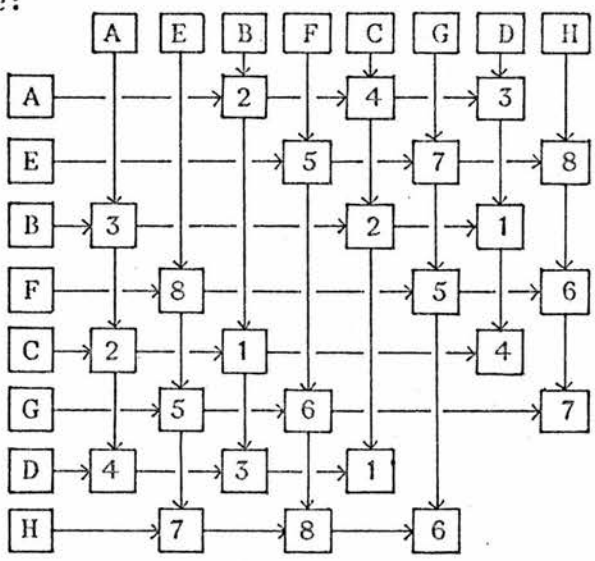
E	5	7	8
8	F	5	6
5	6	G	7
7	8	5	H

A		2		4		3	
	E		5		7		8
3		B		2		1	
	8		F		5		6
2		1		C		4	
	5		6		G		7
4		3		1		D	
	7		8		6		H

Two Volume Matrices Mixed in One Matrix

Figure 2.

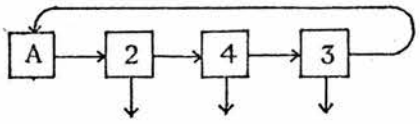
If the second matrix form is restructured as a linked-list matrix the result will be:



Two Volume Matrices Mixed as a Linked List Structure

Figure 3.

It can be seen that this arrangement is in reality no different from the linked-list structure for the two separate matrices. If the rows and column pointer links are closed to form loops or ring structures as shown in Figure 4.



Column or Row as a Ring Structure

Figure 4.

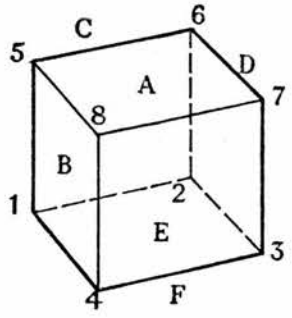
then it becomes relatively easy to access all the facets making up a particular volume surface; however, they are distributed throughout a storage space. It can be seen that in the case of the union operation where internal facets must be removed, this can be carried out using this structure relatively easily. The difficulties with this data structure occur where it is necessary to relocate any section of the data in any way other than as a single block. If pointers are expressed as relative addresses then base-displacement accessing will permit the whole linked structure to be moved. If any subsection of the block needs to be moved, pointers are complicated entities to update even if two way linked lists are used. For this reason, alternative methods for working with these matrix forms were sought.

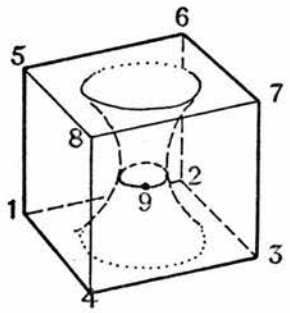
In representing the matrix as a linked list structure the empty cells in the matrix are discarded. It was from a study of the relevance of these blank spaces that an alternative way of compacting the matrix structure in storage was found. When two facets do not have an edge in common, then a blank space is created in the volume matrix. One approach to removing blank spaces could be to give the same name to collections of facets

which are not in contact. Finding the smallest number of names which are necessary to classify facets so that they can be distinguished from their neighbours is usually referred to as the graph colouring problem.

By using the blank spaces in a graph matrix it is possible to define an algorithm which can give a greatly reduced number of facets names, and incidentally can be used to remove the blank spaces. This procedure has not been fully investigated so the idea is illustrated by a series of examples. If the matrix for a cube is considered, it can be seen that there are three pairs of blank spaces not counting the main diagonal. These correspond to pairs of planes which do not have a common dividing edge between them. In the case of a cube these blanks correspond to the three pairs of opposite facets.

A	5	6	7	8	*
8	B	5	*	4	1
5	1	C	6	*	2
6	*	2	D	7	3
7	8	*	3	E	4
*	4	1	2	3	F





A	5	6	7	8	9
8	B	5	10	4	1
5	1	C	6	11	2
6	10	2	D	7	3
7	8	11	3	E	4
9	4	1	2	3	F

Blank Spaces in the Volume Matrix of a Cube

Figure 5.

The argument is based on the idea that if the blanks are filled, then

all the facets must have different colours since each one is in contact with all other facets. The interpretation of what this means is shown in the diagram above. The pairs of new points correspond to loop edges between these facets starting and finishing with the same point. If each of these dividing edges is removed, so making the separate, differently coloured facets - the same facet then the original matrix can be reduced in the following way:

A	5	6	7	8	Ø
8	B	5	Ø	4	1
5	1	C	6	Ø	2
6	Ø	2	D	7	3
7	8	Ø	3	E	4
Ø	4	1	2	3	F

6 x 6

Making B and D the same colour:

A	5+7	6	8	Ø
8+6	B+D	5+2	4+7	1+3
5	1+6	C	Ø	2
7	8+3	Ø	E	4
Ø	4+2	1	3	F

5 x 5

Making A and F the same colour:

A+F	5+7 + 4+2	6+1	8+3
8+6 + 1+3	B+D	5+2	4+7
5+2	1+6	C	Ø
7+4	8+3	Ø	E

4 x 4

Making E and C the same colour:

A+F	5+7+4+2	6+1+8+3
8+6+1+3	B+D	5+2+4+7
5+2+7+4	1+6+8+3	C+E

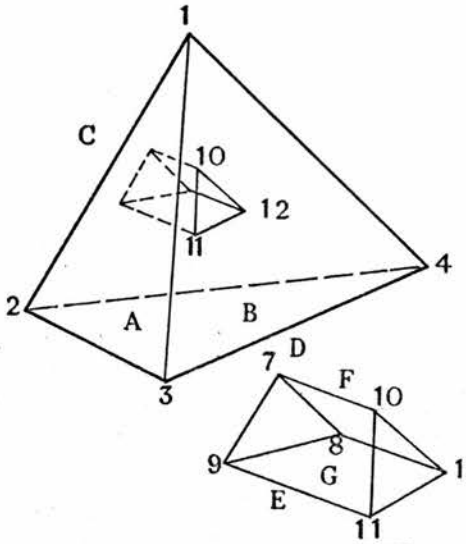
3 x 3

Reducing the Volume Matrix for a Cube

Figure 6

VOLUME MATRIX STORAGE

The final result is a 3x3 matrix which corresponds with the fact that only three colours are needed to colour the surface facets of a cube in a way that no adjacent facets have the same colour.



A	1	1	1	Ø	Ø	Ø
1	B	1	1	1	1	1
1	1	C	1	1	1	1
1	1	1	D	Ø	Ø	Ø
Ø	1	1	Ø	E	1	1
Ø	1	1	Ø	1	F	1
Ø	1	1	Ø	1	1	G

7 x 7

Ø	1	1	1	1	1
1	Ø	1	1	1	1
1	1	Ø	1	1	1
1	1	1	Ø	Ø	Ø
1	1	1	Ø	Ø	1
1	1	1	Ø	1	Ø

6 x 6

Ø	1	1	1	1
1	Ø	1	1	1
1	1	Ø	1	1
1	1	1	Ø	1
1	1	1	1	Ø

5 x 5

Using a Boolean Matrix in Facet Reduction

Figure 7

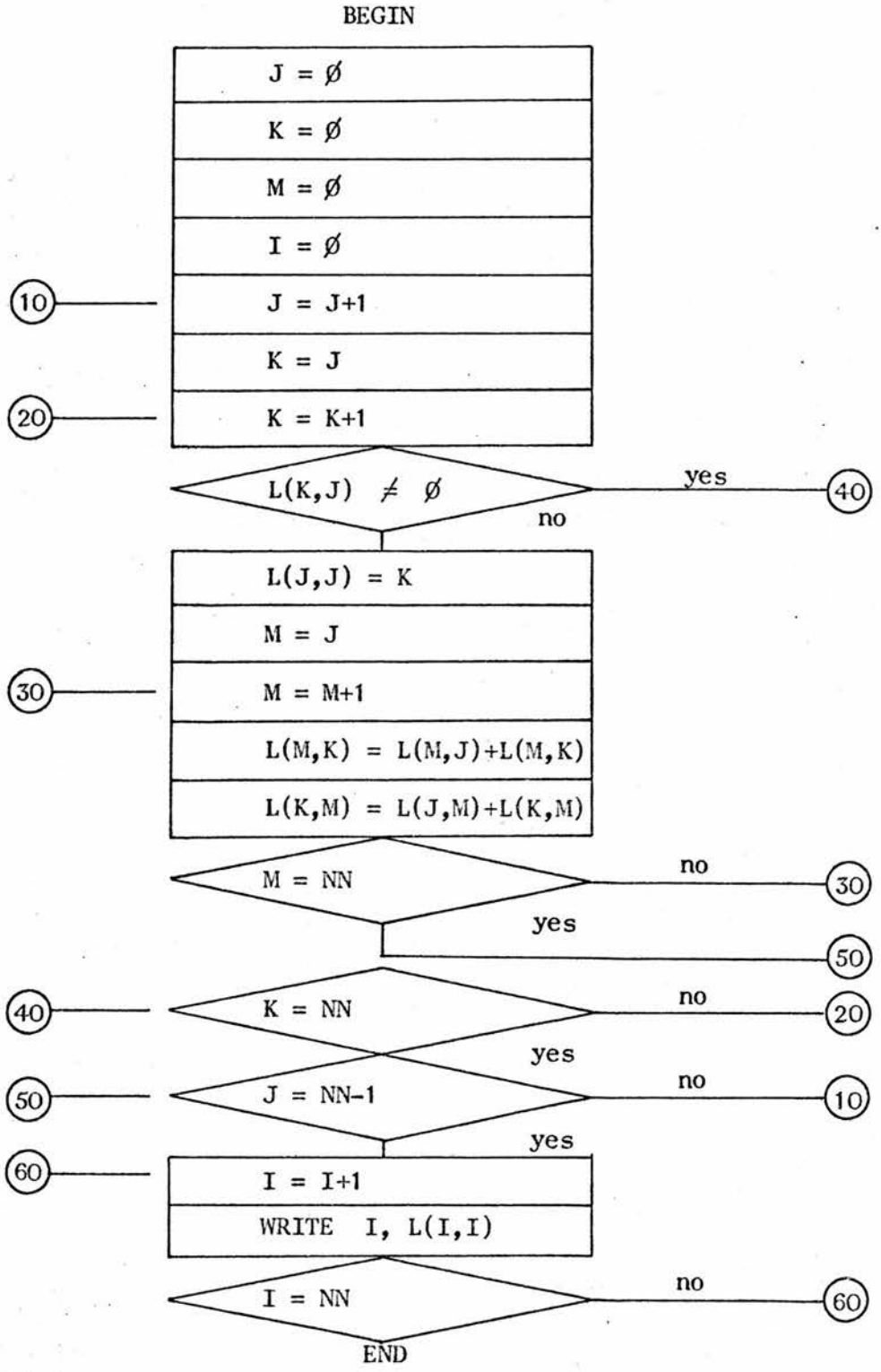
This procedure can be carried out using a boolean matrix where merely the presence or absence of a link is shown. If the same operation is carried out for the volume shown in Figure 7 the result is a five by five matrix showing every face linked to every face. This situation consequently requires five colours. What is not clear is whether this is the minimum number required. It is also not immediately apparent whether the order in which facets are paired off will affect the final

number of colours. It has been postulated that only four colours are necessary to colour a planar graph, though it has only been possible to prove that five colours will be sufficient.

Originally this matrix form was thought to represent only planar graphs. This was a mistake which arose from considering the planar graph to be a network on any surface. However, on looking into the definition of the planar graph with more care, one of its properties was found to be that it could be projected onto the surface of a sphere (Busacker and Saaty, 1965). The volume subtraction process which has already been described can result in a volume with a hole through it; the surface of this volume though it can be represented as a matrix cannot be mapped onto the surface of a sphere. Consequently, this matrix form, though it can represent a network on the surface of a volume, must include a wider class of graphs than that defined by planar graphs. In spite of this, the dual nature of the area/point and point/area matrices seems to be operationally valid even where the rules are stretched to include closed loops.

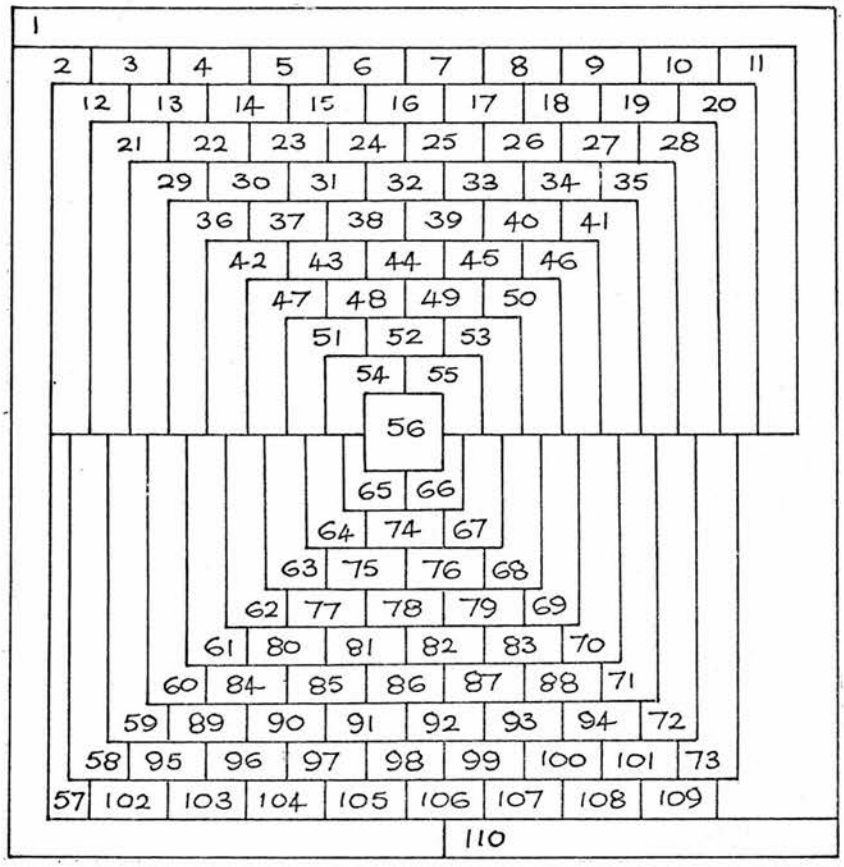
A simple routine was written to test out this matrix reduction procedure. The input data is required as an adjacency matrix where the rows and columns correspond to facets and where two adjacent facets are indicated by 1, and non-adjacency along an edge is indicated by Ø. The flow diagram for this algorithm is given below, where L(NN,NN) is the input data matrix and NN is the number of facets. The output is a linked list showing which facets have been grouped under the same colour, consequently the number of Øs, indicating the end of lists, gives the

number of colours required.



Matrix Reduction Algorithm

Figure 8



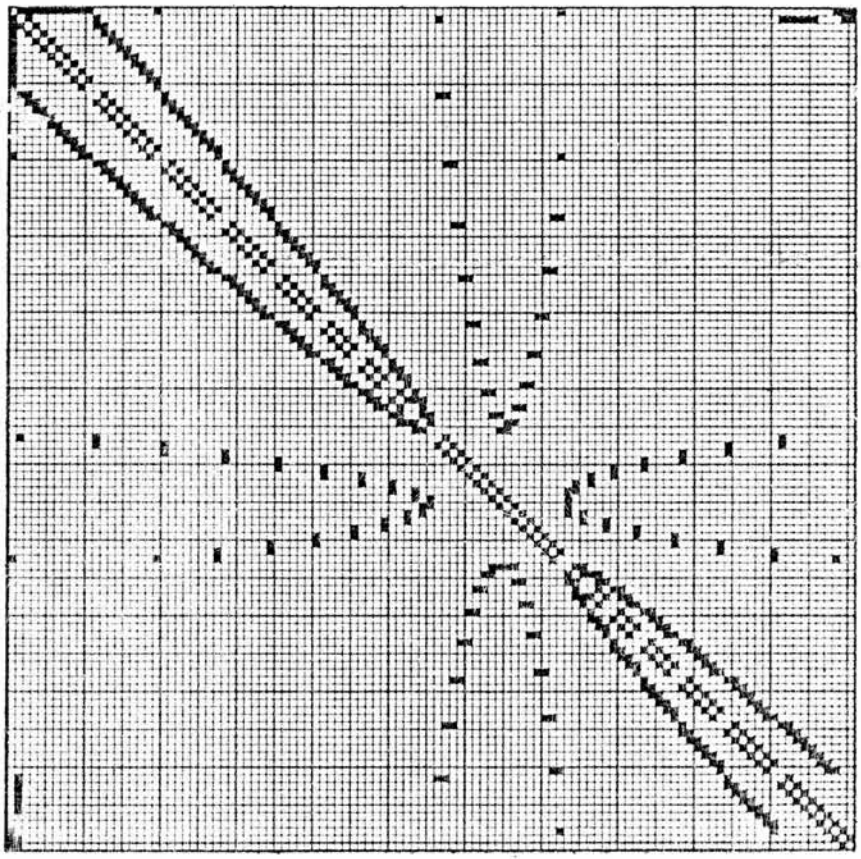
Test Data For the Matrix Reduction Algorithm[1]

Figure 9

The algorithm given in Figure 8 , was tested on the graph given in Figure 9. This graph was published in the Scientific American Magazine in this uncoloured form in an April 1st article, where it claimed that it was the first planar graph found which needed more than four colours! It was complex enough that first attempts to colour it by hand were unable to disprove this claim. It consequently seemed an ideal example to test the previous procedure. An adjacency matrix was set up for the 111 zones, Figure 10, and this was processed to give the output list given in Figure 10. The result indicated the need for five colours. All that was then necessary was to find a four colour solution to this graph, and it would be clear that the order in which blank spaces were removed would affect the number of colours needed.

1. Scientific American Magazine, April 1975.

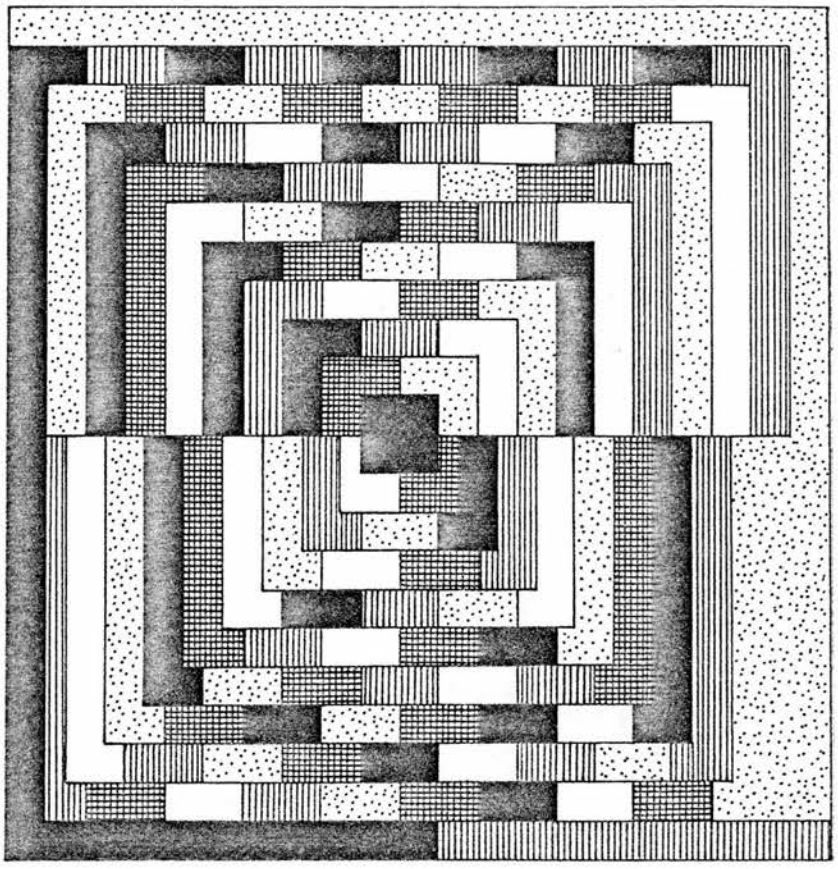
VOLUME MATRIX STORAGE



1	12	21	22	41	45	61	66	81	87	101	105
2	4	22	25	42	46	62	65	82	85	102	106
3	5	23	26	43	49	63	70	83	90	103	108
4	6	24	27	44	50	64	68	84	91	104	110
5	7	25	31	45	48	65	69	85	89	105	0
6	8	26	32	46	51	66	71	86	88	106	109
7	9	27	30	47	52	67	72	87	94	107	0
8	10	28	33	48	53	68	73	88	95	108	111
9	11	29	34	49	54	69	75	89	92	109	0
10	21	30	38	50	55	70	74	90	93	110	0
11	22	31	35	51	56	71	76	91	96	111	0
12	14	32	36	52	57	72	77	92	97		
13	15	33	37	53	58	73	78	93	98		
14	16	34	39	54	61	74	79	94	99		
15	17	35	40	55	59	75	81	95	100		
16	18	36	41	66	60	76	82	96	101		
17	19	37	44	57	64	77	83	97	102		
18	28	38	42	58	62	78	80	98	107		
19	29	39	43	59	63	79	84	99	103		
20	23	40	47	60	67	80	86	100	104		

Input Matrix and Output List

Figure 10.



Computer Solution to the Colouring Problem

Figure 11

The coloured version of the graph prepared from the output in Figure 10 is given above. The areas which are grouped together by the same colour are:

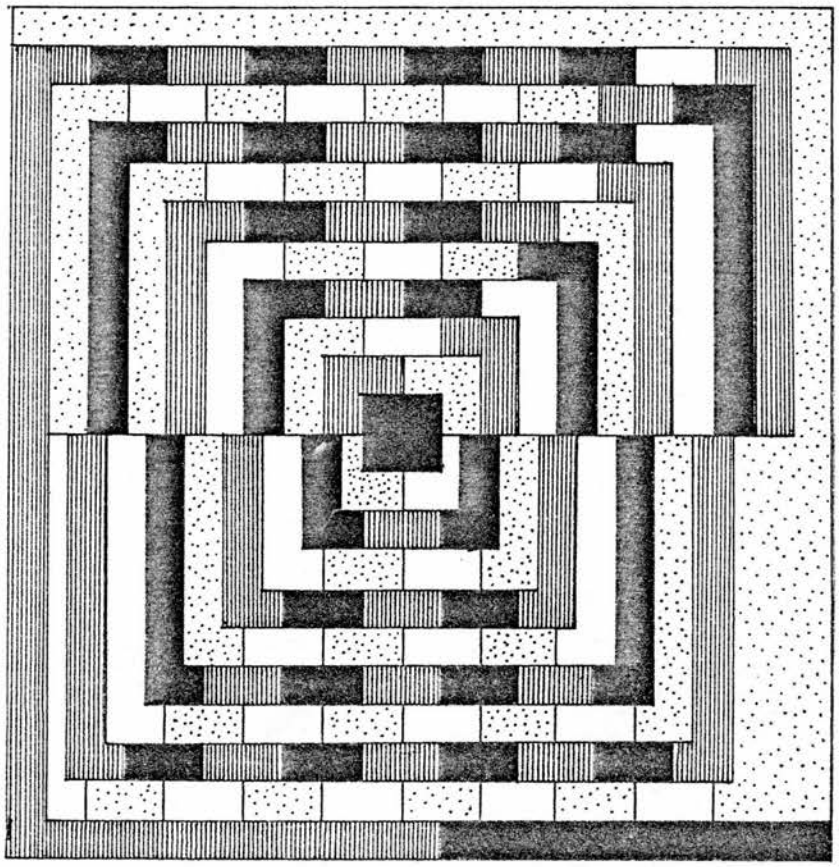
Colour 1: 1, 12, 14, 16, 18, 28, 33, 37, 44, 50, 55, 59, 63, 70, 74, 79, 84, 91, 96, 101, 105.

Colour 2: 2, 4, 6, 8, 10, 21, 24, 27, 30, 38, 42, 46, 51, 56, 60, 67, 72, 77, 83, 90, 93, 98, 107.

Colour 3: 3, 5, 7, 9, 11, 22, 25, 31, 35, 40, 47, 52, 57, 64, 68, 73, 78, 80, 88, 95, 100, 104, 110.

Colour 4: 13, 15, 17, 19, 29, 34, 39, 43, 49, 54, 61, 66, 71, 76, 82, 85, 89, 92, 97, 102, 106, 109.

Colour 5: 20, 23, 26, 32, 36, 41, 45, 48, 53, 58, 62, 65, 69, 75, 81, 87, 94, 99, 103, 108, 111.



Four Colour Solution to the Colouring Problem

Figure 12

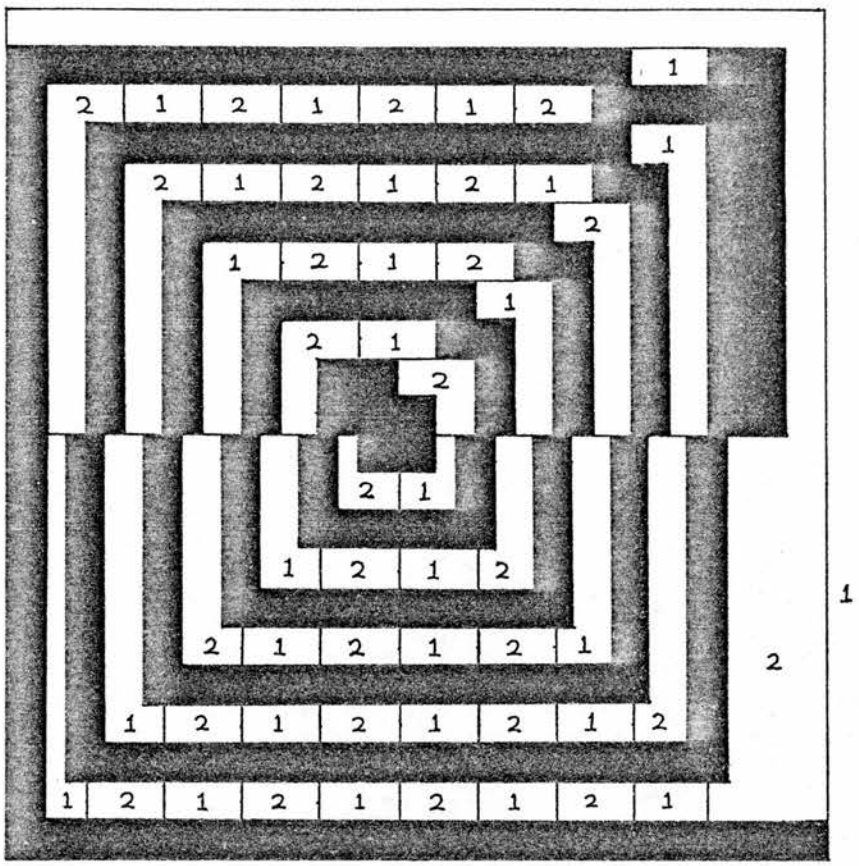
In the four coloured solution to the graph given in Figure12the zones were grouped together in the following way:

Colour 1: 54, 74, 53, 48, 62, 78, 69, 40, 38, 36, 84, 86, 88, 35, 26, 24, 22, 58, 96, 98, 100, 73, 11, 19, 8, 6, 4, 2.

Colour 2: 56, 64, 67, 49, 47, 77, 79, 46, 39, 37, 60, 85, 87, 71, 27, 25, 23, 21, 95, 97, 99, 101, 20, 9, 7, 5, 3, 110.

Colour 3: 66, 52, 63, 76, 50, 44, 42, 80, 82, 70, 34, 32, 30, 59, 90, 92, 94, 28, 17, 15, 13, 57, 103, 105, 107, 109, 111.

Colour 4: 65, 55, 51, 75, 68, 45, 43, 61, 81, 83, 41, 33, 31, 29, 89, 91, 93, 72, 18, 16, 14, 12, 102, 104, 106, 108, 1.



Partitioning Zones for a Four Colour Solution

Figure 13

Figure 13 illustrates the strategy adopted in finding the four-colour solution. The idea was to partition the zones into two sets so that a vertex which occurred at the junction of three zones could never have these three zones in one set. The simplest way of implementing this strategy was to start with an arbitrary zone and then form a spiral round it. Where awkward local situations occurred they could be solved by converting the two arms of the spiral into a pair of interlocking trees. Once the two sets have been formed, then it is a simple matter to distinguish neighbouring zones in one set by alternating the colour from zone to zone. This limits the use of closed loops to those which have an even number of zones in a ring.

VOLUME MATRIX STORAGE

The use made of this four-colour solution was to create a collection of four by four matrices where the zone colours took the place of facet names. This resulted in the 12321 cell storage space requirement for the previous 111 x 111 matrix being reduced to 880 cells for 55 4 x 4 matrices (Figure 14). It seemed that there was still enough information within these packed matrices to perform many of the operations proposed in the Section above. However, though worth future investigation, this line of enquiry was taken no further.

A	1	14	7
2	B	4	1
5	3	C	13
1	4	3	D

A	6	10	2
17	B	13	3
9	14	C	4
10	2	5	D

A	8	24	5
7	B	15	6
11	28	C	22
6	7	9	D

A	12	16	9
27	B	19	8
15	18	C	10
8	19	11	D

A	15	22	11
14	B	35	12
23	20	C	16
12	13	31	D

A	38	26	17
21	B	23	18
25	24	C	33
16	17	18	D

A	23	30	21
40	B	25	20
31	42	C	19
22	21	20	D

A	25	34	27
24	B	29	28
35	30	C	26
26	27	45	D

Four Colours used to Compact a Graph Matrix

Figure 14

The practical approach for the stage reached by the program system was to use the coordinate searching and sorting procedures presented briefly in Chapter 7. If line segments are stored in sets, each line segment represented by the four-entry coordinate A/B, 1/2, then the same length coordinate can be used for storing the point coordinates and plane facet coefficients. The tests on point and plane equations create a set of values which can then be used to select the lines which are affected

and need to be changed in an operation. These can then be called into storage and, using the coordinate sorting operation, either transformed into a fixed matrix structure or left as a pointer-structure which uses sparse matrix accessing procedures. The matrix bookkeeping operations described in the previous Chapter can then be applied in a direct way - creating new lines and new facets where appropriate. In the next Chapter linking the operations which use the geometrical definition of points and planes with their names used in a linkage structure, is discussed more fully to show how the volume matrices may be constructed in the first place. The work presented in this Chapter must be the subject of further investigation when more is known of the demands which may be made on such basic storage procedures. It is possible to link the basic procedures described above to work done on associative memory systems - both in hardware and in simulated software systems, and this provides an alternative environment in which to consider these storage requirements.

CHAPTER 11

VOLUME MATRIX CONSTRUCTION

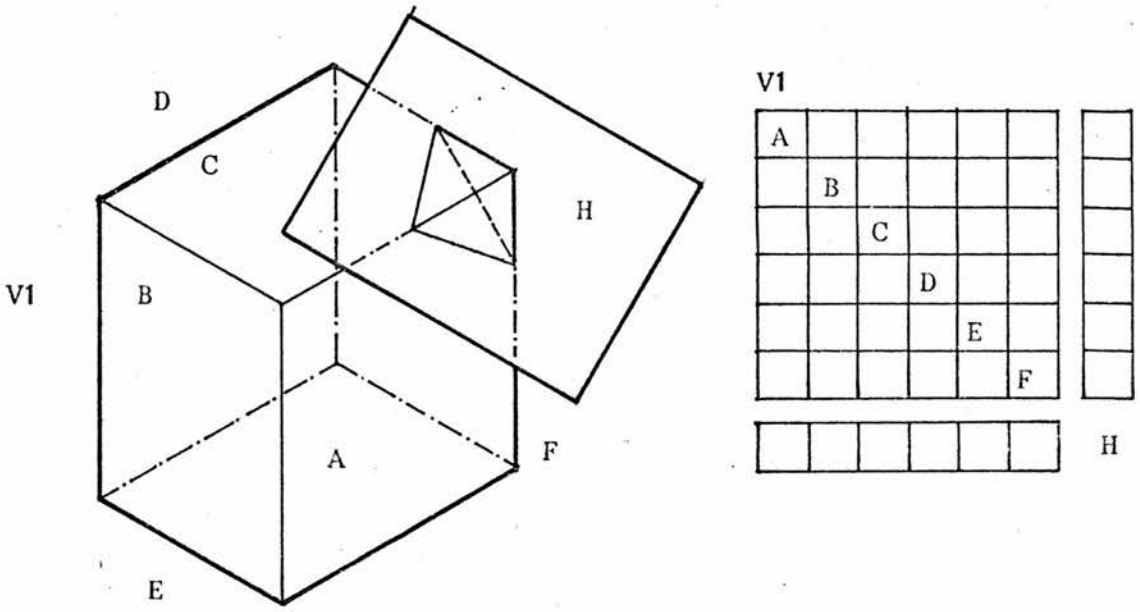
VOLUME MATRIX CONSTRUCTION

After investigating the structure and behaviour of these matrices for a variety of volume operations, the next problem was to find a reasonably simple way of building them. This raises another problem which has been masked in the discussion up to this point by the use of point names instead of coordinates. Where plane surfaces are assumed and where facets have more than three edges it is difficult to obtain by input procedures such as digitising, coordinate values to match the named vertices but which also lie in a plane. The problem is compounded by the fact that a particular vertex must lie in three or more planes simultaneously. This is a practical data preparation problem which has been investigated by Appel, and which must have been solved in one way or another by each of the research workers in this field.

The approach which was adopted in this work is based on the use of a cutting plane, (Wehrli, Smith, and Smith, 1970). By starting with a cube which has its facets parallel to the axes planes, it is possible to define the coordinates of its vertices accurately in a simple way. If this cube is large enough to contain the required volume, then if the facets of this volume are used to cut off external portions of the original cube then the final convex volume which is required will be left. What this means in practice is that data input is in the form of plane coordinates rather than point coordinates. The way this can be carried out as a matrix operation can be illustrated using the diagram in Figure 1. For each of the coordinates in the point list calculate the value of 'k' using the coefficients of the plane equation of H:

$$k = a.x + b.y + c.z + d$$

VOLUME MATRIX CONSTRUCTION



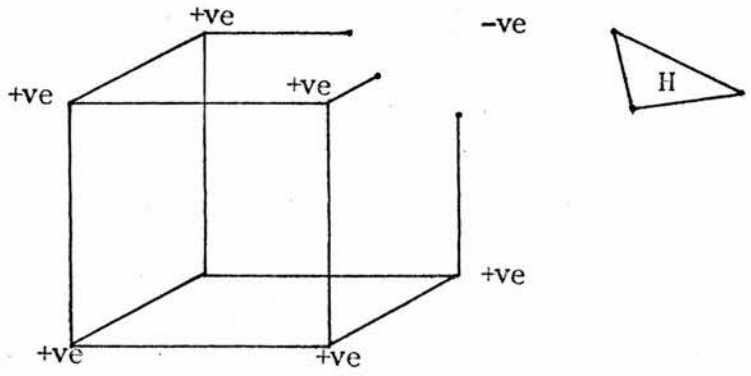
Point Coordinate Array

P1	P2	P3	P4	P5	P6	P7	P8

The Cutting Plane Principle

Figure 1.

If k is positive, then the point will lie on one side of the plane, if it is negative then the point will lie on the other side of the plane, and if the value of k is zero the point will lie in the plane. If each line segment defined by V(I,J),V(J,I) in the volume matrix is examined then only those line segments which pierce the plane, so that the two values of k are of opposite sign need to be redefined. This can be



Testing the vertices of the Universe Cube against a Cutting Plane

Figure 2

done by calculating the intersection point of the plane with the line and substituting this new point for the end point which lies outside the cutting plane. Diagrammatically, this procedure achieves the result shown in Figure 2.

The matrix for the original volume is incomplete until the boundary of the new facet H is defined. This can be done using a matrix book-keeping procedure in the following way: for each line segment examine the relationship between k1 and k2 for the end points p1 and p2; these can be classified by the diagram in Figure 3.

P2 \ P1	+ve	Ø	-ve
+ve	I	IV	V
Ø	IV	II	VI
-ve	V	VI	III

End Point Conditions
Figure 3.

These relationships require the following strategy:

I. Leave both points as they are.

II. See below.

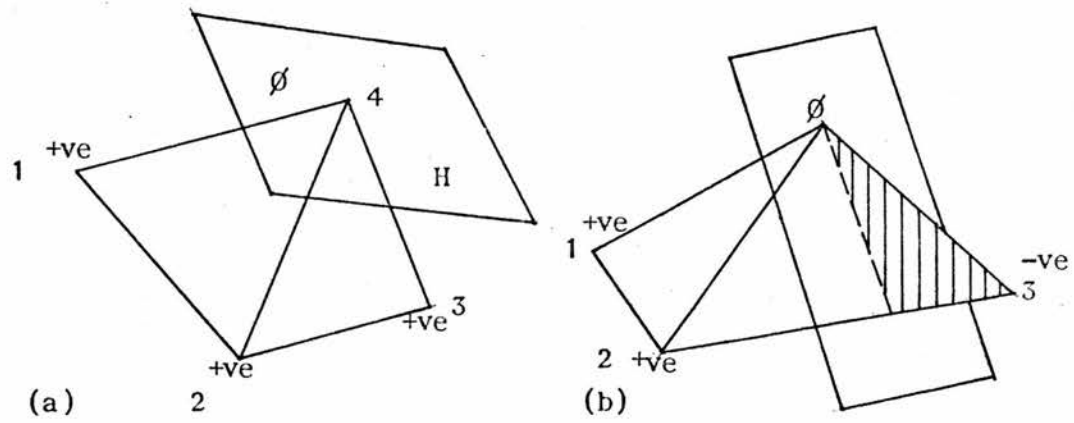
III. Remove both points.

IV. See below.

V. Calculate new point, substitute for outside point, include the new point in Matrix row and column H.

VI. Remove both points.

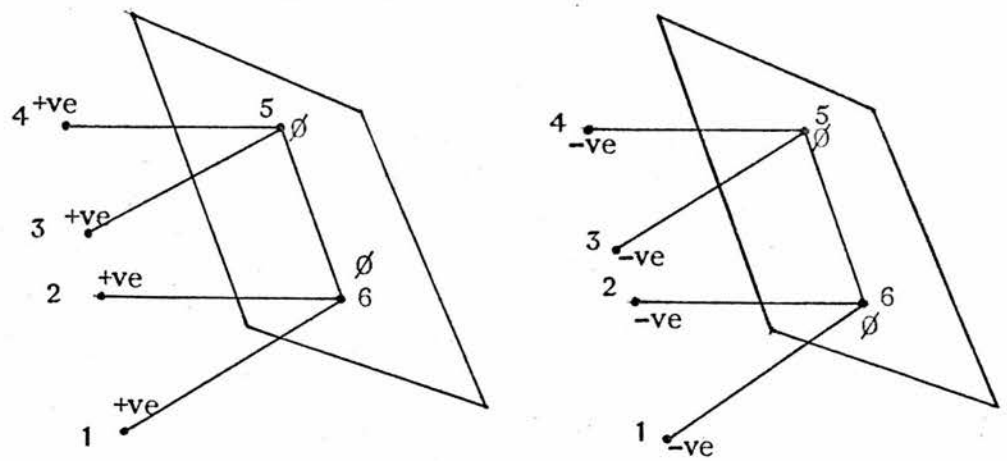
Both of the conditions II and IV raise problems associated with the edge and point contacts already discussed. Consider the situations shown in Figure 4.



Cutting Plane through Vertices

Figure 4

Neither in case (a) nor in case (b) need the lines +/O or O/+ be modified in the matrix. However, in case (a) point 4 will not be included in H, whereas in (b) this point will have to be included. In the case of condition II the situation is even more complex and is illustrated in Figure 5.



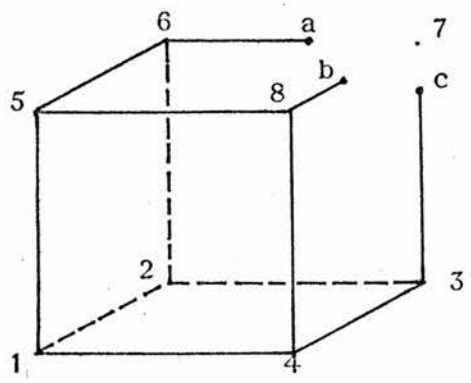
Cutting Plane through Edges

Figure 5.

In the case of line 5/6 in case (c) it can be left as it is; in case (d) it must be removed. It can be seen that in the case of line segments which give k relationships of the form +/O, O/O and O/+, their treatment cannot be established in isolation, as the other conditions can, but

must be collected together and resolved for the whole new volume description.

In the simple case considered above the result of applying these operations will be:



A	5	6	7	8		
8	B	5		4	1	
5	1	C	6		2	
6		2	D	7	3	
7	8		3	E	4	
	4	1	2	3	F	
						H

1	2	3	4	5	6	7	8
+	+	+	+	+	+	-	+

Matrix Definition of a New Facet

Figure 6

Generating the new points, and substituting for negative point 7:

$V(A,D):7 \longrightarrow a$

$V(E,A):7 \longrightarrow b$

$V(D,E):7 \longrightarrow c$

Subsequent matrix bookkeeping gives the operations

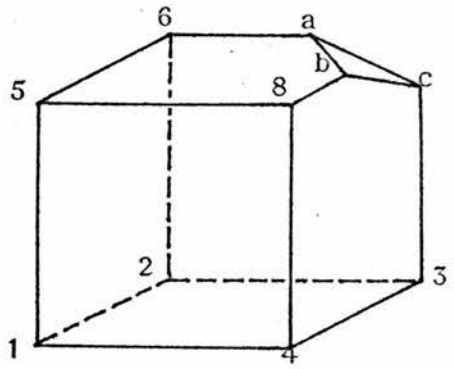
$V(D,H) \longrightarrow a, \qquad V(H,A) \longrightarrow a$

$V(A,H) \longrightarrow b, \qquad V(H,E) \longrightarrow b$

$V(E,H) \longrightarrow c, \qquad V(H,D) \longrightarrow c$

so the final matrix becomes:

VOLUME MATRIX CONSTRUCTION



A	5	6	a	8		b
8	B	5		4	1	
5	1	C	6		2	
6		2	D	c	3	a
b	8		3	E	4	c
	4	1	2	3	F	
a			c	b		H

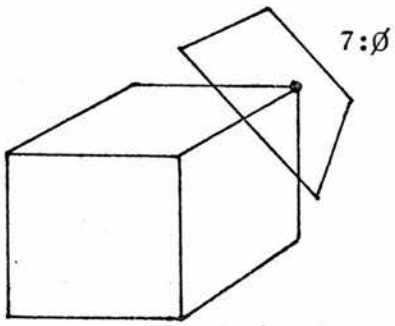
Final Matrix for the New Volume

Figure 7

Where zero values for points appear, as long as the other points are all positive or all negative, then either the matrix stays as it is or the whole volume is outside the cutting plane.

In the case of point contact the two results are shown in Figure 8.

Total volume inside the plane.



1	2	3	4	5	6	7	8
+	+	+	+	+	+	Ø	+

The volume matrix remains unaltered.

Total volume outside the plane.

1	2	3	4	5	6	7	8
-	-	-	-	-	-	Ø	-

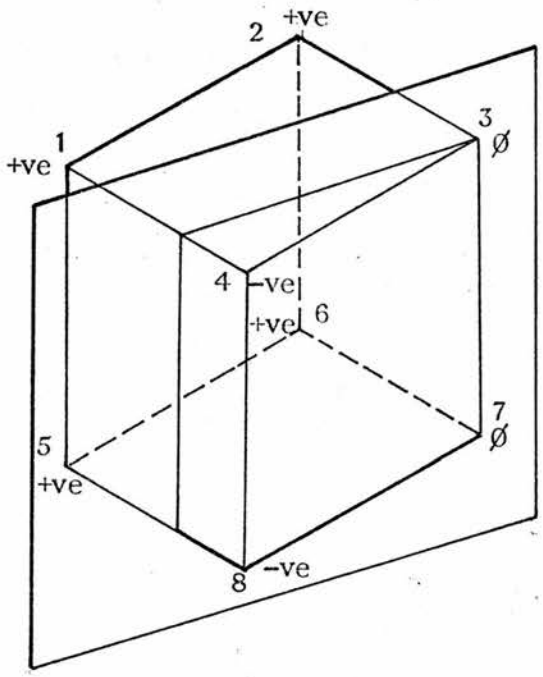
Matrix vanishes

Vertex Contact

Figure 8.

VOLUME MATRIX CONSTRUCTION

The situation is the same in the case of edge contact, so long as all non zero points are the same sign. However, where there are negative and positive values among the non zero points, further action must be taken. Consider the case of a plane cutting an edge shown in Figure 9.



1	2	3	4	5	6	7	8
+	+	Ø	-	+	+	Ø	-

A	5	6	[7]	(8)		
(8)	B	5		(4)	1	
5	1	C	6		2	
6		2	D	[7]	[3]	
[7]	(8)		[3]	E	(4)	
	(4)	1	2	[3]	F	
						H

[Ø]

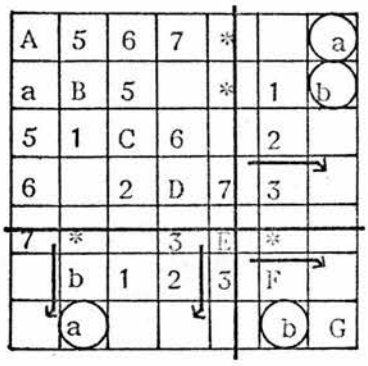
(-ve)

Edge Contact

Figure 9.

VOLUME MATRIX CONSTRUCTION

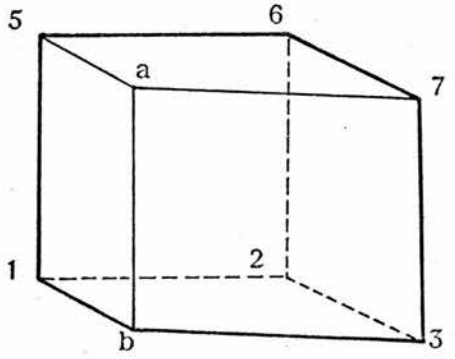
In this matrix the -/- line: E/B, 8/4 can be deleted. The negative points 4 and 8 can be replaced by the new intersection points b and a respectively, in the +/- lines. The -/0 lines and the 0/0 lines can be removed but the zero point must be relocated in H in the appropriate places.



Matrix Operation for Edge Contact

Figure 10

Although a and b can be relocated in H using the same procedure described above, relocating the zero points depends on the removal of the facet E, since all its vertices other than the zero points were negative. The final result consequently is shown in Figure 11.



A	5	6	7		a
a	B	5		1	b
5	1	C	6	2	
6		2	D	3	7
	b	1	2	F	3
7	a		3	b	H

New Volume with an Existing Edge in the New Facet

Figure 11.

VOLUME MATRIX CONSTRUCTION

To be able to remove the external facets such as E, the running total of k values for each area row or column must be kept. If this is done after the negative values have been deleted then any total which is equal to zero will indicate a facet which must also be deleted. The exact way that this process is carried out depends on the way the matrix is represented in storage but, however it is done, it is reasonably simple to implement.

This demonstrates the use of one cutting plane. By repeating this for each facet of a convex volume, the volume matrix for the volume can be automatically set up. It can be seen that, by its nature, this process cannot be used to produce a volume with re-entrant surfaces. Such a volume can be constructed as a volume matrix by adding together the volume matrices of simpler convex volumes.

It was the realisation that the use of the cutting plane could be expressed as the intersection of the space on the positive side of the cutting plane with the original matrix, that made it possible to unify the creation of convex and re-entrant surfaced volumes by means of boolean expressions. If the planes A, B, C, D, E, F and G are cutting planes, the convex volume can be defined as:

A.B.C.D.E.F.G

The addition of two such volumes can be expressed as:

(A.B.C.D.E.F.G) + (H.I.J.K.L.M)

The significant point is that this expression can be manipulated in its own right, using the rules of boolean algebra.

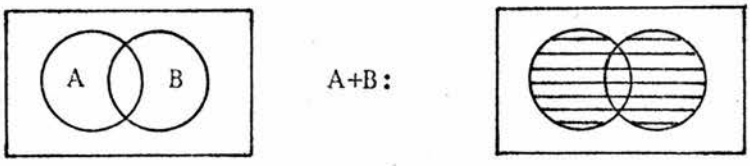
The Use of Boolean Expressions to Define Volumes

It is significant that the three boolean operators: Union "+",

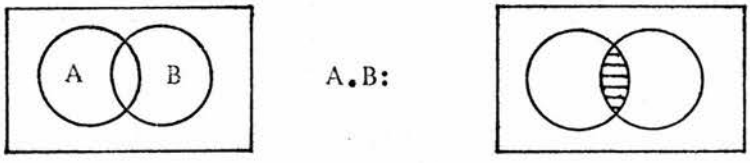
VOLUME MATRIX CONSTRUCTION

Intersection ".", and Complement " ' ", are often illustrated by the use of Venn diagrams shown in Figure 12.

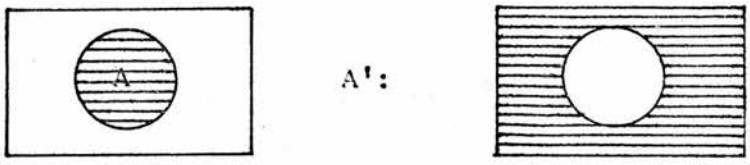
Union



Intersection



Complement



Venn Diagrams

Figure 12.

Each of these operations corresponds to one of the matrix operations which have already been discussed. However, where these operations have to be performed in sequences it is possible to restructure these sequences either for convenience in processing or to reduce the total number of calculations which have to be performed, by applying the rules of boolean algebra which can be summarised in the following list of relationships:

Commutative Law: $A+B = B+A$

$A.B = B.A$

Distributive Law: $A.(B+C) = (A.B)+(A.C)$

$A+(B.C) = (A+B).(A+C)$

VOLUME MATRIX CONSTRUCTION

Identity Elements: $A+\emptyset = A$

$A.I = A$

Complementary Element: $A+A' = I$

$A.A' = \emptyset$

Associative Laws: $(A+B)+C = A+(B+C)$

$(A.B).C = A.(B.C)$

From these relationships a series of theorems provides the additional results:

$$A+I = I$$

$$A.\emptyset = \emptyset$$

$$A+A = A$$

$$A.A = A$$

$$(A.B)+A = A$$

$$(A+B).A = A$$

$$(A')' = A$$

$$(\emptyset')' = \emptyset$$

$$(\emptyset)' = I$$

$$(I)' = \emptyset$$

$$A+(A'.B) = A+B$$

$$A.(A'+B) = A.B$$

$$(A+B)' = A'.B'$$

$$(A.B)' = A'+B'$$

For example, it is possible to remove all the occurrences of the union operator from the following expression:

$$A+(B.C) \longrightarrow (A+B).(A+C)$$

$$\longrightarrow (A'.B')'.(A'.C')'$$

VOLUME MATRIX CONSTRUCTION

If it is necessary to construct the volume defined by the expression: A.B.C.D.(F+G+H+I) using cutting planes and the merging of convex volumes then the volume definition can be rewritten as:

$$A.B.C.D.F + A.B.C.D.G + A.B.C.D.H + A.B.C.D.I$$

which requires convex volume matrices to be generated and then added together. An alternative approach is to rewrite the expression in the form:

$$A.B.C.D.(F'.G'.H'.I')'$$

This is the same as the volume subtraction process so this can be rewritten:

$$(A.B.C.D) - (F'.G'.H'.I')$$

This expression requires two matrices to be generated, using the cutting plane process; and the final volume to be created by subtracting these two matrices from each other. The first stage in investigating the possibilities of using this approach was to build a language processing program which would accept boolean expressions and restructure them in some appropriate way.

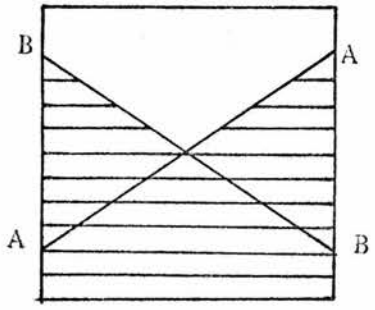
Boolean Expression Analyser

Before considering some of the more complex possibilities it was necessary to build a simple system which could handle boolean expressions relating plane equations. The interpretation of the boolean operators acting on planes can be shown in the sequence of diagrams shown in Figure 13.

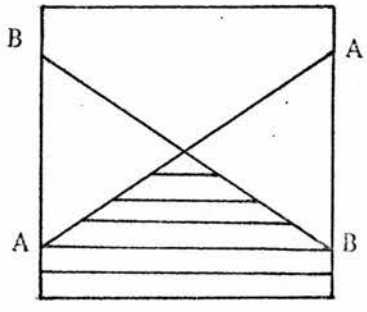
Given the coefficients of a plane equation a, b, c, d and any point x, y, z, then the plane is defined by the set of points which give:

$$0 = a.x + b.y + c.z + d$$

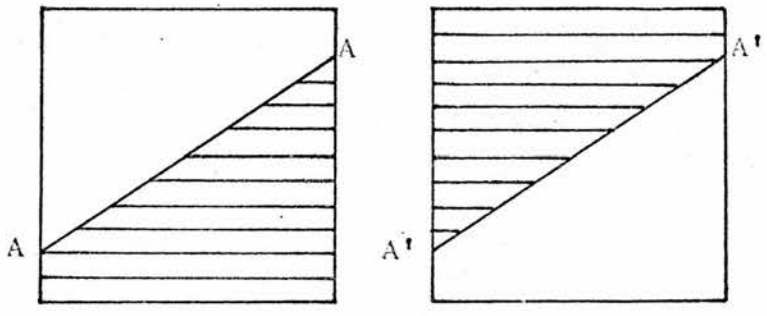
Union A+B



Intersection A.B



Complement A'



Boolean Interpretation of Planes

Figure 13.

The remaining points will give positive or negative values when their coordinates are substituted into this equation, depending on whether they are on one side of the plane or the other. Consequently, the space on one side of a plane can be defined by the inequality:

$$\emptyset \geqslant a.x + b.y + c.z + d$$

and this would lead to the conclusion that the complement space must be defined by the inequality:

$$\emptyset < a.x + b.y + c.z + d$$

This approach, however, raises practical difficulties when an attempt is made to implement it. If the whole space is defined by:

$$k = a.x + b.y + c.z + d$$

where $-00 \leqslant k \leqslant +00$

then a space A can be defined as the set of points which gives $k \geqslant \emptyset$ and conversely $A^{\prime}$ where $k \leqslant \emptyset$. This makes it possible to complement a plane represented by the coefficients (a,b,c,d) simply by multiplying each of these coefficients by -1 to give (-a,-b,-c,-d). It can be seen that the consequence of this is that the set of points which lie in the plane belongs to A and also to $A^{\prime}$, so that $A.A^{\prime} \neq \emptyset$. This is a point which will be returned to at a later stage, but it is sufficient to note that this problem has already appeared in an implicit form in the treatment of tangent planes, edge contacts and point contacts. By establishing the values of k: (0,+) and (0,-) as the two logically complementary properties of a plane, it becomes possible to define a cutting plane A and then to create its complement $A^{\prime}$ in a simple way.

At this preliminary stage in the investigation only two language statements were considered. Firstly, an expression which would relate a name to a plane equation or rather the coefficients of the plane equation was needed and secondly, an expression which assigns a name to a boolean expression using these plane names. In defining the plane descriptions there was a variety of ways in which information could be provided to the computer system to set up the correct plane equation coefficients. The method chosen was to input three point coordinates and then from these automatically calculate the coefficients. In some ways this seems to be defeating the purpose of using plane equations in the first place. It

was done in this way, at this stage in the work, because most of the data used for testing purposes had to be prepared graphically and it was easier to extract point coordinates from the drawings than plane equations. The form of this language statement is defined by the grammar rules:

<Plane Definition> ::== <Plane Name> , '=',< Expression> ,';'

<Expression> ::=='(', <Coordinate> ,< Coordinate> ,< Coordinate> ,
<Coordinate> , ')',';'

where Coordinate is a built in phrase. The first three coordinates are used to define the coefficients of the plane, and the fourth co-ordinate is used to define the orientation x of the plane, in other words, the correct signs for the plane coefficients. For input purposes the fourth point is selected to lie 'inside' the plane.

The second language statement was to associate a name with a boolean expression relating plane-names, as the definition of a volume. The form this expression took is defined by the grammar rules:

< Volume Definition> ::==< Volume Name> ,'=',< Expression> ';'

< Expression> ::== <Phrase> ,< Rest of Expression>

< Rest of Expression> ::== '+',< Expression> | < Nul>

< Phrase> ::== <Operand> ,< Rest of Phrase>

<Rest of Phrase> ::== '.',< Phrase> | < Nul>

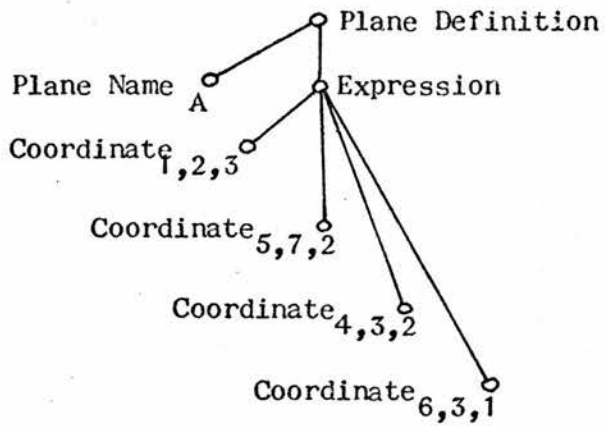
<Operand> ::==<Name> | ''',<Name> | '(', Expression ,')' |
''',<'(',<Expression>,')'

Where <Name> , <Nul> , are built in phrases.

Using these grammar rules in a 'top-down' parser it was possible to convert these two forms of input statement into an analysis record which

could then be used to prepare alternative data structures as volume descriptions, or to generate output.

The statements which these grammar rules allow to be used are illustrated below. The accompanying diagrams show the tree structure of the analysis records created from them.



A = (1,2,3,5,7,2,4,3,2,6,3,1)

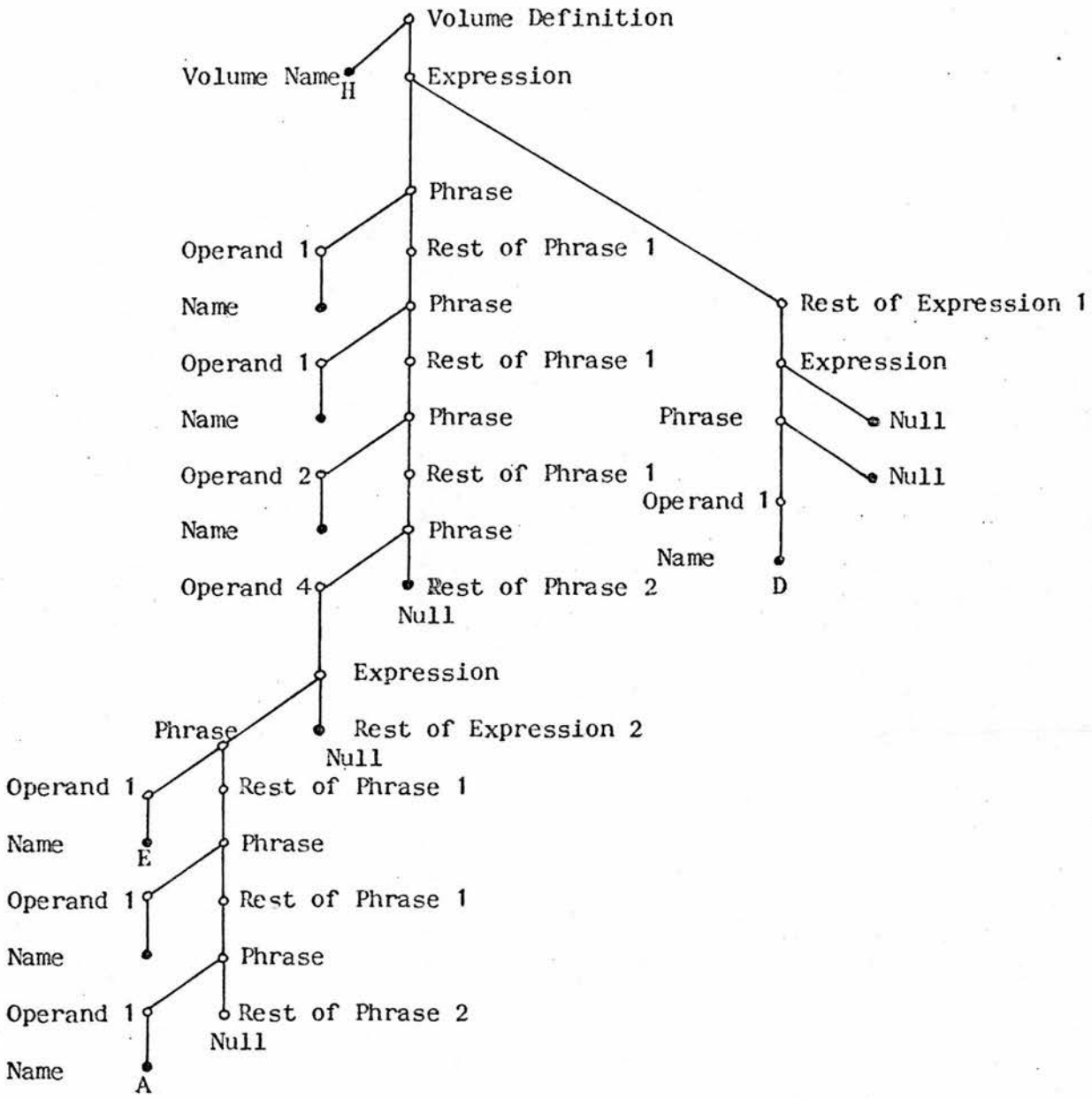
Plane Definition Statement

Figure 14.

Using these two forms of expression, it is possible to define a set of planes and from these planes to define a wide variety of different volume descriptions, built from them. Such a program could take the form:

```
BEGIN:
A = (1,2,3,5,7,2,4,3,2,6,3,1);
B = (7,...                    );
C = (....                     );
D = (....                     );
E = (....                     );
F = (....                     );
G = (....                     );
```

VOLUME MATRIX CONSTRUCTION



H = A.B.'C.'(E.C.'A) + D

Volume Definition Statement

Figure 15.

```
H = A.B.'C.'(E.C.'A) + D;
I = G + E.C.F.A.'(D+'G);
J = ....                ;
END.
```

VOLUME MATRIX CONSTRUCTION

It can be seen that there are several immediate additions which can be profitably made to this language. If the set of plane coefficients (a,b,c,d) are expressed in direction cosine form, then it is possible to define a plane parallel to this plane but a perpendicular distance 'h' from it, by the coefficients (a,b,c,d+h). This makes it possible to define a new operand in these expressions of the form (A+9). If the three axes planes are originally defined as A, B, and C, then all the subsequent planes necessary for defining rectangular structures parallel to these planes can be generated in this way.

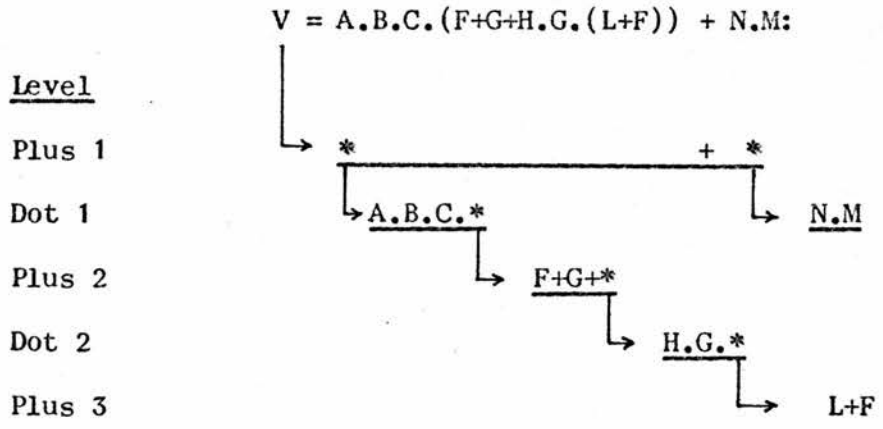
It is only slightly more difficult to implement the plane transformation of rotation, so that A*T can define a plane rotated from position A by T. This has not been implemented in the first stages of this work because it raises general issues in the storage policy for named entities which is considered later. If two planes A and A*T exist in a program should the subsequent transformation of A to A*F require A*T to be transformed to A*T*F? (Comba, 1968). Obviously, it will depend on the situation, but to resolve this problem explicitly in the language structure, will require two classes of transformed plane or volume descriptions. Firstly, those which are independent of the original elements, and secondly, those where the relation to the original element is an important functional aspect of the description. This issue is avoided in the simple system by making it impossible to reuse a plane name for a new plane name, and similarly for volume names.

A further extension which it is also not difficult to implement is to permit volume descriptions to be created from existing volume descriptions. This, again, is a facility which will require the permitted ways in which one description can depend on another to be resolved. Before this could

be attempted, it was necessary to examine the use of these expressions in simple situations.

The semantic stage of analysis for the plane definitions consisted of calculating the plane equation coefficients and setting them up in a table indexed to the plane name in a dictionary. The analysis record created by the parsing routine, though adequate for immediate interpretation in the case of arithmetic operations, was too cumbersome to act as a convenient storage structure for volume definitions. The semantic analysis of the volume definitions consisted of restructuring the form of the analysis record tree, into a new tree structure which was more appropriate for the behaviour of the boolean expressions. During this change the named references to planes were replaced by direct references to the table of plane coefficients.

The new form for volume definitions can be illustrated by the expression in Figure 16.



Tree Structure for Boolean Expressions

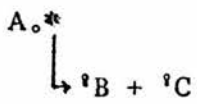
Figure 16.

The advantage of this tree structure is that it is possible to let the level implicitly define whether the operation is + or ., and the operations switch alternately from level to level. In the example considered there were no sub-expressions of the form '(A+B) or '(A.B). However, even where these occur they can be reduced to the same form by the appropriate application of De Morgan's Laws:

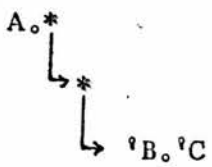
$$'(A.B) \longrightarrow 'A+'B$$

$$'(A+B) \longrightarrow 'A.'B$$

This expansion can be implemented in a very simple way by inserting an extra level in the tree structure and complementing all the elements inside the brackets, so that instead of A.'(B.C) the tree structure becomes:



and A.'(B+C) either becomes A.'B.'C or

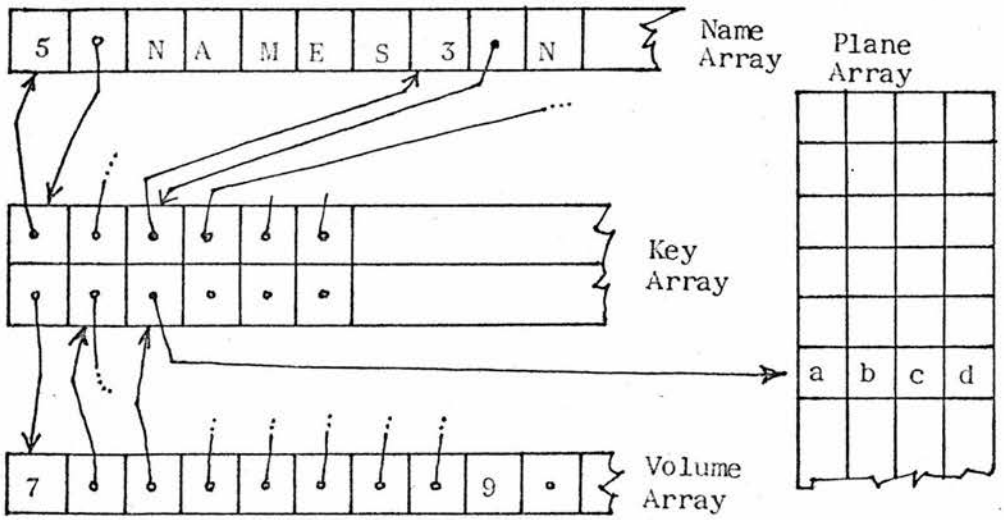


This conversion gives a form for storing the boolean expressions which permits the operators to be removed since they are implied by the organisation of the data.

The end product of this restructuring procedure in the first experimental program was a data structure shown in the diagram in Figure 17.

The arrays for volume descriptions and for names were arranged to handle variable length entries, though at this stage no thought was given

VOLUME MATRIX CONSTRUCTION



Data Structure at the end of Restructing the Boolean Expression

Figure 17

to redefinition or overwriting existing definitions. The plane equation coefficients being of fixed length could be stored in a two dimensional array. The Key array is used as a convenient way of accessing these different entities by indexes. The entry in the Volume array is demonstrated for the expression:

$$V = A.A + B.B + C$$

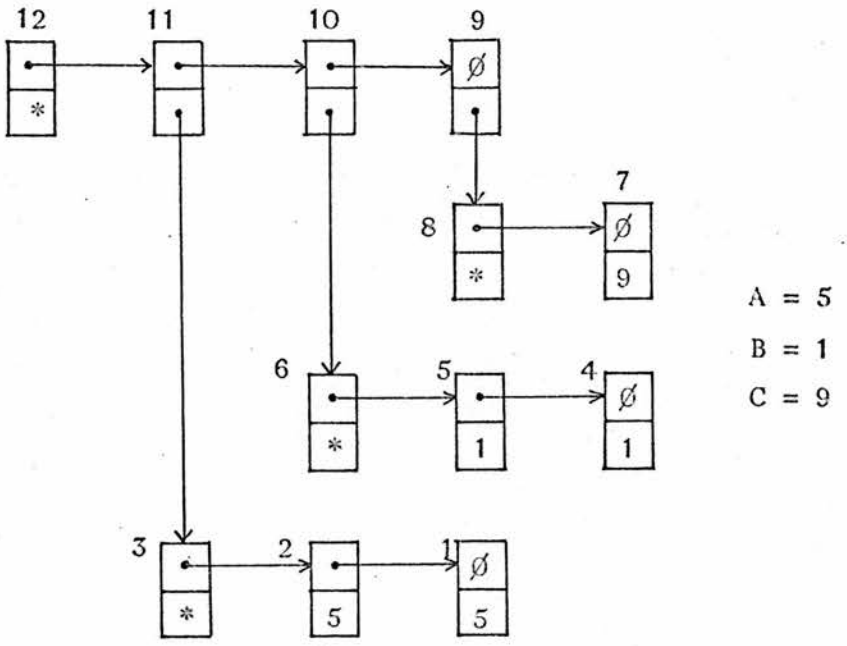
The entry would be 24 units long, taken in pairs:

1:	5	0	A
2:	5	10001	A
3:	10002	10002	
4:	1	0	B
5:	1	10004	B
6:	10005	10005	
7:	9	0	C
8:	10007	10007	
9:	10008	0	
10:	10006	10009	
11:	10003	10010	
12:	10011	10011	

Example of a Volume Array

Figure 18.

The numbers greater than 10,000 are internal pointers; the other numbers except for Ø are indexes to planes referenced in the Key array. Ø indicates the end of a pointer list. Diagrammatically, the same information becomes:



Equivalent Tree Structure

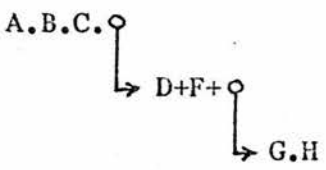
Figure 19.

Once this structure has been set up it is available for a variety of uses. Though its form is standardised it is probably incorrect to refer to it as a canonical structure, since it is still possible to have two different expressions and therefore data structures which, if they were restructured, could be shown to be the same. One way to establish this form of equivalence is to convert the expression to either the disjunctive normal form or the conjunctive normal form (Kaye, D., 1968). This was not attempted at this stage, but what was done was to expand

the nested expression into an un-nested expression not requiring brackets. As has already been shown, this is one way in which the description of a complex volume could be restructured so that the cutting plane procedure could be used to build it as a matrix.

The tree structure which has just been described, made this task a fairly straightforward operation, by using two routines DOTT and PLUS to process each of the levels corresponding to their names. As the operation passes from level to level these routines call each other recursively. An expression of the form:

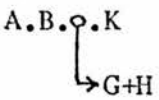
A.B.C.(D+F+G.H)



is expanded to give

A.B.C.D + A.B.C.F + A.B.C.G.H

At each of the 'dot' levels the routine DOTT outputs all the names, while at each 'plus' level the routine PLUS selects one name for output. In either case where an element in a level is a pointer to a lower level, a call is made to the other routine. Once a terminal symbol has been reached at a plus level the routine PLUS returns to the level above, whereas the DOTT routine only returns to the level above when it reaches the end of the list at its current level. At each plus level a pointer has to be kept for the last element output so that on the next call the next one can be selected. In the tree diagram this pointer is conveniently kept in the space *. In the case of the simple expression

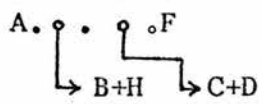


it can be seen that two runs through the tree will give the required output:

A.B.G.K + A.B.H.K

However, if the following expression is to be processed correctly:

A.(B+H).(C+D).F



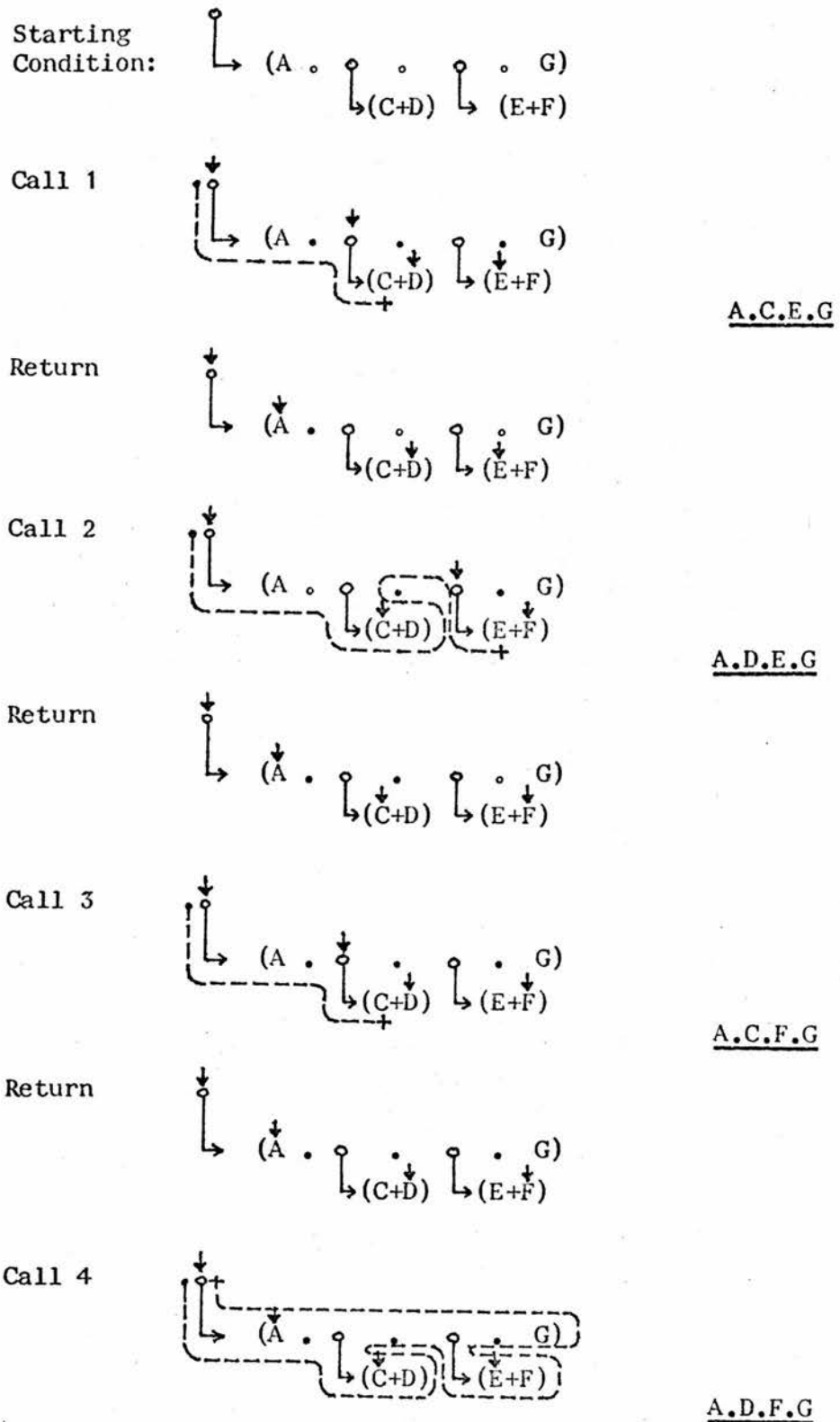
to give:

A.F.B.C + A.F.B.D + A.F.H.C + A.F.H.D

it can be seen that only one plus, sub-expression of a dot phrase, can be incremented at one time.

This correct selection was achieved by passing an 'activating signal' down from the top of the tree, and only when an echo of this signal returned from the bottom of the tree were any of the pointers incremented. An echo is only generated when the activating signal reaches a terminal element in the tree, or an incrementing operation takes a pointer to the end of a phrase at any level. When this happens not only is an echo generated but the pointer is reset to the beginning of the phrase. A pointer is kept at each of the dot levels to indicate at which position the activating signal should be passed down to lower levels. This pointer is incremented when an echo returns from a lower level in the same way that is the case for the plus levels. However, if no echo returns this pointer is reset to the first position in the phrase. This device ensures that all the correct combinations of elements are created. Diagrammatically, the operations for the expansion of the expression A.(C+D).(E+F).G can be presented in the following way:

VOLUME MATRIX CONSTRUCTION



Expansion of a Nested Boolean Expression

Figure 20

VOLUME MATRIX CONSTRUCTION

By repeatedly calling the PLUS routine for the same tree until an echo is returned to the top of the tree, the whole expression is ultimately expanded.

Written in the IMP language these two routines are:[1]

```
        %ROUTINE PLUS (%INTEGERNAME N,K)

        %INTEGER  J
        J=LIST(1,N)
        %IF  LIST(1,J)  10000  %THENSTART
        DOTT(LIST(1,J),K)
NEXT:   %IF  K=1  %THENSTART
        INCR(N,K)
        %FINISH
        %RETURN
        AL(L)=LIST(1,J)
        L=L+1
        -> NEXT
        %END


        %ROUTINE DOTT (%INTEGERNAME N,K)

        %INTEGER  J,KSAVE
        J=N
NEXT:   J=LIST(2,J)
        %IF  J=0  %THENRETURN
        %IF  KSAVE=0  %THEN -> NEAR
        K=0
        %IF  J=LIST(1,N)  %THEN K=1
NEAR:   %IF  LIST(1,J)  10000  %THENSTART
        PLUS(LIST(1,J),K)
        %IF K=0  %THEN LIST(1,N)=LIST(2,N)
NOWT:   %IF K=1  %THENSTART
        INCR(N,K)
        %IF K=1  %THENRETURN
        %FINISH
        -> NEXT
        %FINISH
        AL(L)=LIST(1,J)
        L=L+1
        -> NOWT
        %END
```

Both these routines call a common routine INCR which increments the positional pointer for the activating signal represented by the parameter

1. See Appendix C.

K. The arrays AL and LIST are global arrays declared external to these routines. LIST is dimensioned from 10001 to 11000 which allows internal pointers to be distinguished from the name pointers. The routine INCR is:[1]

```
%ROUTINE INCR (%INTEGERNAME N,K)

LIST(1,N)=LIST(2,LIST(1,N))
K=0
%IF LIST(1,N) = 0  %THENRETURN
K=1
LIST(1,N)=LIST(2,N)
%END
```

Once the expanded expression had been produced, there were still several tidying-up operations which needed to be done. In the expansion of the expression:

$$\underline{(A+B).(A+C)} \longrightarrow A.A + A.C + B.A + B.C$$

it can be seen that some of these dot phrases can be simplified.

$$A.A \longrightarrow A$$

to give $A + A.C + B.A + B.C$

It can be seen that if the whole expression is considered then:

$$A + A.B + A.C \longrightarrow A$$

because A.B and A.C are included in the space defined by A. This gives an expression of the form:

$$\underline{A + B.C}$$

Another form of simplification can be seen where the original expression is:

$$\underline{(A+B).('A+B)} \longrightarrow A.'A + A.C + 'A.B + B.C$$

But $'A.A \longrightarrow \emptyset$

so the final expression $\longrightarrow \underline{A.C + B.C + 'A.B}$

1. See Appendix C.

VOLUME MATRIX CONSTRUCTION

The simplification of the individual phrases was relatively easy to achieve. Using the Key array and, by marking the occurrence of a name for each phrase with a positive or negative flag, it was possible to locate duplication. Where the name had already been used it could subsequently be ignored. Where the complement of a name had already occurred, then the whole phrase could be deleted.

The removal of included phrases proved to be a more difficult task. The following process was evolved, but it was found at a later stage that there were already more general algorithms in existence which could have been adapted to this task. Consider the expression in Figure 21

1	2	3	4	5	6	7	8	9	10	11	12	13
A	B	+	B	E	+	A	C	E	+	A	B	C
7	4	Ø	12	9	Ø	11	13	5	Ø	1	2	8
3	2	Ø	4	3	Ø	4	4	2	Ø	1	1	3

Name Pointers

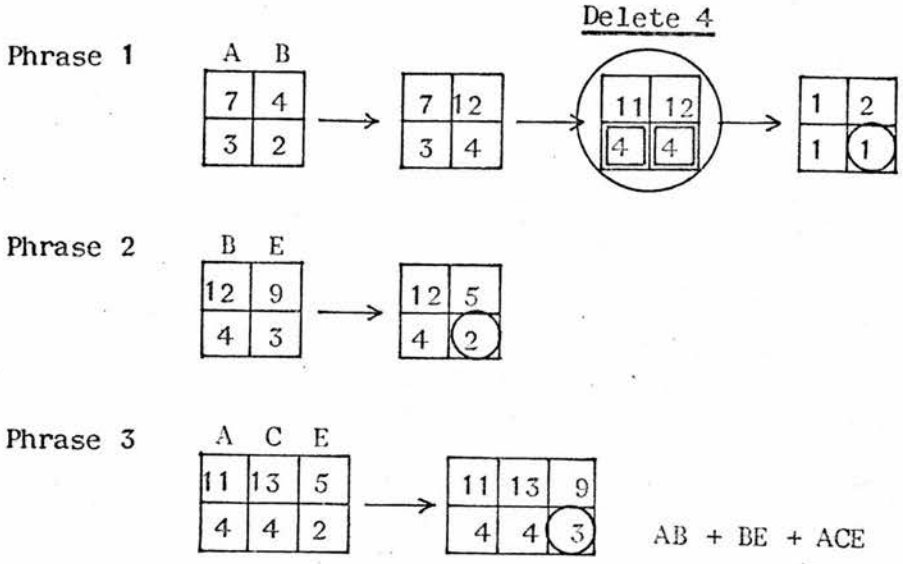
Corresponding Phrase Pointers

Data Structure for Removing Included Phrases

Figure 21.

The first stage was to set up the array of ring pointers shown in the diagram above. By passing through the expression once, using the Key array, it was possible to link all the elements with the same name into rings. Associated with each of these pointers is a reference to the phrase in which the next occurrence of the name is to be found. The basic principle of the ensuing operation is that if the phrase pointers of one phrase all point to the same following phrase, then this following phrase is redundant. The problem is to distinguish this phrase with corresponding elements, from all the intervening phrases. This can be done without undue searching by using the implicit order of the data

in the pointer array. For each phrase, increment the name-pointer associated with the smallest phrase-pointer until one of the phrase-pointers returns to the original phrase. Diagrammatically, this process is shown in Figure 22.



Removing Included Phrases

Figure 22.

Boolean expressions are used in the analysis of switching circuits and it was found in the analysis for the hardware display processor presented in the last section, that this test for inclusion was part of a more general procedure developed by Quine and McCluskey (1956) for use in electronics. This procedure also includes the ability to reduce the following expressions:

$$A.B.C + A.B.{}'C \longrightarrow A.B$$

The starting point for the Quine McCluskey method is the minterm list of the function which for the expression:

$$A.B.C + B.C.D \longrightarrow A.B.C.(D+{}'D) + B.C.D.(A+{}'A)$$

is given by $A.B.C.D + A.B.C.{}'D + {}'A.B.C.D$

VOLUME MATRIX CONSTRUCTION

This is the Disjunctive Normal form referred to previously, and because a particular collection of minterms is a unique representation, this expression can be represented by:

$$f(A,B,C,D) = m(7,14,15)$$

This decimal representation of a minterm is produced by representing A and ^{9}A by 1 or 0 and similarly for the other names, so that A.B.C.^{9}D becomes 1110 which is the binary representation for 14.

The Quine McCluskey method can be demonstrated by the example using the expression:[1]

$$f(A,B,C,D) = m(0,2,3,6,7,8,9,10,13)$$

The first step is to represent these minterms in their binary form as shown in Figure 23.

1	2				3				4
0	$\bar{A}$	$\bar{B}$	$\bar{C}$	$\bar{D}$	0	0	0	0	0
2	$\bar{A}$	$\bar{B}$	C	$\bar{D}$	0	0	1	0	1
3	$\bar{A}$	$\bar{B}$	C	D	0	0	1	1	2
6	$\bar{A}$	B	C	$\bar{D}$	0	1	1	0	2
7	$\bar{A}$	B	C	D	0	1	1	1	3
8	A	$\bar{B}$	$\bar{C}$	$\bar{D}$	1	0	0	0	1
9	A	$\bar{B}$	$\bar{C}$	D	1	0	0	1	2
10	A	$\bar{B}$	C	$\bar{D}$	1	0	1	0	2
13	A	B	$\bar{C}$	D	1	1	0	1	3

Minterms

1. Decimal
2. Names
3. Binary
4. Number of 1^{9}s in each Minterm.

Minterm List

Figure 23.

1. Example taken from Introduction to Switching Theory and Logical Design, Hill & Peterson.

VOLUME MATRIX CONSTRUCTION

	1					2					3				
0	0	0	0	0	✓	0	0	*	0	✓	*	0	*	0	
1	0	0	1	0	✓	*	0	0	0	✓	0	*	1	*	
	1	0	0	0	✓	0	0	1	*	✓					
2	0	0	1	1	✓	0	*	1	0	✓					
	0	1	1	0	✓	*	0	1	0						
	1	0	0	1	✓	1	0	0	*	✓					
	1	0	1	0	✓	1	0	*	0	✓					
3	0	1	1	1	✓	0	*	1	1	✓					
	1	1	0	1	✓	0	1	1	*	✓					
4						1	*	0	1						

Ordering the Minterm List and Reducing it

Figure 24.

The next step is to classify each of the binary minterms by the number of 1's it contains, and group them accordingly as shown in section 1 of the Figure shown above.

The second stage is to group together the minterms which are different in only one term, in other words the values in the four places corresponding to A,B,C,D for both minterms are the same except for one position. The minterms which are combined in this way are marked with a tick to show that they are included in a term in column 2. The combination of 1000 and 1001 gives 100*, and this is equivalent to the reduction:

$$A.{}'B.{}'C.{}'D + A.{}'B.{}'C.D \longrightarrow A.{}'B.{}'C$$

The second column can then be processed in the same way to give the

third column. In this example the elements in the third column will not combine to give a fourth column. Once this reduction is complete, the unmarked terms represent the phrases in the restructured expression, in this case:

$$'B.C.'D + A.'C.D + 'B.'D + 'A.C$$

It can be seen that this expression can still be reduced and the rest of the operation is comparable with the inclusion test already developed. This will remove 'B.'D.C since it is included in 'B.'D to give the minimal form of expression:

$$\underline{A.'C.D + 'B.'D + 'A.C}$$

The expansion algorithm described previously does not produce minterms. However, its output can still be processed in the same way. The phrases in the expression:

$$A.B.C + B.C.D$$

can be modified to give the minterms:

$$A.B.C.D + A.B.C.'D + 'A.B.C.D$$

However, they can also be regarded as intermediate results to the first stage of the Quine McCluskey procedure, so long as they are expressed with respect to the order: A,B,C,D as A.B.C.* and *.B.C.D. Even though the complete minimisation which this procedure produces, was not achieved, the routines developed were adequate for the next stage of the investigation.

Once the reduced single level boolean expression could be formed automatically, the next problem was how it could be used to generate volume

matrices. The first stage was to create the convex volumes defined by the individual phrases in the expression. This was done using the cutting plane procedure described above. By setting up the program:

```
A=(1,8,0,7,10,0,5,12,0,3,6,6);
B=(3,6,6,7,10,6,5,12,6,3,6,0);
C=(1,8,6,5,12,6,5,12,0,7,10,0);
D=(7,10,0,5,12,6,5,12,0,1,8,6);
E=(3,6,0,7,10,0,7,10,6,1,8,6);
F=(3,6,0,6,6,1,8,6,5,12,6);

V= A.B.C.D.E.F;
```

It was then possible to take the volume V and use it to control the cutting plane program to generate the edges of the volume expressed in the form A/B,p1/p2. The output from this test run is given below. The ordering to give the line boundaries of the volume's facets was carried out after the line segments had been generated. The use of the matrix data structure means that a particular line segment is only processed once for each cutting plane. If the process were to be carried out using polygon boundaries even if actual intersection points were only calculated once, there is still an unpleasant bookkeeping task locating the same line segment in the boundary of the adjacent facet. It was in the next stage of this study that the real advantage of the single occurrence of edge line segments really became apparent.

As has already been observed, the problem with the convex volumes created from the phrases of an expanded boolean expression is that they overlap in space. Once the convex volumes had been created as volume

VOLUME MATRIX CONSTRUCTION

Table 1 Test Output from Volume Construction Program

DATA: VOL=A.B.C.D.E.F;

PARSE

5121	1.0001	8.0001	0.0000	0	7
5121	5.0001	12.0001	0.0000	0	13
5121	7.0001	10.0001	0.0000	0	19
5121	3.0001	6.0001	0.0000	0	1
6145	3.0001	6.0001	6.0000	0	31
6145	7.0001	10.0001	6.0000	0	37
6145	5.0001	12.0001	6.0000	0	43
6145	1.0001	8.0001	6.0000	0	25
7169	1.0001	8.0001	6.0000	0	55
7169	5.0001	12.0001	6.0000	0	61
7169	5.0001	12.0001	0.0000	0	67
7169	1.0001	8.0001	0.0000	0	49
8193	7.0001	10.0001	0.0000	0	79
8193	5.0001	12.0001	0.0000	0	85
8193	5.0001	12.0001	6.0000	0	91
8193	7.0001	10.0001	6.0000	0	73
9217	3.0001	6.0001	0.0000	0	103
9217	7.0001	10.0001	0.0000	0	109
9217	7.0001	10.0001	6.0000	0	115
9217	3.0001	6.0001	6.0000	0	97
10241	3.0001	6.0001	0.0000	0	127
10241	3.0001	6.0001	6.0000	0	133
10241	1.0001	8.0001	6.0000	0	139
10241	1.0001	8.0001	0.0000	0	121

STOP 55

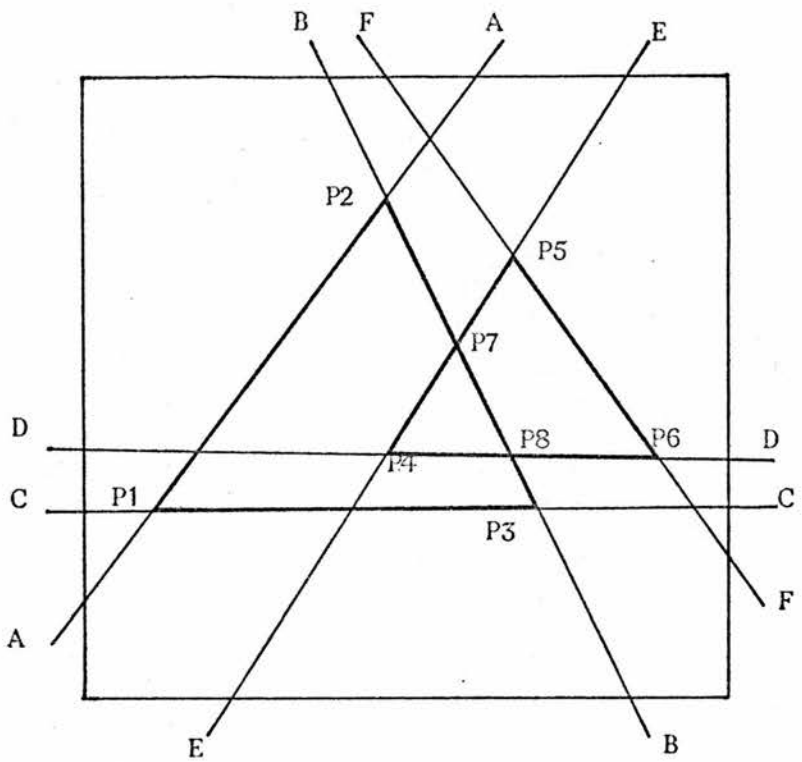
matrices there were two possible lines of development. Firstly, the overlapping regions could be removed using the matrix operations, or some other approach based on the wire frame models of volumes. Secondly, some approach could be developed which continued to use the properties of the boolean expression definition.

The second approach was investigated first. The basic principle behind this approach can be shown with the aid of the diagram in Figure 25 It is simpler to explain in terms of a two dimensional layout, but the

same relationships hold in three dimensions. Given the expression:

A.B.C + D.E.F

the geometrical realisation of this expression could be:



Two Dimensional Interpretations of a Boolean Expression

Figure 25.

If the triangle A.B.C is created first, then only those sections of D.E.F which lie outside A.B.C need to be created. In other words the area defined by:

$$'(A.B.C).D.E.F$$

$$('A+'B+'C).D.E.F$$

$$'A.D.E.F + 'B.D.E.F + 'C.D.E.F$$

However, it can be seen that this new collection of convex zones may well overlap themselves so the expression has to be modified to give:

VOLUME MATRIX CONSTRUCTION

$$'A.D.E.F + 'B.A.D.E.F + 'C.A.B.D.E.F$$

so the modified expression for the whole volume becomes:

$$\underline{A.B.C + 'A.D.E.F + 'B.A.D.E.F + 'C.A.B.D.E.F}$$

If the boundary lines of these zones are created for the triangles in the diagram above the result would be:

$$A.B.C \longrightarrow p_1, p_2, p_3, p_1$$

$$'A.D.E.F \longrightarrow \emptyset$$

$$'B.A.D.E.F \longrightarrow p_5, p_6, p_8, p_7, p_5$$

$$'C.A.B.D.E.F \longrightarrow \emptyset$$

This approach was tried out on a volume defined by the expression:

$$V1+V2 \longrightarrow A.B.C.D.E.F + G.H.I.J.K.L$$

The file for the two separate convex volumes is given in the next Table. Having done this, the modified version of the second convex volume - V2 was created, in this case:

$$'A.(V2) + 'B.A.(V2) + 'C.A.B.(V2) + 'D.A.B.C.(V2) + 'E.A.B.C.D.(V2) + 'F.A.B.C.D.E.(V2)$$

In practice, all but two of these phrases were empty. The new facets are shown by the negative facet names in the following file.

The plan and side elevation of this volume are shown in the next diagram. Two interpenetrating blocks at right angles to each other, the larger block being V1 and the smaller V2. In the modification of V2, its central portion is removed leaving a final description composed of three mutually exclusive blocks.

VOLUME MATRIX CONSTRUCTION

TABLE 2 Test Output from Volume Construction Program

V1	5121	1.0001	8.0001	0.0000	0	7
	5121	5.0001	12.0001	0.0000	0	13
	5121	7.0001	10.0001	0.0000	0	19
	5121	3.0001	6.0001	0.0000	0	1
	6145	3.0001	6.0001	6.0000	0	31
	6145	7.0001	10.0001	6.0000	0	37
	6145	5.0001	12.0001	6.0000	0	43
	6145	1.0001	8.0001	6.0000	0	25
	7169	1.0001	8.0001	6.0000	0	55
	7169	5.0001	12.0001	6.0000	0	61
	7169	5.0001	12.0001	0.0000	0	67
	7169	1.0001	8.0001	0.0000	0	49
	8193	7.0001	10.0001	0.0000	0	79
	8193	5.0001	12.0001	0.0000	0	85
	8193	5.0001	12.0001	6.0000	0	91
	8193	7.0001	10.0001	6.0000	0	73
	9217	3.0001	6.0001	0.0000	0	103
	9217	7.0001	10.0001	0.0000	0	109
	9217	7.0001	10.0001	6.0000	0	115
	9217	3.0001	6.0001	6.0000	0	97
	10241	3.0001	6.0001	0.0000	0	127
	10241	3.0001	6.0001	6.0000	0	133
	10241	1.0001	8.0001	6.0000	0	139
	10241	1.0001	8.0001	0.0000	0	121
V2	13313	1.0000	10.0001	1.0000	0	7
	13313	1.0000	10.0001	5.0000	0	13
	13313	3.0000	12.0002	5.0000	0	19
	13313	3.0000	12.0001	1.0000	0	1
	14337	3.0000	12.0001	1.0000	0	31
	14337	3.0000	12.0002	5.0000	0	37
	14337	7.0001	8.0001	5.0000	0	43
	14337	7.0001	8.0001	1.0000	0	25
	15361	7.0001	8.0001	1.0000	0	55
	15361	7.0001	8.0001	5.0000	0	61
	15361	5.0001	6.0001	5.0000	0	67
	15361	5.0001	6.0001	1.0000	0	49
	16385	5.0001	6.0001	1.0000	0	79
	16385	5.0001	6.0001	5.0000	0	85
	16385	1.0000	10.0001	5.0000	0	91
	16385	1.0000	10.0001	1.0000	0	73
	17409	1.0000	10.0001	5.0000	0	103
	17409	5.0001	6.0001	5.0000	0	109
	17409	7.0001	8.0001	5.0000	0	115
	17409	3.0000	12.0002	5.0000	0	97
	18433	3.0000	12.0001	1.0000	0	127
	18433	7.0001	8.0001	1.0000	0	133
	18433	5.0001	6.0001	1.0000	0	139
	18433	1.0000	10.0001	1.0000	0	121

VOLUME MATRIX CONSTRUCTION

TABLE 3 Test Output from Volume Construction Program

V2a	13313	1.0000	10.0001	1.0000	0	7
	13313	1.0000	10.0001	5.0000	0	13
	13313	3.0000	12.0002	5.0000	0	19
	13313	3.0000	12.0001	1.0000	0	1
	14337	4.0001	11.0001	1.0000	0	31
	14337	3.0000	12.0001	1.0000	0	37
	14337	3.0000	12.0002	5.0000	0	43
	14337	4.0001	11.0001	5.0000	0	25
	16385	2.0001	9.0001	5.0000	0	55
	16385	1.0000	10.0001	5.0000	0	61
	16385	1.0000	10.0001	1.0000	0	67
	16385	2.0001	9.0001	1.0000	0	49
	17409	4.0001	11.0001	5.0000	0	79
	17409	3.0000	12.0002	5.0000	0	85
	17409	1.0000	10.0001	5.0000	0	91
	17409	2.0001	9.0001	5.0000	0	73
	18433	2.0001	9.0001	1.0000	0	103
	18433	1.0000	10.0001	1.0000	0	109
	18433	3.0000	12.0001	1.0000	0	115
	18433	4.0001	11.0001	1.0000	0	97
	-7169	4.0001	11.0001	5.0000	0	127
	-7169	2.0001	9.0001	5.0000	0	133
	-7169	2.0001	9.0001	1.0000	0	139
	-7169	4.0001	11.0001	1.0000	0	121
V2b	14337	6.0001	9.0001	5.0000	0	151
	14337	7.0001	8.0001	5.0000	0	157
	14337	7.0001	8.0001	1.0000	0	163
	14337	6.0001	9.0001	1.0000	0	121
	15361	7.0001	8.0001	1.0000	0	175
	15361	7.0001	8.0001	5.0000	0	181
	15361	7.0001	8.0001	5.0000	0	181
	15361	5.0001	6.0001	5.0000	0	187
	15361	5.0001	6.0001	1.0000	0	169
	16385	4.0001	7.0001	1.0000	0	199
	16385	5.0001	6.0001	1.0000	0	205
	16385	5.0001	6.0001	5.0000	0	211
	16385	4.0001	7.0001	5.0000	0	193
	17409	4.0001	7.0001	5.0000	0	223
	17409	5.0001	6.0001	5.0000	0	229
	17409	7.0001	8.0001	5.0000	0	235
	17409	6.0001	9.0001	5.0000	0	217
	18433	6.0001	9.0001	1.0000	0	247
	18433	7.0001	8.0001	1.0000	0	253
	18433	5.0001	6.0001	1.0000	0	259
	18433	4.0001	7.0001	1.0000	0	241
	-9217	6.0001	9.0001	1.0000	0	271
	-9217	4.0001	7.0001	1.0000	0	277
	-9217	4.0001	7.0001	5.0000	0	283
	-9217	6.0001	9.0001	5.0000	0	265

VOLUME MATRIX CONSTRUCTION

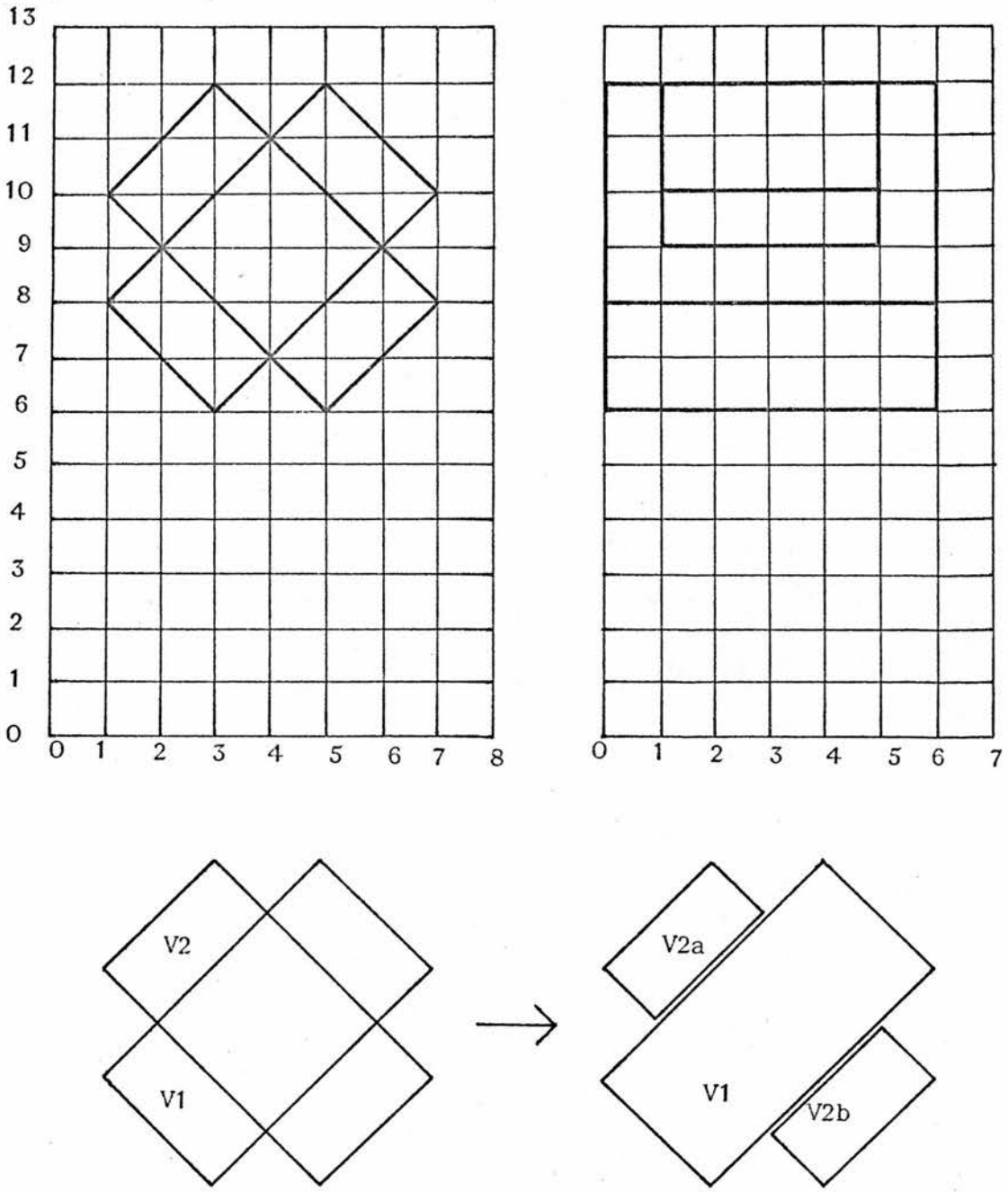


Diagram of Test Data

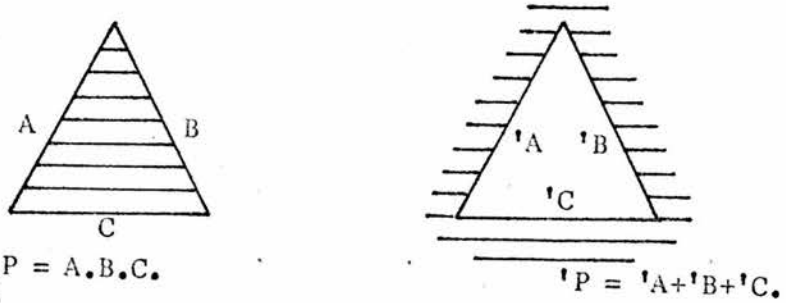
Figure 26

Though this process is completely automatic, it has obvious drawbacks. Many of the phrases which would be processed as if they were sub-

volumes would turn out to be empty. The number of subvolumes generated for testing even for these two starting volumes is eight and it can be seen that, with larger numbers of starting volumes, the number of tests escalates at a phenominal rate. If the original convex volumes, expressed as volume matrices, had been used, then many of these tests could have been rejected as unnecessary by a simple envelope test on the maximum and minimum values of the x,y,z values of the vertex coordinates.

A more direct way of using the volume matrices was found while attempting to do something else. Following the aim to make the maximum direct use of the boolean notation originally being used to describe the volumes, an attempt was made to extend the notation to include the bounded area of planes forming facets and the bounded length of lines forming edges, in a more explicit form. This investigation could not be taken very far, because the properties of the extended notation seemed to be ambiguous, and a more extensive study of the formal properties of this system was beyond the scope of the present study.

If the two triangular regions in the following diagrams are considered, the first as a 'solid', the second as a 'void': then, if the edge which



Solid and Void
Figure 27

divides A from 'A is expressed as @A then by inspecting the diagrams, it is possible to represent the edge of the triangle created by A as @A.B.C,

so that the boundaries of the two triangles P and 'P can then be expressed by:

$$\partial P \longrightarrow \partial(A.B.C) \longrightarrow \partial A.B.C + \partial B.C.A + \partial C.A.B$$

$$\partial' P \longrightarrow \partial('A+'B+'C) \longrightarrow \partial'A.'B.'C + \partial'B.'C.'A + \partial'C.'A.'B$$

This gives a result which is similar in form to the result of differentiating a vector product:

$$(u.v)' \longrightarrow u'.v + u.v'$$

$$(uxv)' \longrightarrow u'xv + uxv'$$

Also, the entities ∂A and $\partial A'$ seemed to correspond to the closures of the sets A and 'A. However, if A represents the space $\emptyset \leqslant a.x + b.y + c$ then it has already been shown that it is convenient to make 'A represent the space $\emptyset \geqslant a.x + b.y + c$, which makes ∂A correspond to $\emptyset = a.x + b.y + c$. To achieve this, it seems to be necessary to make $\partial A = \partial'A$. For a limited set of operations the following list of productions for the extended calculus gave consistent results which corresponded to the cutting-plane operations described above and some of the operations carried out implicitly during the matrix manipulations:

1	$A.'A$	$\longrightarrow$	$\emptyset$ *
2	$A+'A$	$\longrightarrow$	I
3	$\partial'A$	$\longrightarrow$	∂A
4	$\partial(A.B)$	$\longrightarrow$	$\partial A.B + \partial B.A$
5	$\partial(A+B)$	$\longrightarrow$	$\partial A.'B + \partial B.'A$
6	$\partial A.I$	$\longrightarrow$	∂A
7	$\partial A.\emptyset$	$\longrightarrow$	$\emptyset$
8	$\partial A.A$	$\longrightarrow$	∂A
9	$\partial A.'A$	$\longrightarrow$	∂A

10	$'(@A)$	$\longrightarrow$	I	*
11	$@'(A.B)$	$\longrightarrow$	$@(A.B)$	
12	$@A.B + @A.C$	$\longrightarrow$	$@A.(B+C)$	
13	$@(@A)$	$\longrightarrow$	$\emptyset$	*
14	$@(@(A.B))$	$\longrightarrow$	$@A.@B$	
15	$A + @A$	$\longrightarrow$	A	
16	$B + @A$	$\longrightarrow$	B	*

These conversions are in no specific order, and it is possible to 'derive' some of them from others. Giving meaning to some of these productions is at times difficult. The boundary of $\emptyset$ or I does not seem to make much sense. The following conversions suggest that the definition of such a boundary is arbitrary or undefined in this system:

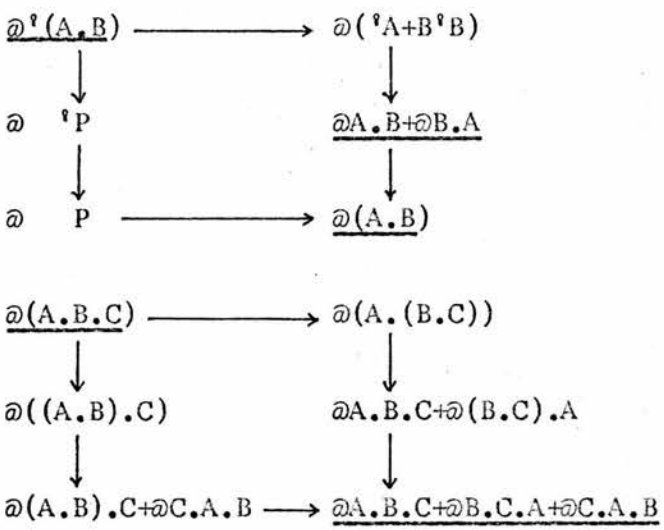
$$\begin{array}{ccccc} \underline{@(A.'A)} & \xrightarrow{4} & @A.'A+@A+'A & \xrightarrow{8,9} & @A+@A \\ \downarrow 1 & & \downarrow 12,2 & & \downarrow \\ @(\emptyset) & ---- & @A.(I) & \xrightarrow{6} & \underline{@A} \end{array}$$

$$\begin{array}{ccccc} @(A+'A) & \xrightarrow{} & @A.A+@A.'A & \xrightarrow{8,9} & @A+@A \\ \downarrow 2 & & \downarrow 12,2 & & \downarrow \\ @(I) & ---- & @A.(I) & \xrightarrow{6} & \underline{@A} \end{array}$$

The real difficulties seem to arise from the productions marked with an asterisk. The justification for these conversions, if it exists, seems to be of the order $00 + 1 = 00$.

It is the consequence of this argument which has not been investigated and on which the uses of these productions seem to rest. For example, the intersection of two volumes with a vertex or edge in contact in the

matrix manipulations has been regarded as empty so that $V.'V \rightarrow \emptyset$, but the tests would give the result $V.'V \rightarrow \partial V$. If ∂A is regarded as infinitesimally small in the context of A, then it can be ignored but does this argument extend to make (16) a valid conversion? Similarly, the entity $\partial(\partial A)$, if compared with the double differential, should exist but in the context of ∂A can be ignored. It would appear that an operation equivalent to taking the limit in calculus is required. Bearing these reservations in mind, the following results of direct substitution intuitively appear to be satisfactory:



The reduction of a volume defined by the boolean expression A.B.C.D.E.F to the matrix form would require the boundary of the volume as facets to be defined, followed by the edges of the facets, and finally the points representing the vertices, in other words, the definition of 'the boundary of a boundary of a boundary'. As has already been discussed, the boundary of a boundary of A is defined to be empty. However, the boundary of a boundary of intersecting spaces cannot be empty, and the following expansion seems to be necessary or something like it.

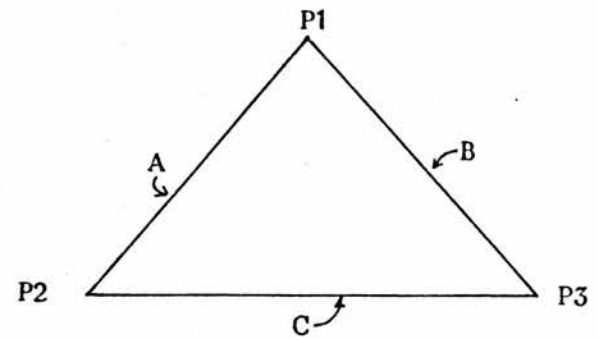
VOLUME MATRIX CONSTRUCTION

$$\partial(\partial(A.B))$$
$$\downarrow$$
$$\partial(\partial A.B + \partial B.A)$$
$$\downarrow$$
$$\partial(\partial A.B).'(\partial B.A) + \partial(\partial B.A).'(\partial A.B)$$
$$\downarrow$$
$$(\partial\partial A.B + \partial A.\partial B).('(\partial B) + 'A) + (\partial\partial B.A + \partial A.\partial B).('(\partial A) + 'B)$$
$$\downarrow$$
$$(\partial A.\partial B).('A) + (\partial A.\partial B).('B)$$
$$\downarrow$$
$$\partial A.\partial B + \partial A.\partial B$$
$$\downarrow$$
$$\underline{\partial A.\partial B}$$

The application of this form of substitution for the boundary of a boundary of a triangle would need to take the form:

$$\underline{\partial(\partial(A.B.C)) \longrightarrow \partial A.\partial B.C + \partial A.\partial C.B + \partial B.\partial C.A}$$

This result corresponds to the three vertices of the triangles:



P1: $\partial A.\partial B.C$

P2: $\partial A.\partial C.B$

P3: $\partial B.\partial C.A$

The Boundary of the Boundary of a Triangle

Figure 28

It is clear that the ideas presented in the last few pages have not been fully worked through. The goal of this line of development can be illustrated by considering the OBLIX hidden line removal algorithm. If the previous notation were effectively implemented, then this algorithm could be expressed very compactly in the following way:

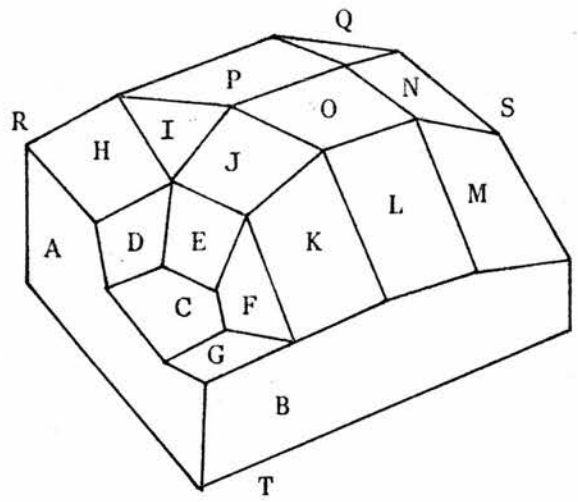
```
CYCLE J=1,N
E = @A.'(B(J))
DRAW E
A = A + B(J)
REPEAT
```

Even if the ideas in the last few pages cannot be developed further, it is clear that the two alternative ways for representing volumes discussed in this section would be valuable in the same system. So far the boolean expression description has been discussed in the role of data input. In the next chapter it will be shown that there is a way in which it can be used to create displays. Certain volume data are easier to enter in point coordinate form and so it is necessary to briefly consider the creation of boolean expression descriptions from such a structure.

The volume descriptions in question have already been examined briefly in Chapter 7, and an example is shown in Figure 47 in that chapter. The data which results from a topographic survey are typically in the form of point heights. If these points are triangulated, then an approximation of the original surface is created using plane facets. If each facet is given a name then if a three level boolean expression is used each name need only occurs in the volume definition once. An example for a simple volume is shown in Figure 29, and a simple algorithm can be constructed to create such a description from the set of triangles.

Conclusions

The work in this Section allowed the ideas presented in the Chapter on



$(A.B.S.R).(Q.N.M.L.K.O.J.I.P + H).(\overline{D}+\overline{E}+\overline{F}+\overline{G}+\overline{C})$

Creating a Boolean Expression from a Triangulated Surface

Figure 29.

System Development to be extended. At one level it allowed a set theoretic data-base access language to be constructed. This has not been investigated in detail but what appeared to be the primitive operations were discussed briefly in Chapter 7. In such a language the boolean expression can be used to manipulate inverted files related to property names. If such names are extended to reference ordered sets or relation coordinates, and a particular type is defined to cover the geometrical objects considered in this Section then the boolean expressions can also be used to define zones in space. Finally, at this level it should also be possible to include the volume matrices as a particular form of ordered set.

If further work on these ideas is successful then the boolean expression offers to provide some useful English Language forms as alternatives. The intersection of two or more name sets for example

corresponds to the use of nouns in apposition:

John, the farmer ...

The set of objects referenced by 'John' when intersected with the set of objects referenced by 'the farmer' would give the subject of this sentence. The use of the relative clause seems to be an extension of the same idea:

John, who was a farmer ...

In this case the who seems to act as a dummy name in apposition with 'John', and the first stage is to reduce the relative clause to give a set corresponding to 'who'. Similarly, the relation pair stored as a coordinate can be used in the form:

John, the father of Andrew ...

'Andrew' would initially be matched with the second element in a set of ordered pairs called 'Father, Child'. The 'of' would act as an operator which returns the set of Fathers matching a child called Andrew. This set would then be intersected with the name set 'John'.

It seems that there would be several modes in which such expressions could be interpreted. If taken as a question, then they would be matched with information in the data base to see if they were true, not known or false.

Is John the father of Andrew?

This stagement would access and check this relationship in the data structure. In such simple cases the information would have to be explicitly stored for the system to be able to give any other response than 'not known'. Where sets are defined as the result of applying

functions to other sets it is possible for the system to generate answers not explicitly defined by external statements. If the question:

Is London in England?

is posed, then if the location of London is stored as a geographic coordinate then a point in polygon test on the boundaries of 'England' will be able to provide the appropriate true or false answer. It would appear that where cluster analysis has been used to create a classification a very similar procedure should be applicable. An important aspect of including a spatial location with a named physical object is that it seems the simplest way to provide a unique identifier.

Though these ideas have been briefly examined they will have to be developed in future work. There are considerable difficulties in maintaining the internal bookkeeping for such a free structure which have not been fully resolved. The next useful statement which will be included in the volume definition language described above will be the RESTRICTION. If names are being used to access name sets then with a large data-base it will become necessary to use very long boolean expressions to denote a single object. Consequently, when addressing operations to a data-base it will be useful to initially define the universe of discourse, by a series of restrictions on the meaning of names. If this is implemented in a block structured language then the restriction takes the place of declarations. An example of such an expression would be

John: John who lives at 99 Fish Street

Within its block this statement would restrict John to the individual referenced by the address. At the end of the block John could again be used generally. An alternative implementation could be constructed round a specially defined use of the definite and the indefinite article.

This area seems to be very rich in future possibilities. However, the next stage of this study went on to consider the alternative response of the system to questions about location. Instead of giving a verbal response the system could produce a diagram, map or other display which illustrates the relationships which are required. This subject is discussed in the next chapter on Graphic Models.

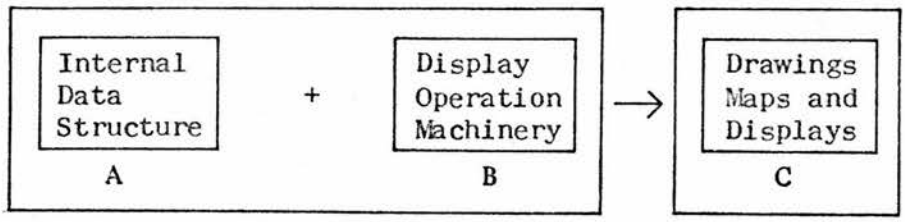
CHAPTER 12

GRAPHIC MODELS

GRAPHIC MODELS

It is convenient to divide the data structures used to represent information for automatic data processing into three groups. The first covers the external forms of data used to enter information into the machine system, the second the internal forms used in machine processing, while the third covers the representation used to communicate the results of data processing to the system users.

In the preceding chapters the primary interest was in the organisation of the internal data, but it was impossible to discuss this subject without using external forms of representation. For this reason many of the issues which concern the input and output of information have already been implicitly covered. The formalism used for entering and manipulating volume descriptions has been described at some length, but the generation of graphic output needs to be examined in greater detail. Though many illustrations were given to show the result of the internal manipulations which were being discussed, no indication was given how these could be produced automatically. The subject of this Chapter is the process summarised in Figure 1.



Generating Graphic Output

Figure 1.

There is a wide variety of data structures which can potentially be included in box A, in that they can be successfully converted into a

graphic form of output. Similarly, there is a wide variety of graphic forms which can be included in box C. Since it was difficult to consider all the possibilities the first stage was to limit the subject to a manageable size. It seemed reasonable to restrict the internal data structures which would be considered to those analysed in the earlier chapters in this Section. The contents of box C were more difficult to restrict, but some help was provided by the limitations imposed by the machinery available in box B to automatically create graphic output. Most common display devices are only capable of rendering points or lines, and to some extent this restricts the kinds of display it is easy to produce.

It so happens that the data structures which have already been discussed are particularly suited to be used with line drawing devices. Though it could be argued that this was no accident, but was the result of catering for existing output machinery, doing so would only present one aspect of the overall problem. It is relatively easy to design a simple machine for displaying areas directly. Consider a machine which projects a circular area of light onto a photographic film. The position and the radius of the disc of illuminated film could be controlled by input data. More complex shapes could be built up by overlaying a collection of discs with the appropriate size and position. If the recording signal is a light beam bright enough for the photographic film to be fully exposed from a single disc of light then the total figure will be produced by boolean addition since: exposure + exposure = exposure. The problem with this scheme is the form that the internal data structure would have to take and its lack of versatility for

internal manipulations.

Another problem is that having an internal data structure which can be easily converted into a line drawing does not guarantee that the drawing will be easy to read.

The use of parametric equations to define smooth surfaces for aircraft and ship hulls appeared to provide a simple way of creating line drawings. If one of the parameters is fixed then a surface line is generated by continuously varying the other parameter (Forrest, 1968 and Armit,1970). These lines are generally three dimensional, in that they cannot be contained within a plane. The result is that when they are projected onto a display plane the drawing produced can be deceptive.[1] Armit suggested that the only reliable way of displaying these surfaces as a drawing was to generate surface contours or plane parallel profile sections.

The ideal solution seemed to be an internal data structure which was not only suitable for the internal manipulations necessary, but which also made it simple to create readable graphic output. If a compromise had to be made then its nature would depend on the application of the system. For example, where data are obtained from a remote sensing device, and the output is used to drive automatic cutting machinery, then the economic emphasis is likely to be different from a Computer Aided Design System. The most efficient solution is to design the data structure to suit the principal work load. In the first case if drawings are necessary then the overhead from creating a secondary data structure for

1. See Figure 47, Chapter 7.

display purposes could be lost in the savings produced by using a well designed data structure for the main activity.

It is very difficult to provide rules for creating easily read drawings. If a graphic model is initially considered to be a pattern on a plane surface, then the graphic effects which can be used to build up the display can be listed under three broad headings:

Two Dimensional Effects

(1) Position marks - points.

(2) Links between points - Lines.

(3) Boundary lines - a partitioned display surface.

(4) Surface character - Solid areas of colours, tone or texture.

(5) Object shape - similar and dissimilar objects.

(6) Reference grids - measurements.

Three Dimensional Effects

(7) Overlap - obscuring recognised shapes.

(8) Linear perspective - relative size of recognised objects.

(9) Aerial perspective - colour tone and texture variations.

(10) Transparency - Overlap with colour and tone variations.

(11) Illumination - shadows and tone variations.

(12) Reference Grids - Contours, relative spatial location.

(13) Surface character - reflectivity, lustre.

(14) Edge Effects - local tone and colour variations.

Movement Effects

(15) Parallax - patterns of movement consistent with perspective.

(16) Changing shape - relative movement other than parallax.

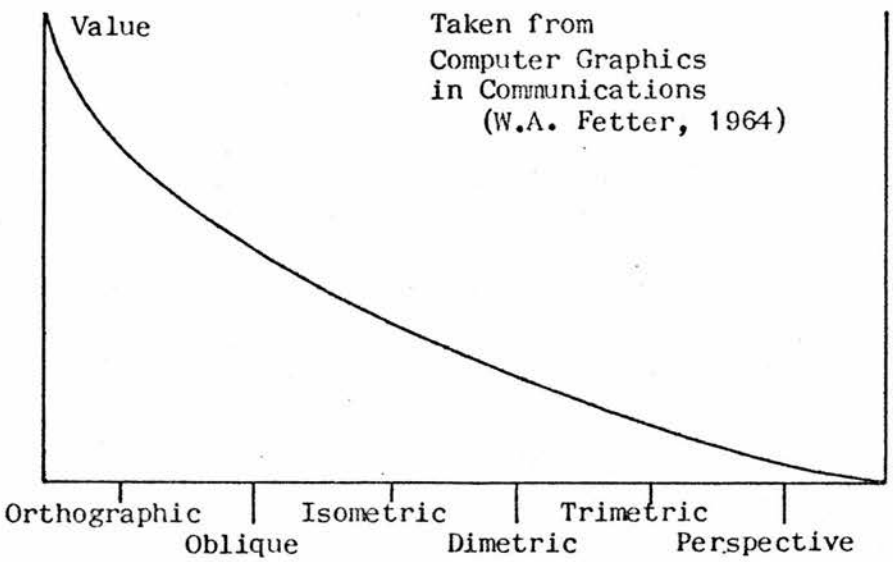
(17) After Image - Colour and Textural effects.

(18) Optical illusions, impossible figures, figure field effects.

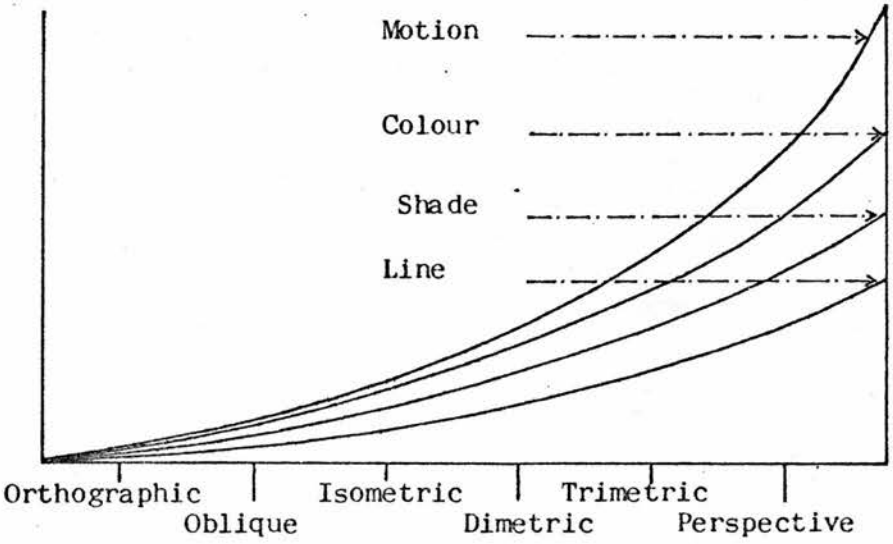
It can be seen that many of these effects are interrelated. The problem is that it is very difficult to improve on this classification, for example to give independent categories. When presented with a display it is possible to respond to it as a flat two dimensional surface and it is possible to 'look into it' and respond to three dimensional effects. The kind of problems which can occur if only one effect is taken into account can be illustrated in the following way. In cartography symbols of different size are often used to indicate different populations for towns and cities. If the radius of a circle is made proportional to the numbers of people in each town, then two competing effects may be produced. If the map is read two dimensionally then the correct relative size of each town may be assessed. However, if the map is read three dimensionally then though the rank ordering of towns will still be correct, the tendency will be to relate the absolute sizes of the populations to the depth of the symbol and an incorrect impression may be given. Inconsistent or competing effects can be useful, however; they are often employed consciously by artists and graphic designers to give a sense of movement, and produce dynamic effects in a static display.

In order to limit the contents of box C in Figure 1, it was decided to concentrate on the creation of technical drawings. On one hand, many years of practical use have evolved conventions which reduce the chance of ambiguity; on the other, special training in their use make it less likely that competing effects will lead to incorrect interpretations

being made. Starting from the volume descriptions described in previous chapters, the first problem to be tackled was the creation of plan, section, and simple projected drawings.



Ease of Measurement



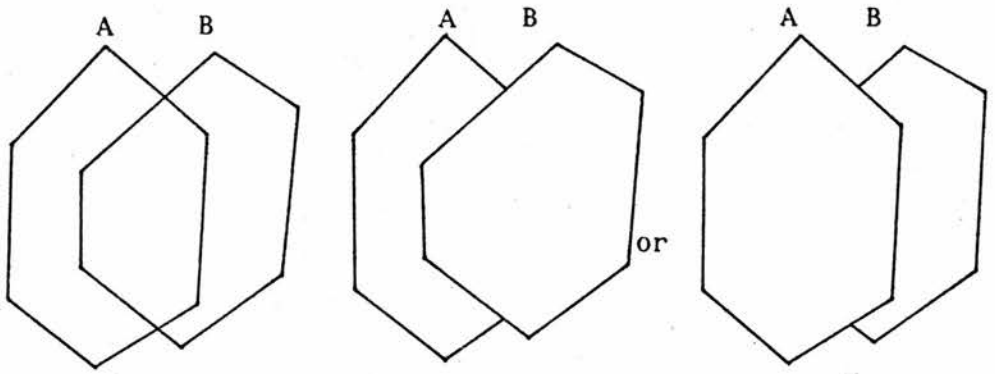
Ease of Understanding Drawings

Figure 2.

The advantage of starting with simple perspective projections is that straight lines in three dimensional space are transformed into straight

lines in the display space. This means that a data structure where lines are represented by their end vertices can be transformed merely by redefining the vertex coordinates. This is a simple task to program. The difficulty in creating a display is that all the edge lines of a volume will be transferred to the picture plane. It consequently becomes necessary to remove those lines and line segments which would be obscured by intervening objects. This is the hidden line problem and its simplest form occurs as the overlap of polygons.

The use of overlap in a display is principally a two dimensional exercise. If the outline of an object is considered as it would appear on a picture plane, it is a polygon. If two polygons overlap on the picture plane, then it is merely a matter of removing the section of the polygon boundary which is overlapped. That is the portion of the boundary of the furthest away polygon which lies inside the other polygon. The two alternatives for polygons A and B are shown in Figure 3.

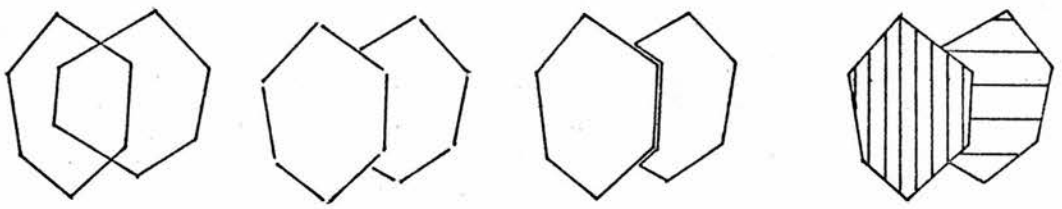


Overlap of Two Polygons

Figure 3.

This process has already been discussed in the description of the OBLIX program in the second Section.

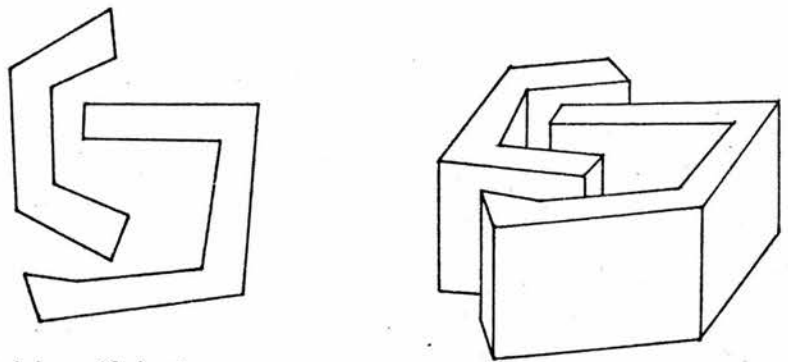
If the polygons are being treated as zones, for example where they may be shaded, then the operation is a little more complex. Though the obscuring polygon boundary will be the same, the polygon which has been covered will have an incomplete boundary. A new section of boundary will have to be generated from the boundary of the other polygon. This procedure is summarised in Figure 4.



Overlap of Polygon Zones

Figure 4.

Where more complex objects are required in a display the overlap of outlines is not adequate to solve the hidden line problem. The difficulty is illustrated by the interlocking volumes in Figure 5. In this case it is not possible to classify one object as totally in front of the other, though it is possible to classify each surface



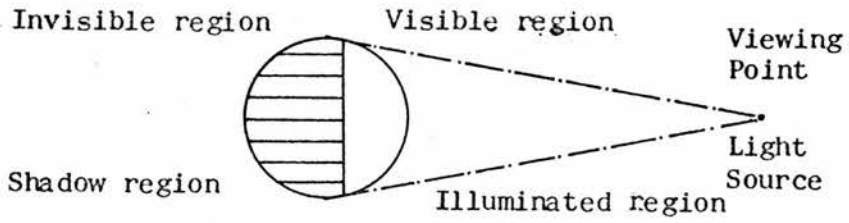
Interlocking Objects

Figure 5.

facet in this way. In this case the use of overlap depends on having plane facets but before restricting the problem it appeared to be worth examining the general hidden line problem where curved surfaces were included as part of a volume's description.

Hidden Line Removal - The General Case

It is clear that it is only the surface of an opaque object which is seen. It is also clear that in a particular direction of viewing, the nearest surface to the eye will be the visible surface. Other surfaces will be obscured by the first. This is taken as the simplest evidence that light travels in straight lines, the geometrical relationships being the same as those which govern the definition of shadows. If the source of illumination is considered to be located at the viewing point, then the visible portions of the object will coincide with the illuminated portions while, conversely, the invisible portion will be in shadow.

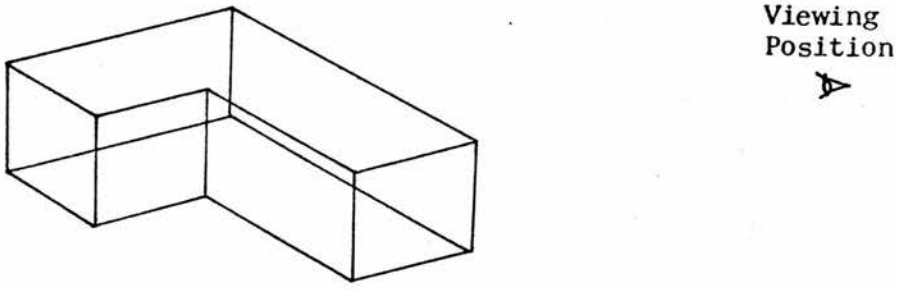


The Relationship between Shadows and Visibility

Figure 6.

An alternative statement of the hidden line problem is the projection onto the picture plane of only those points or lines which lie within the illuminated area of an object's surface, when the source of illumi-

nation coincides with the viewing position. If the boundary line between the illuminated area and the shadow area is called the shadow line contour, then in the simple case shown in Figure 6 this line will also divide the areas of surface which are facing the light source from those which are facing away from it. From the discussion of the OBLIX hidden line removal algorithm, it is clear that this will not always be the case. Where a collection of objects is considered then it is possible for one object to obscure or cast a shadow on the front face of another object. Also where a concave object is encountered, it is possible for front facing surfaces to be in shadow. For this reason it is useful to define the boundary which separates the front facing surfaces from back facing surfaces and give it a different name, in this discussion the 'hidden line contour'.

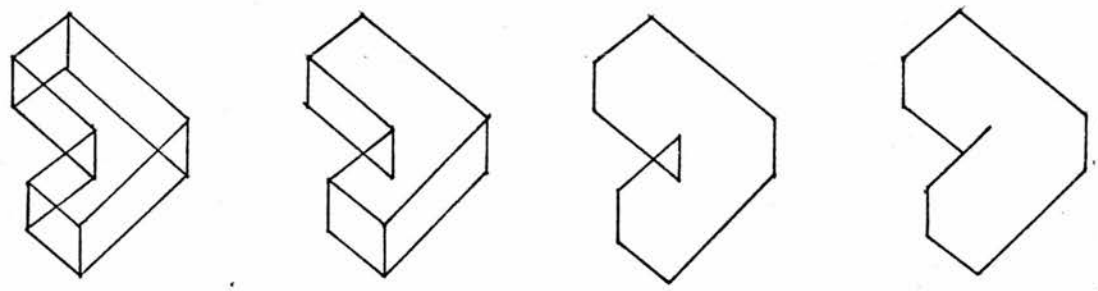


'L' Shaped Block

Figure 7

Consider the 'L' shaped block shown in Figure 7 , if the back facing surfaces are removed then the result is that shown in Figure 8.

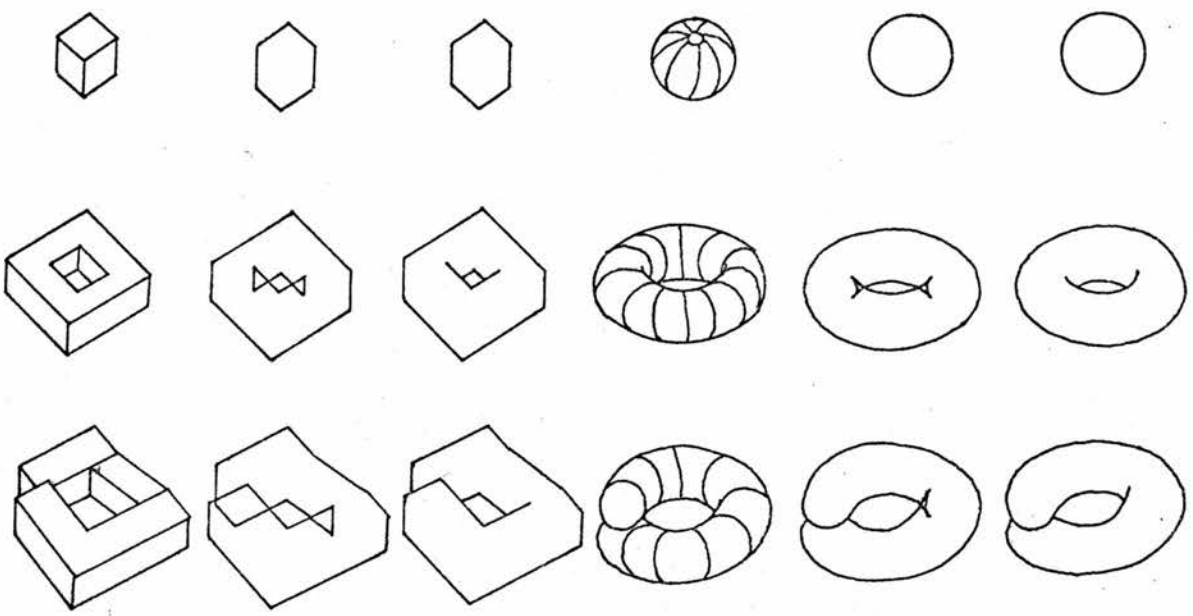
The second diagram in Figure 8 shows all the front facing facets and the edges between them. The third diagram shows only the projection of the hidden line contour onto the picture plane, where the lines between



Front Facing Facets for the 'L' Shaped Block

Figure 8.

two front facing surfaces have been removed. It can be seen in this case that a simple polygon overlay procedure will produce the final corrected display with all the hidden lines removed. However, the advantage of the general approach appears as soon as objects with smooth curved surfaces are considered.



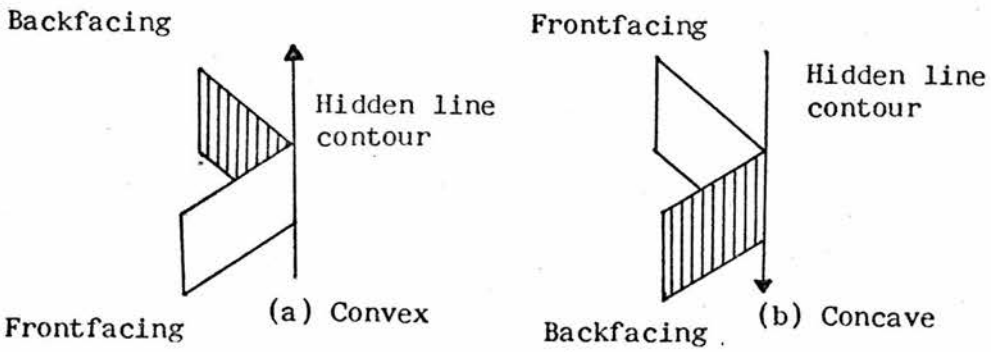
Hidden Line and Shadow Line Contours

Figure 9.

In most of the early programs written for hidden line removal, the discarding of back facets was regarded as a preprocessing task which was carried out before the real problem of hidden line elimination was undertaken. It can be seen that in the case of smooth unfaceted objects this is a task which takes on a much more important role. Though the definition of the hidden line contour does not totally solve the display problem it is one step in the right direction, since the shadow line contour when projected on the picture plane is a section of the hidden line contour. In the case of smooth surfaced objects, even when they are built up from surface patches, these lines may need to be constructed since neither need already exist as edges in the original volume description.

Properties of Hidden Line and Shadow Line Contours

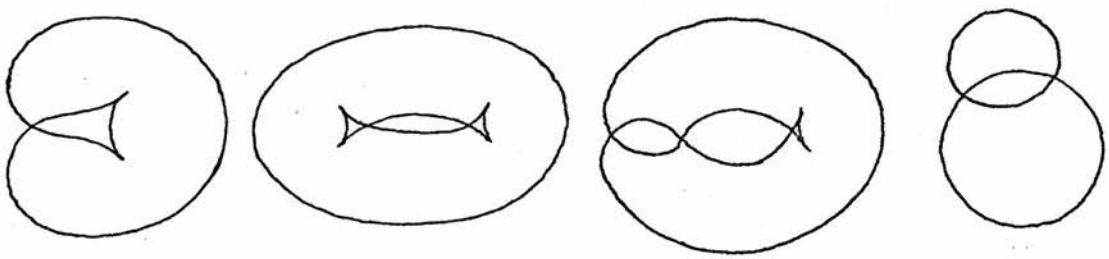
The hidden line contour is the line which divides the front facing section of an object's surface from the back facing section. It can be seen by studying the object in Figure 8 that the front facing surface does not have to be located in front of its adjacent back facing surface. Both the alternatives shown in Figure 10 are possible.



Front and Back Facing Surfaces and the Hidden Line Contour

Figure 10.

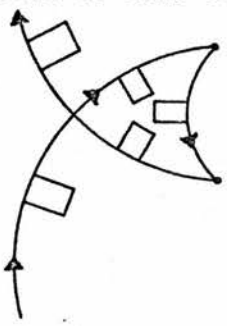
It is interesting to note that in diagram (a) the hidden line contour if formed by a convex section of the surface, whereas in diagram (b) the hidden line contour is formed by a concave section of the object's surface. When an object has a concave section to its surface, then it is possible to have a self crossing hidden-line contour:



Self Crossing Hidden Line Contour

Figure 11.

When projected on the picture plane, the adjacent sections of surface lying on each side of this line will appear on the same side of it. Where this line has a directional characteristic this side will always be the same relative to this direction.

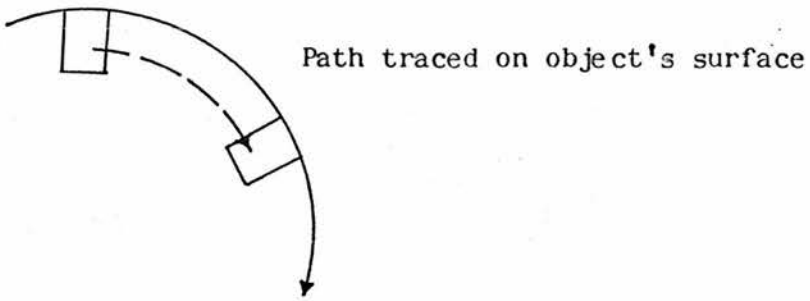


Inside and Outside the Hidden Line Contour

Figure 12.

The surface within a single contour loop will be a single connected area. This means that within one of these loops it should be possible to trace a path from one section of the visible surface to another, without having

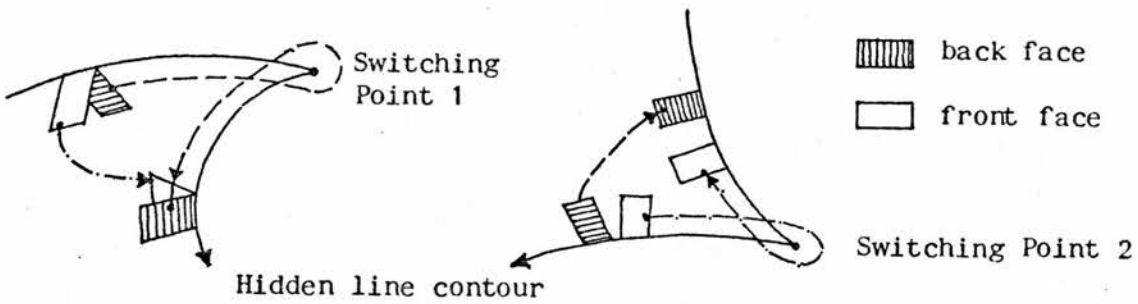
to cross the hidden line contour.



Linking visible points on the Volume Surface

Figure 13.

Where such a path, projected on the picture plane, does in fact have to cross the projected contour line, it indicates that the hidden line contour has passed from a convex section of the surface to a concave section. The same is true in reverse, where the path from one back-facing section to another crosses the hidden line contour on the picture plane, there has been a switch from a concave surface to a convex surface.

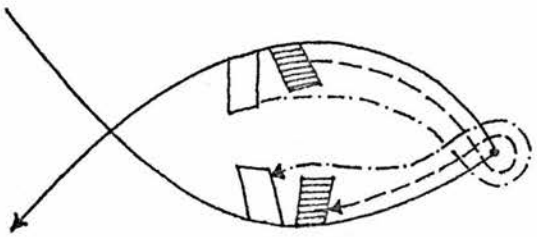


Hidden Line Contour Switching Points

Figure 14.

The concave section of the hidden line contour will never be visible. This is obvious when it is realised that this section of the line occurs where a back face is nearer the viewer than the associated front face, Figure 10 , so a solid section of the object will always lie between this section of the hidden line contour and the viewer. It has been assumed for a smooth continuous surface that, where this transition occurs, the

hidden line contour projected on the picture plane will form a cusp or sharp change of direction. For hidden line removal, the location of these points would mean that this section of the contour line could immediately be discarded. It would appear possible for the two switching points to coincide as shown in Figure 15.

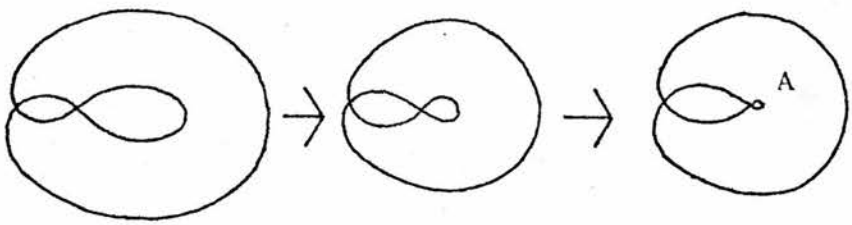


Double switching point

Double Switching Point

Figure 15.

This corresponds with the situation which would result from the sequence of operations shown in Figure 16.



Creating a Double Switching Point

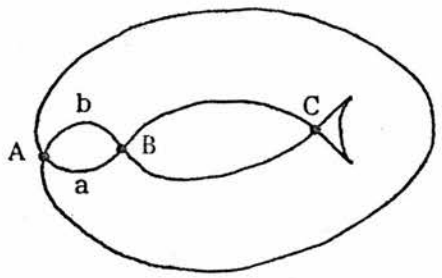
Figure 16.

At such a point a ray of light from the viewing point could conceivably pass through a point sized hole at A.

Conversion of the Hidden Line Contour to the Shadow Line Contour

Assuming the hidden line contour has been created for a single object, where there is only one contour loop and where this loop is not self

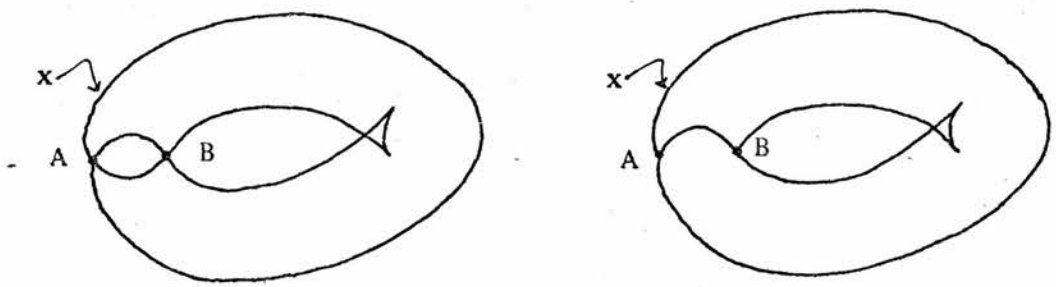
crossing, then the hidden line contour will be the same as the shadow line contour. Where the shadow line contour does cross itself, there are two principal ways in which this intersection can occur:



Intersection Points for a Hidden Line Contour

Figure 17.

The two intersections 'A' and 'B' correspond to the condition of simple overlap. Consequently, the shadow line contour depends on whether 'a' is dominant or 'b' is dominant. Even though, when A is resolved, B will also be resolved, this situation will usually be treated as though A and B are independent, and the hidden line contour will be examined intersection point by intersection point. If, following the direction of the arrow, the intersection points are found and resolved in sequence, then it can be seen that line 'x' will be switched off at 'A' and switched back on again at 'B'.



Simple Overlap, Pairs of Intersection Points

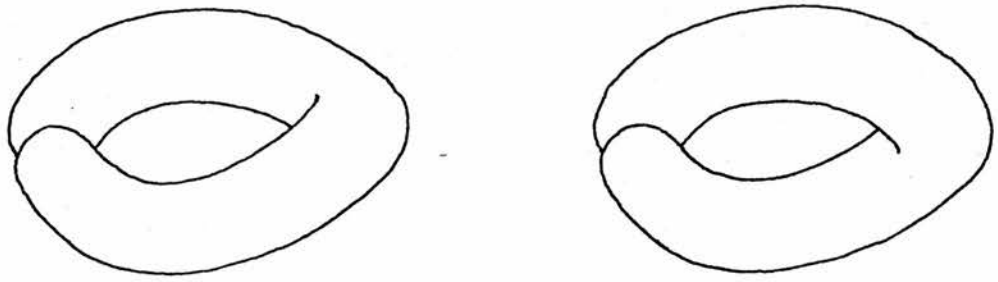
Figure 18.

This leaves the single intersection point 'C'. It can be seen that the straight forward processing of intersection points will not be sufficient in this case. However, assuming that it is possible to locate the switching points 'd' and 'e' then it is possible to remove the line segment 'de'. This leaves a shadow line contour of the form:

Complex Overlap. Single Intersection Point

Figure 19.

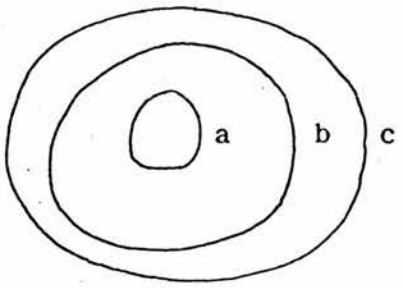
The two line segments which intersect at 'C': '1' and 'm' can then be tested to see which one lies nearer the viewer and is therefore dominant. It can be noted in passing that the resolution of 'C' is independent of 'A' and 'B', so that the following two results are possible:



Dependent and Independent Intersection Points

Figure 20.

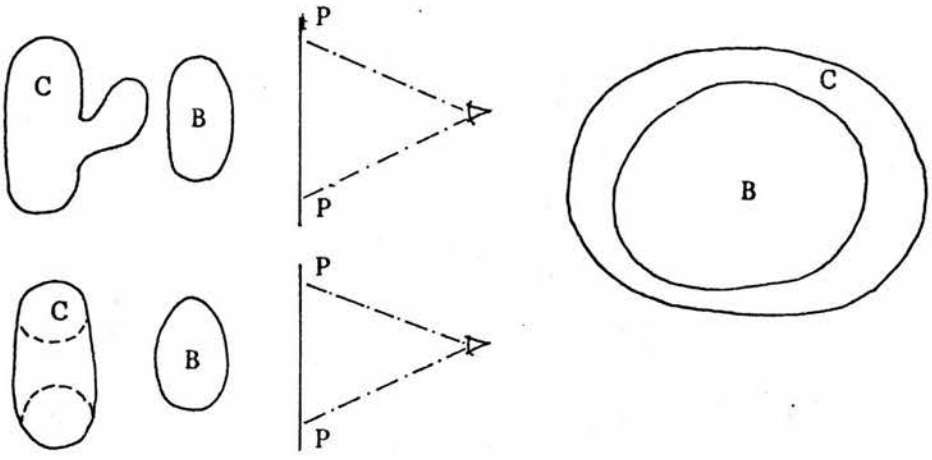
Where there are more than one contour loop in a projected scene, it is not always possible to depend on tests for overlap carried out solely at points of intersection. Consider the following hidden line contours:



Three Concentric Hidden Line Contour Loops

Figure 21.

'b' and 'c' are the normal outline contours for objects B and C, however 'a' is a secondary contour for the object C. 'a' can be caused in two ways: firstly, it can be the result of a lump on the surface of C, and secondly, it can be the result of a circular hole punched through C:



Alternative Interpretations, and the Shadow Line Contour

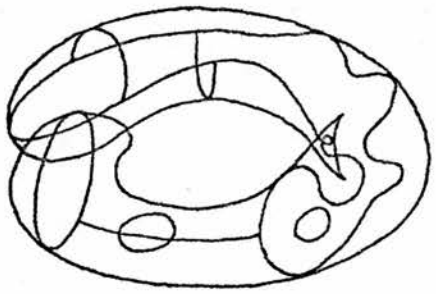
Figure 22.

In both cases if B lies in front of C then the secondary boundary loop of C will be obscured. It thus seems to be necessary to carry out some

form of overlap test for all concentric loops at least once.

Hidden or Surface Line Removal

The conversion of the hidden line contour to the shadow line contour is only part of the process normally referred to as hidden line removal. There will often be surface lines which are important parts of the display that do not form part of an object's edge as projected on the display surface. If the same object discussed above is considered, with a selection of surface lines, the resulting image projected on the picture plane could be:

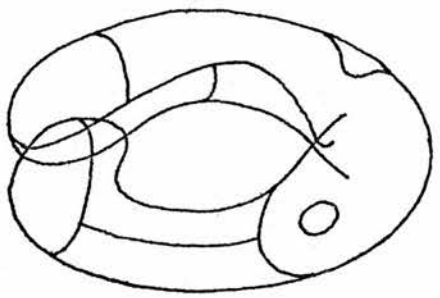


Object with Surface boundary lines

Figure 23

As has already been observed surface lines will only be visible if they occur on a front facing surface. In the general case where the surface is not made up of plane facets, this may be difficult to evaluate from the projection of the object on the picture plane. In Figure 23 it can be seen that a line which passes from a front face to a back face will create a projected line inside the hidden line contour, but touching it at the point of transition. It is not necessary for a

surface line which touches the hidden line contour to change its visibility at this point of contact, but if it does change its visibility then either it touches a contour line or, in certain circumstances, crosses it. If sections of the surface lines which are on the rear faces of the object in Figure 23 are removed the result is shown in Figure 24.

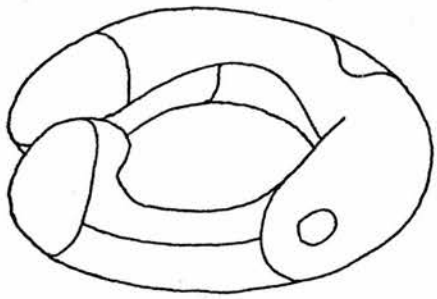


Removing Surface Lines from Back Surfaces

Figure 24.

The final stage in the removal of hidden surface lines is to locate those lines which are on front surfaces but which are obscured by either nearer sections of the same object, or other independent objects lying in the same line of sight. It would appear that this can be done using a simple overlap test. The final rendering of the object shown in Figure 24 is given in Figure 25 . What was initially a jumble of lines has become an intelligible display.

The exact way in which operations described in the last three pages would be implemented must depend on the data structure used for representing curved surfaces. Since only faceted objects have been examined in earlier chapters it was not possible to take this line of



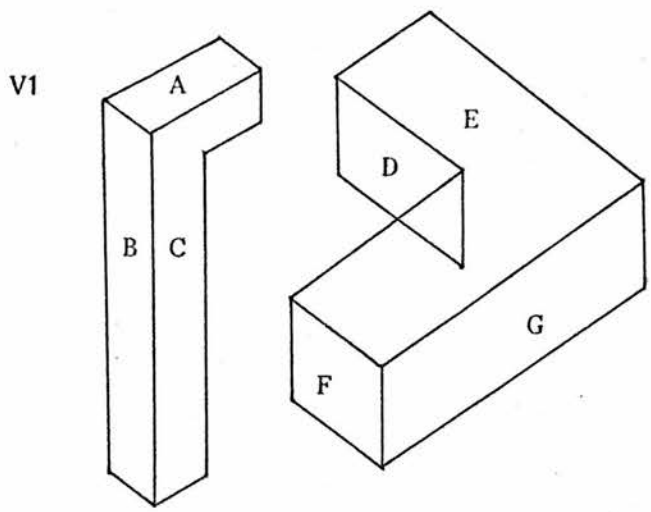
Final Drawing

Figure 25.

investigation any further. Though the same strategy can be applied to objects made up from plane surfaces, there is a variety of simplifications which can be made because the two contour lines must be made up from existing edge lines. It is interesting to note that in the OBLIX study, even though a very simple hidden line algorithm was used, it was found necessary to define the shadow line contour for drawings where the profile lines were suppressed to provide the minimum graphic information for the surface to be intelligible.

A technique for removing hidden lines from displays of faceted objects was developed from the polygon overlay study described in Chapter 7. It can be seen that the output file from the overlay routine provides sufficient information to select visible lines, and therefore to create a display. Consider the objects shown in Figure 26.

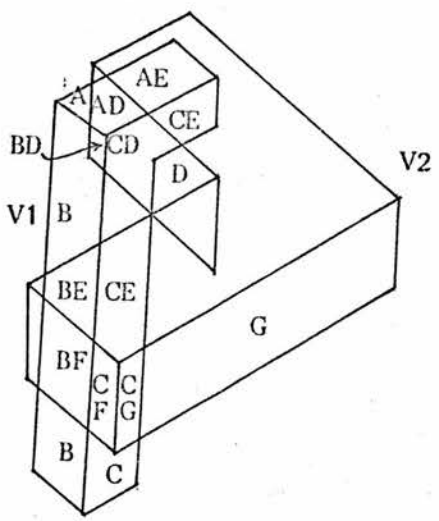
If V1 and V2 in Figure 26 are overlayed as shown in Figure 27 then if the operation were simple overlay the output information would be the zones of overlap between differently named polygon facets. For



Stage I: Remove Back Faces

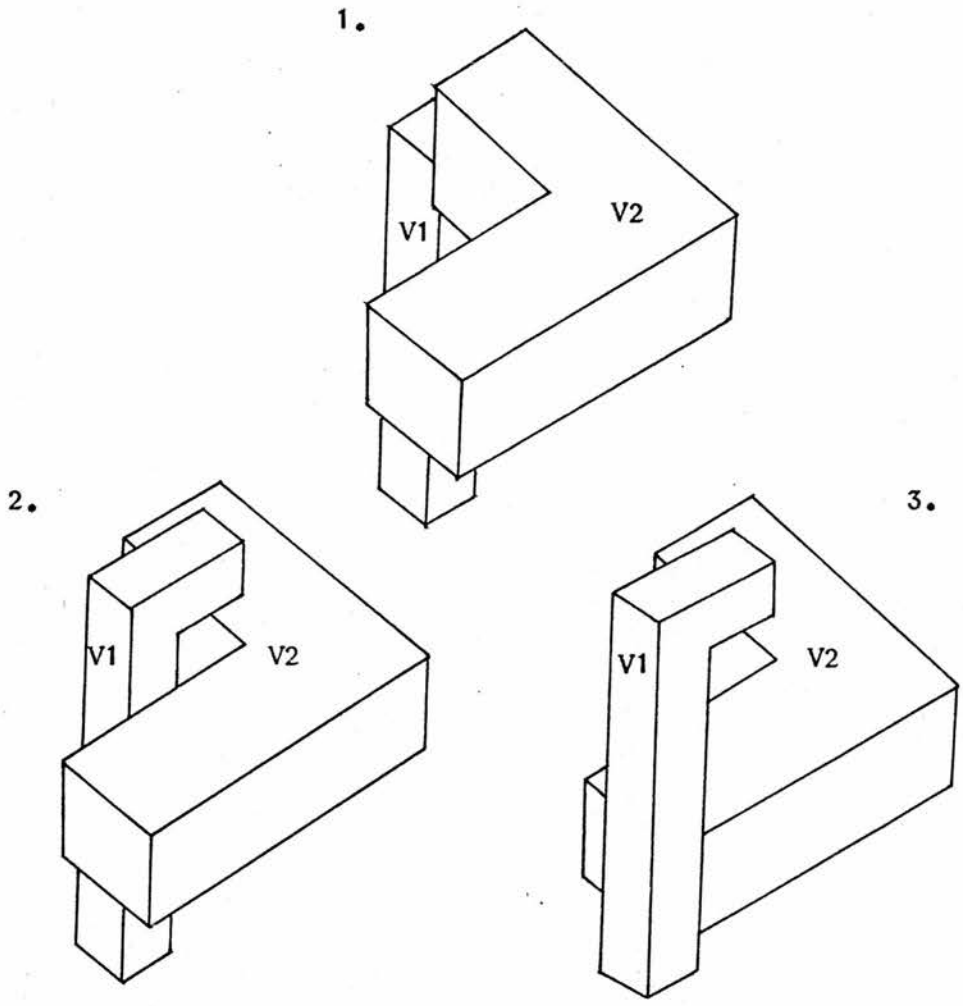
Figure 26

a three dimensional object the coordinates not only provide the relative location in the picture plane they also provide depth information. In this case there appear to be at least three relationships which these two objects could be holding to each other, from which the next stage in the hidden line removal must choose one.



Polygon Overlay of Facets of V1 and V2

Figure 27



Possible Spatial Relationships of V1 and V2

Figure 28

The simplest way to demonstrate this procedure is to use an example. Consider facet B in volume V1. The output from the overlay routine could be a file of the vertices of B including the intersection points where the polygon B overlaps other polygons in the picture plane. The overlay will have been carried out using only the x' and y' values of the perspective coordinate x', y', z'. However the z' coordinate will be included in the point record for each vertex. This means that when

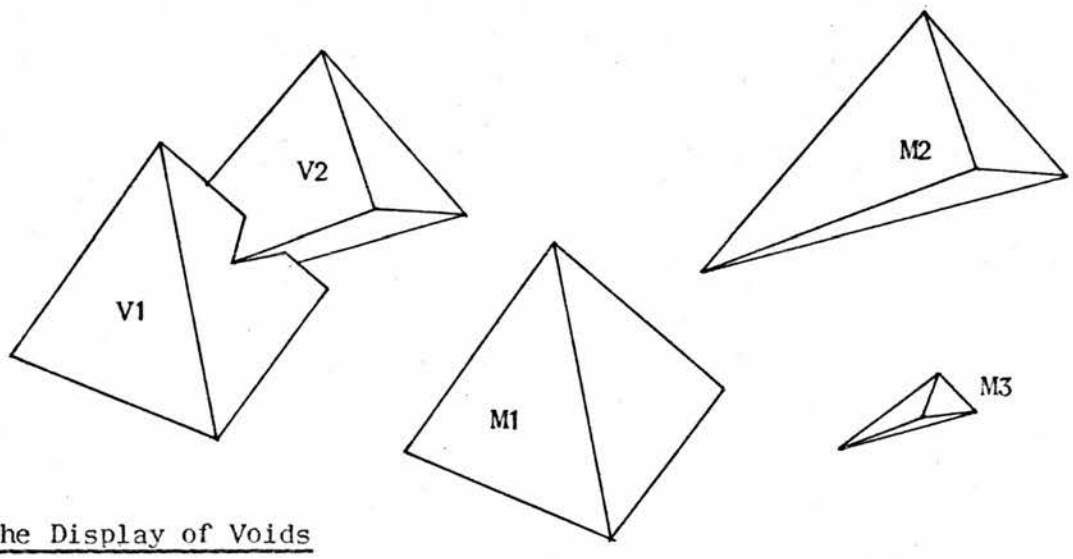
an intersection point is reached in the boundary of B it is possible by following the ring pointers linking identical points - in the picture plane - to find the z' values for all the other boundaries which intersect B at this intersection point. If the boundary of B is nearest to the picture plane then the intersection point may be ignored. Once the visibility of the first point in a boundary has been found this procedure makes it possible to extract the visible lines from the overlay file by one pass through the data.

This procedure is similar to most of the other hidden line algorithms which have been developed for faceted objects. It makes use of scene coherence, implicitly during the coordinate sorting operation which underlies the polygon overlay strategy. Other algorithms carry out a similar operation to localise comparisons, some in the z direction first some in the y direction first. In this case the choice was already made because the routine was not originally developed for hidden line removal. It can be seen that the provision for extra values to be included as associated data in each coordinate record, provided the versatility which made this extension of the polygon overlay routine possible, by including the z coordinate.

The next stage in this investigation was to consider the use of volume matrices as an input data structure for volume display procedures. It was clear that the matrix could be converted into a collection of facet boundary loops, using the multiplication procedure described at the beginning of this section. This data could then be used in conjunction with the overlay routine which has just been described. In

fact, it is not essential to presort the edges into boundary rings, since the overlay routine will link the line segments with matching end coordinates together, but doing so will save storage space.

There was one problem which still had to be resolved before the matrix data structure could be used in this way. This was the occurrence of 'Voids', which could appear as part of a matrix description of an object resulting from solutions to the naming problem. Consider the object shown in Figure 29 and the three components which would occur in the matrix for the object.

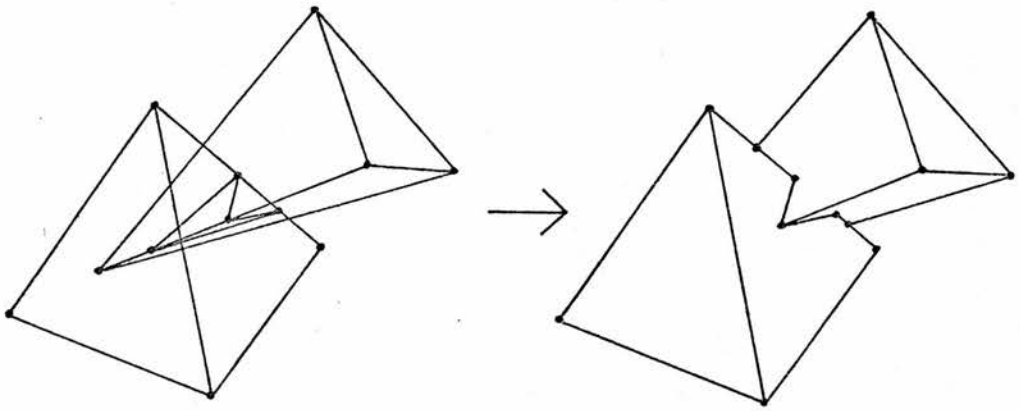


The Display of Voids

Figure 29.

At first sight it would appear that the intersection lines between V1 and V2 still needed to be established. In fact they have already been found - they occur in the edges of the void M3. Consequently, all the vertices necessary for the final display, subject to overlap tests, exist explicitly in the data structure. All that is necessary is to link them together to create the correct edges for the output drawing.

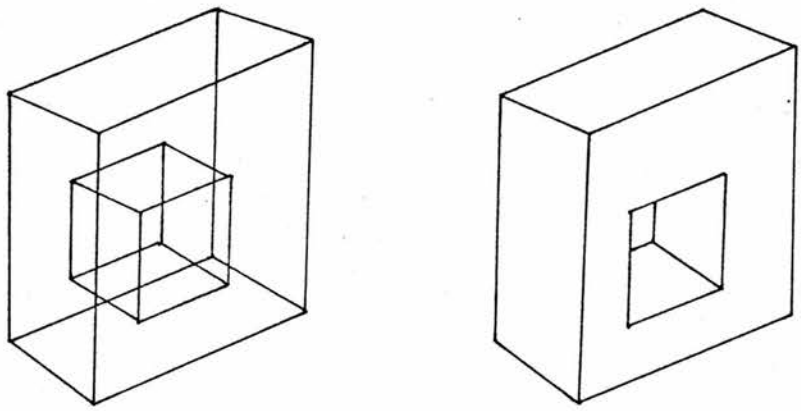
Volumes M1 and M2 can be treated in the same way as the objects in Figure 28 . The back faces can be discarded and the polygon facets overlayed in the picture plane. If the same procedure is adopted for the void, then diagrammatically the result would be successful in this case, in that simple overlap tests will remove the unwanted line sections. This operation is illustrated in Figure 30.



Polygon Overlay for Voids

Figure 30

If the volume shown in Figure 31 is represented by the matrix for a solid and a void, then it is clear for this alternative that the edges of the back faces of the void will be needed, to create the correct output.



Back Faces of a Void used for Creating Graphic Output

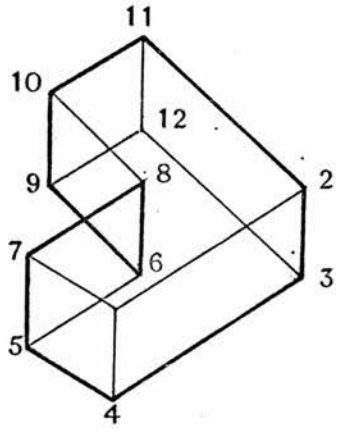
Figure 31

It has already been observed that unless specific tests are carried out it is not known whether a particular matrix represents a solid or a void. The usual test depends on the sense of rotation exhibited by the boundary points of a polygon when it is projected onto the picture plane, and this would be reversed for a void. Consequently, if the same test is used, then for a void, front faces would be discarded and back faces selected. Though this would solve the problem for Figure 31 , it would create difficulties in the original case shown in Figure 30 . Overall it appeared that there would be enough information in the output file from a polygon overlay operation, to create a correct display if all the facets of volumes were included in the input file. In this context it seemed possible to limit the selection procedure, so that it only processed front facets: only when these were found to intersect with back facets would it be necessary to access the edges of back planes.

The Generation of Hidden Line Contours

A different line of investigation, based on the use of volume matrices seemed worth following. This was the automatic creation of Hidden Line Contours from the original matrix data structure. Even if it did not provide an alternative strategy for solving the hidden line problem it would be useful to classify the silhouette lines of a volume for other graphic effects.

Take as an example the 'L' shaped object shown in Figure 32
If the surface facets are classified as front or back planes, indicated by + and - in Figure 32 , then it is possible to auto-



	−	+	+	−	+	−	−	+
−	A	3	4	5	6	9	12	
+	4	B	1				3	2
+	5	4	C	7				1
−			5	D	8			7
+	9			6	E	10		8
−	12				9	F	11	10
−	3	2				12	G	11
+		1	7	8	10	11	2	H

Volume Matrix for an 'L' Shaped Block

Figure 32.

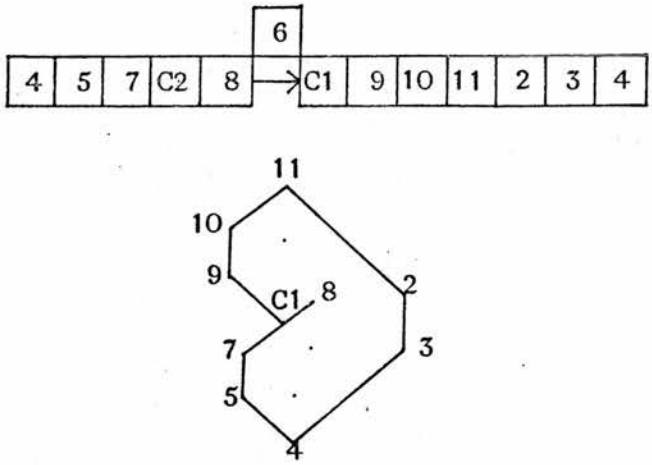
matically select all the line segments which lie between positive and negative surfaces. If these line segments are chain sorted the result will be either the boundary of the front or the boundary of back facing areas, depending on the direction of the line, as shown in Figure

	4	5	7	8	6	9	10	11	2	3	4
+ve		A	D	D	D	A	F	F	G	G	A
-ve	C	C	H	E	E	E	H	H	B	B	

Hidden Line Contour

Figure 33.

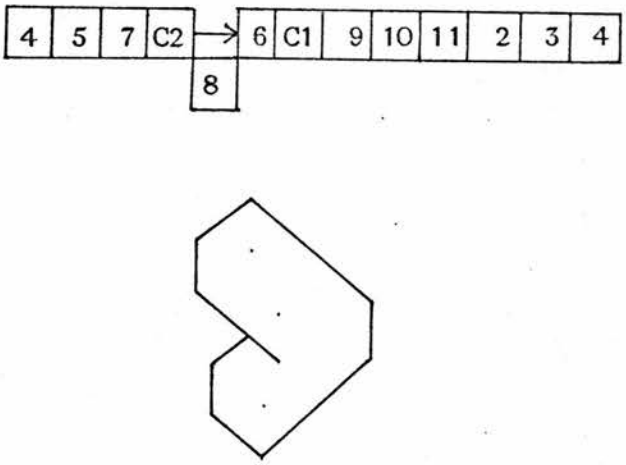
Where this line crosses itself as it does in the view shown in Figure 32 then the crossing point in the picture plane C will correspond to two points on the surface of the object C1 and C2. If C1 is nearer to the picture plane than C2, then assuming that the switching points are 8 and 6, then the shadow line contour can be created automatically to give the object shown in Figure 36.



Shadow Line Contour Alternative I

Figure 34

Alternatively, if C2 is found to lie nearer to the picture plane then the object would be displayed in the form shown in Figure 35.

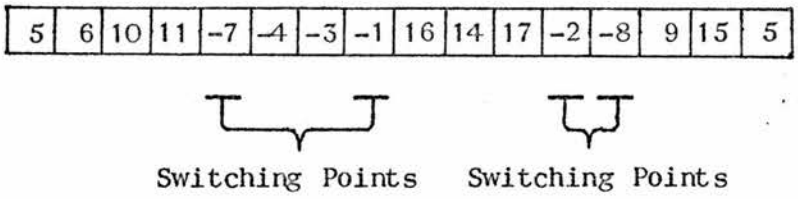


Shadow Line Contour Alternative II

Figure 35.

The problem was to locate the switching points. One approach which gave interesting results, was to include in the volume description an explicit classification of the angle at which facets meet at edges. If the end points of line segments in re-entrant surfaces are tagged by making their name negative in the volume matrix, then if they are

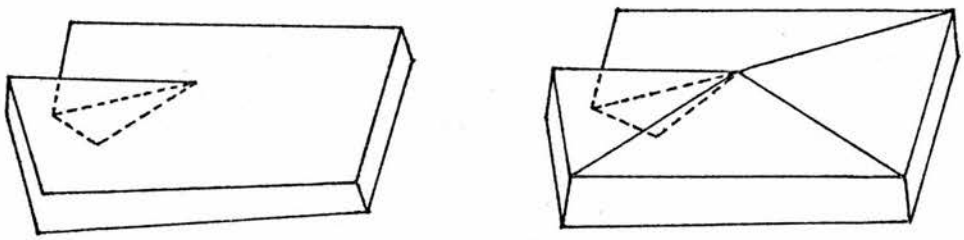
selected as part of the hidden line contour it will be immediately apparent which points are switching points. For example, in the stream of points in Figure 36 defining a hidden line contour, the switching points will be the first and last negative points in each group of negative points



Selecting Switching Points

Figure 36

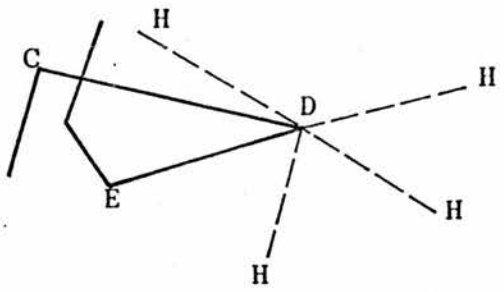
However, this approach did not solve all the problems. When a double switching point occurs there are no line segments to indicate the concave section. The only way to extend this principle seemed to require facets with point contact to be represented in the matrix.



Volumes with Double Switching Points

Figure 37

An alternative test which would work for the volumes in Figure 37 is that for D, the switching point, the lines DH lie in the obtuse angle formed by the Hidden Line Contour CDE, as shown in Figure 38 rather than the internal angle.



Hidden Line Contour with a Double Switching Point

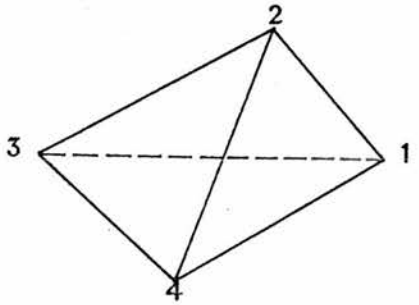
Figure 38

Though the definition of the hidden line contour was relatively easy, the subsequent conversion of this line to the shadow line contour did not appear to be a very efficient way of solving the hidden line problem for faceted objects. When it is realised that surface lines also have to be tested against the shadow line contour in such a procedure, it is clear that the overlay technique already described is the superior approach, and it does not rule out the definition and use of the hidden line contour, for example to emphasise the outline of an object.

What emerged from this investigation was the significance of marking re-entrant edges. The relevance of this idea became apparent when an attempt was made to include it in the volume matrix building operations which have already been discussed. If a convex solid is defined then all its vertices will be positively tagged; conversely, if the transpose volume matrix is created to define a void then all the vertex tags would have to be made negative. A more complex example of this idea in practice, is shown in Figure 39 where one volume is subtracted from another.

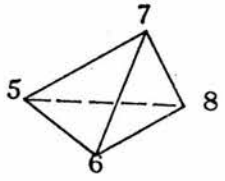
GRAPHIC MODELS

A	4	2	1
1	B	4	3
4	3	C	2
2	1	3	D

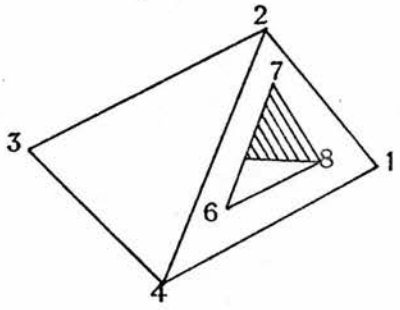


E	6	5	7
5	F	8	6
7	5	G	8
6	8	7	H

E	-5	-7	-6
-6	F	-5	-8
-5	-8	G	-7
-7	-6	-8	H



A	4	2	1	7	6	8
1	B	4	3			
4	3	C	2			
2	1	3	D			
6				E	-5	-7
8				-6	F	-5
7				-5	-8	G



Tagging Concave Surface Lines

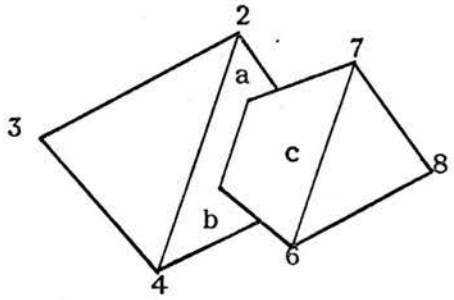
Figure 39

Where the angles between the faces adjacent to the edges 6/7, 7/8 and 8/6 must be less than 180^{o}, by definition, the complement must also be less than 180^{o}; consequently, these sides will be positive in the final matrix. The reverse situation is shown in Figure 40 where the new edges a/b, b/c and c/a are going to be re-entrant.

Other related situations which can give results automatically are shown in Figure 41.

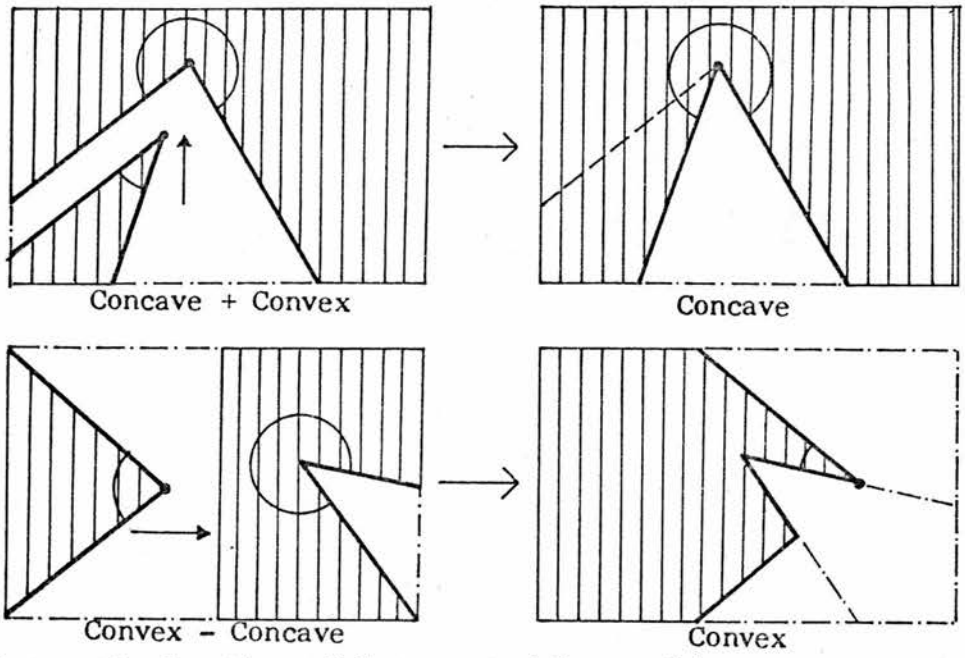
Not all the relationships between facets at edges are going to be given automatically, independent of the geometry of the objects. Where two volumes are merged at identical faces, then the outcome

A	4	2	1	-a	-b	-c	
1	B	4	3				
4	3	C	2				
2	1	3	D				
-b				E	6	a	7
-c				b	F	8	6
-a				7	c	G	8
				6	8	7	H



Tagging Convex Surface Lines

Figure 40



The Automatic Creation of Concave and Convex Edges

Figure 41

will depend on the shape of the original volumes. The link between this process and the next stage in the work, was the realisation that the boolean operations used in volume definition in the previous Chapter, contained much the same information that it was possible to include in these volume matrices by tagging vertex names. Planes

which were multiplied together gave convex surfaces whereas planes which were added together gave concave surfaces. The next step in the development was to look at the boolean form of volume description, which was part of the double data structure proposed in the previous Chapter, and see whether it could be usefully exploited in hidden line removal and other graphic output operations.

Two other sources contributed to the growth of this idea. The first was a paper by C. Jones,1971, 'A New Approach to The Hidden Line Problem', where the author discussed the division of the 'scene space' into convex zones. By creating a linkage tree between these spaces it was possible to rapidly select those cells which would contribute to the display. In a sense this was another way of representing and using the properties of scene coherence. Its immediate application in this work was in the representation of cellular structures, such as the rooms in a building, as volume matrices.

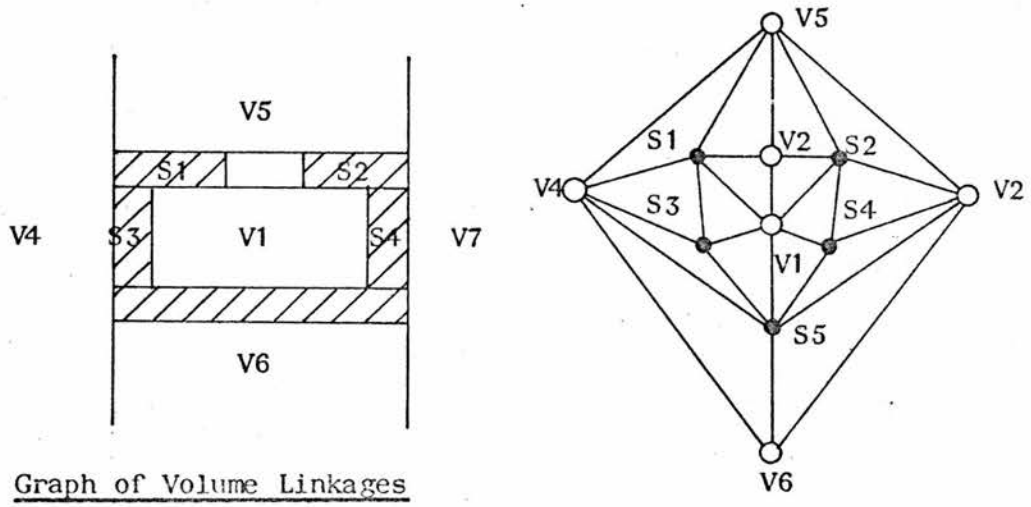
If the facets of a volume are named by linking together the two volume names on either side of the surface in question, in the same way that lines were named by the two zones they separated, then a volume matrix could take the form shown in Figure 42.

V0/V1	1	3	4
4	V0/V2	1	2
1	2	V0/V3	3
3	4	2	V0/V4

Volume Names Used in a Volume Matrix

Figure 42

Each facet occurs in two matrices when using this representation. The use of only solid surfaces, on the basis that the voids could be represented by the complement of these surfaces, is replaced by the inclusion of both solids and voids in the scene description. This makes it possible to create void descriptions that are convex as well as solid zones which are convex. If the names of these zones are used as the nodes in a graph, then given that the surface of one of the voids contains the viewing position, it becomes possible to separate those sub-volumes which it might be possible to see, from those it would be impossible to see. This linkage is carried out on the basis that a line of sight can only pass from a front face to a back face. In Figure 43 this means that if the viewing point lies in V6, then it will not be possible to see any part of V5, V2, or V1; but if the viewing point was in V5, then only V6 would be left untested.



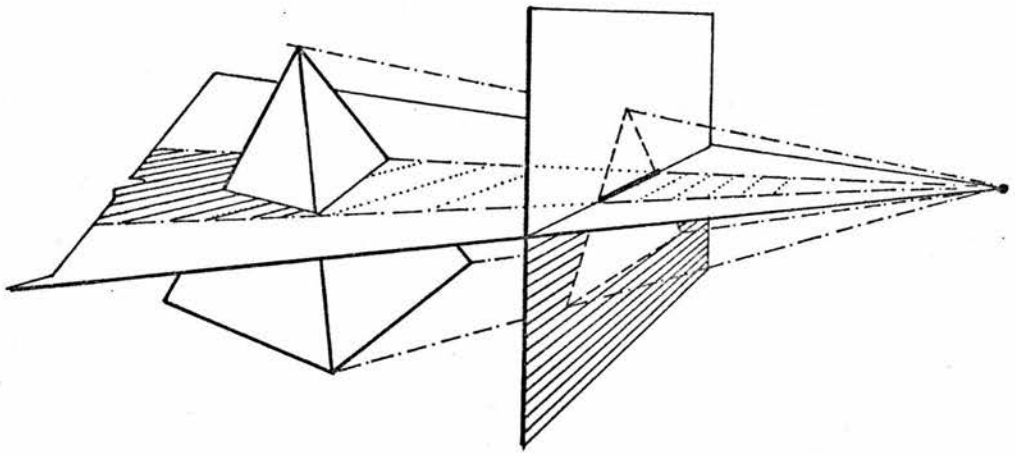
Graph of Volume Linkages

Figure 43

If the direct application for this technique was inside cellular structures, its extended application was with the subdivision of the external space around objects, into convex cells. When this idea was

linked with the use of boolean expressions for defining a scene, it was realised that the list of minterms for the solid regions would divide the solid into a convex cellular structure, while the minterms for the complement would define convex external cells. Not all the minterms would exist or, for that matter, would be needed but the basic idea seemed to provide a sound starting point.

The second source which contributed to the direct use of boolean expressions for creating displays, came from work done on hidden area removal algorithms. This approach to creating displays was based on the use of TV or Cathode Ray Tube rasters. The basic principle behind the algorithms used in this display procedure is illustrated in Figure 44.

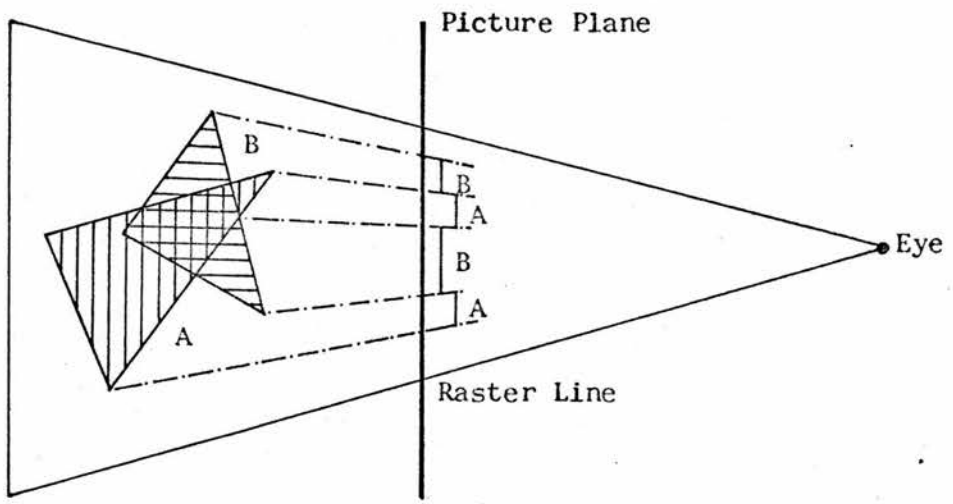


Projection of an Object on a Raster Line

Figure 44.

A plane is taken through the raster line, and the projection of the section cut through the object is represented in the raster line by intensity or colour changes. The effect of many raster lines is a

photographic shaded picture. Strictly speaking, the display is still a line drawing, but the overall effect is of an areal display. The nearest surface to the picture plane is again taken as the visible surface. Normally, the volume descriptions prepared for the display would be non-overlapping, but an interesting result is produced where object descriptions are made to overlap. If this occurred for line drawing data the result would be incorrect, always assuming that the program did not fail to run. In this case a correct and consistent picture is produced, and the result is the union of the two overlapping volumes. The reasons for this can be seen in Figure 45 , bearing in mind that it is always the nearest surface to the picture plane which is shown.

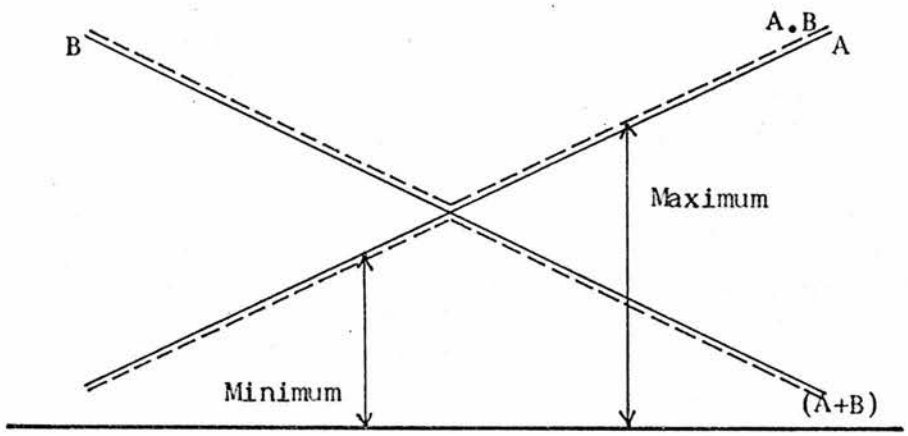


Section through a Raster Line

Figure 45

In this procedure, finding the minimum distance from the picture plane corresponds with the display of the union of two surfaces. In 'Boolean Systems' by D. Kaye, 1968 the author defines the boolean operators for points in the Cartesian Plane such that the sum of two points gives

the point furthest from the origin, while the product of two points gives the point nearest the origin. If the reverse interpretation of the operators is made in terms of maximum and minimum distances to the picture plane it is possible to create a display of the intersection of two planes as shown in Figure 46 .

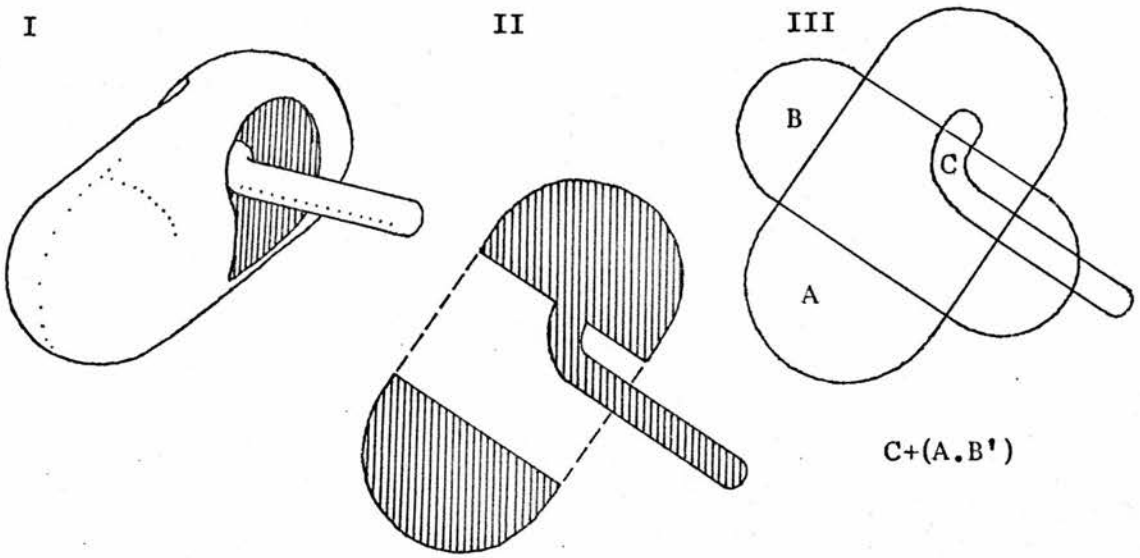


Boolean Operations for Visibility

Figure 46

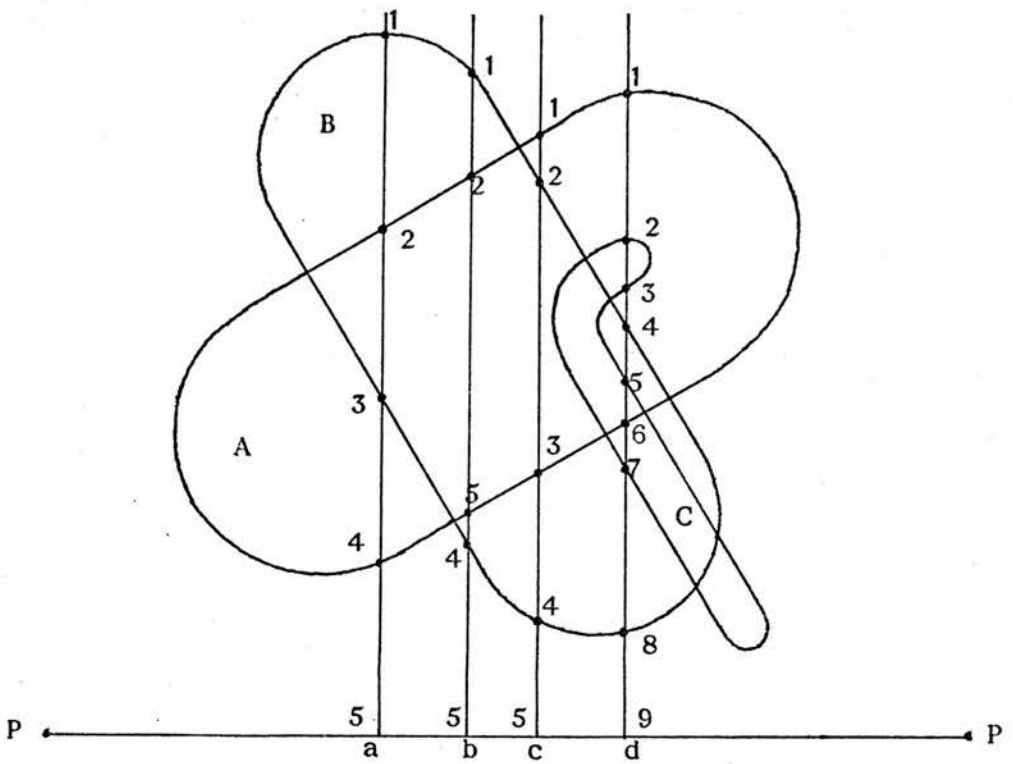
This concept was initially experimented with, using the object illustrated in diagram I in Figure 47, a section through this object is shown in the centre diagram, and a possible volume definition as a boolean expression is presented in diagram III. The simple line scan method of creating a display would break down if it were applied to this description.

The first stage was to consider the different results which would be obtained from viewing lines cutting this object in different relationships to the three surfaces.



Experimental Object for Display

Figure 47



Distances of Surfaces for Different Raster Points

Figure 48

The first distinction which had to be made was between an intersection with a back surface and an intersection with a front surface. In the

following attempt to use the boolean expression as a way of selecting the visible surface the front intersections are considered positive and the rear intersections negative. The first apparently successful rules are shown applied to the four raster points a, b, c, and d from Figure 48 , in the tables given below.

The procedure was to enter values of the intersection points into the boolean expression in the order in which they were found, moving back from the picture plane. The first value to be added to the expression which created an output for the whole expression was considered to belong to the visible surface. To start the operation it was necessary to prime the variables associated with each of the surfaces. If the surfaces were 'solid' then this value was 0, while when the surfaces were of voids this value was 00.

Raster Point a

	C	+	(A	.	'B)	=	V	
	0	0	0	0	00	=	0	
A:4	0	0	4	0	00	=	0	4 + 0 → 4
	0	0	4	4	00	=	0	4 00 → 4
	0	4	4	4	00	=	4	4 + 0 → 4

The Visible Surface at a is A

Raster Point b

	C	+	(A	.	'B)	=	V	
	0	0	0	0	00	=	0	
B:-4	0	0	0	0	00	=	0	-4 + 00 → 0
A:3	0	0	3	0	0	=	0	3 + 0 → 3
A:-2	0	0	0	0	0	=	0	-2 + 3 → 0
B:1	0	0	0	0	1	=	0	1 + 0 → 0

No Surface is Visible at b

Raster Point c

	C	+	(A	.	'B)	=	V
	0	0	0	0	00	=	0
B:-4	0	0	0	0	00	=	0
A:3	0	0	3	0	0	=	0
B:2	0	2	3	2	2	=	2

The Visible Surface at c is B

Raster Point d

	C	+	(A	.	'B)	=	V
	0	0	0	0	00	=	0
B:-8	0	0	0	0	0	=	0
A:+7	0	0	7	0	0	=	0
C:+6	6	6	7	0	0	=	6

The Visible Surface at d is C

This example was limited in size and complexity. Difficulty was found in extending it to cover expressions which had more than one level of nesting, for example:

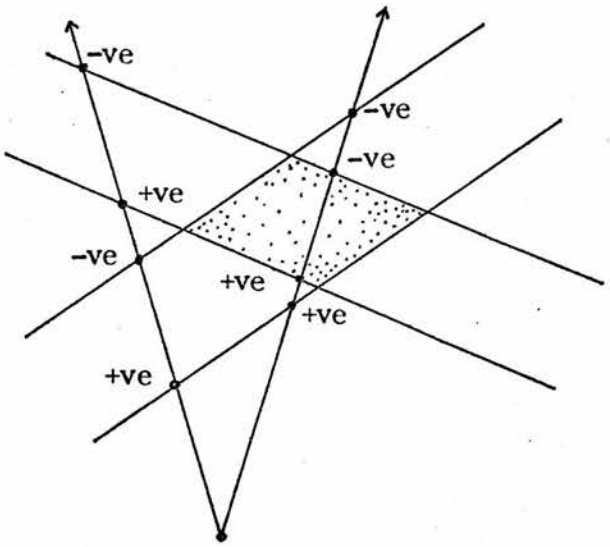
V+(B.C.(A+D))

However, it has already been shown that this need not be a handicap since an expression of the form shown above can be automatically expanded to give:

V + B.C.A + B.C.D

In the example shown in Figure 48 the volumes were enclosed by a single named surface. The next stage was to extend or modify the rules so that volumes made up from intersecting planes could be displayed in this way. Consider the section through the volume

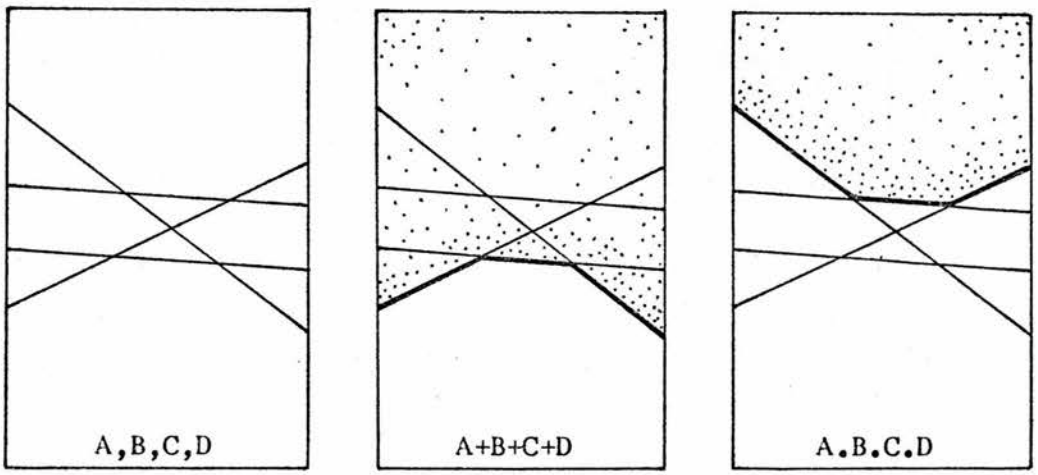
shown in Figure 49.



Surface Visibility for Volume A.B.C.D

Figure 49

It is simple to obtain the visible plane from a concave surface, and the visible plane from a convex surface as can be seen from Figure 50.



Displaying Convex and Concave Surfaces

Figure 50

It can be seen from the geometry of the plane surface that following the intersection points of the viewing ray and the surface which gives the minimum distance to the viewing point will define the concave surface, and conversely following the intersection points which give the maximum distance to the viewing point will define the convex surface. The difficulty is the situation shown in the previous diagram in Figure 49 , where the viewing ray passes outside the object. In the original example the problem was avoided since the viewing ray simply did not intersect any of the objects surfaces. In this case all the plane surfaces will cut the viewing ray even though no surface will be visible. The solution to this problem was found by separating the backface intersections from the front face intersections. The front values are treated as described above for convex surfaces, whereas the backfaces are treated as though they were forming a concave surface. As long as the concave surface lay behind the convex surface then the object was visible. Once the rear surface crossed in front of the front surface the object ceased to exist.

Summarising the rules for displaying an object defined as a boolean expression, where back faces are defined as giving negative distances to the viewing point and front faces as giving position distances:

For each Convex Object Given by a Boolean Product Phrase

1. From the positive distances select the greatest, for example 'x'.
2. From the negative distances select the largest, algebraically, for example 'y'.
3. If $-y < x$, then the viewing ray does not strike the object.

4. If $-y > x$, then the visible surface is X, or x is passed to next stage where there is more than one convex object.

For a Concave Surface or a Collection of Convex Objects

5. From the positive distances select the smallest, say 'z'.
6. Z is the visible surface.

These rules are more convenient than the previous rules because the distances to planes can be processed in any order, as long as the product phrases are evaluated before the overall summation. This meant that a program could be written which processed the boolean expression sequentially creating running maximum and minimum values from which the visible surface eventually emerged.

Once it was possible to create a display from the boolean expression the next task was to extend the input language described in the previous Chapter, to include a display command. The form of this command is given in Figure 51.

DISPLAY:S,2,0,6,2,0,0,7,0,5,4,7,2,500,500;
a b c d

a: Name of Volume File to be Displayed

b: Definition of Picture Frame

c: Definition of Viewing Position

d: Raster Resolution

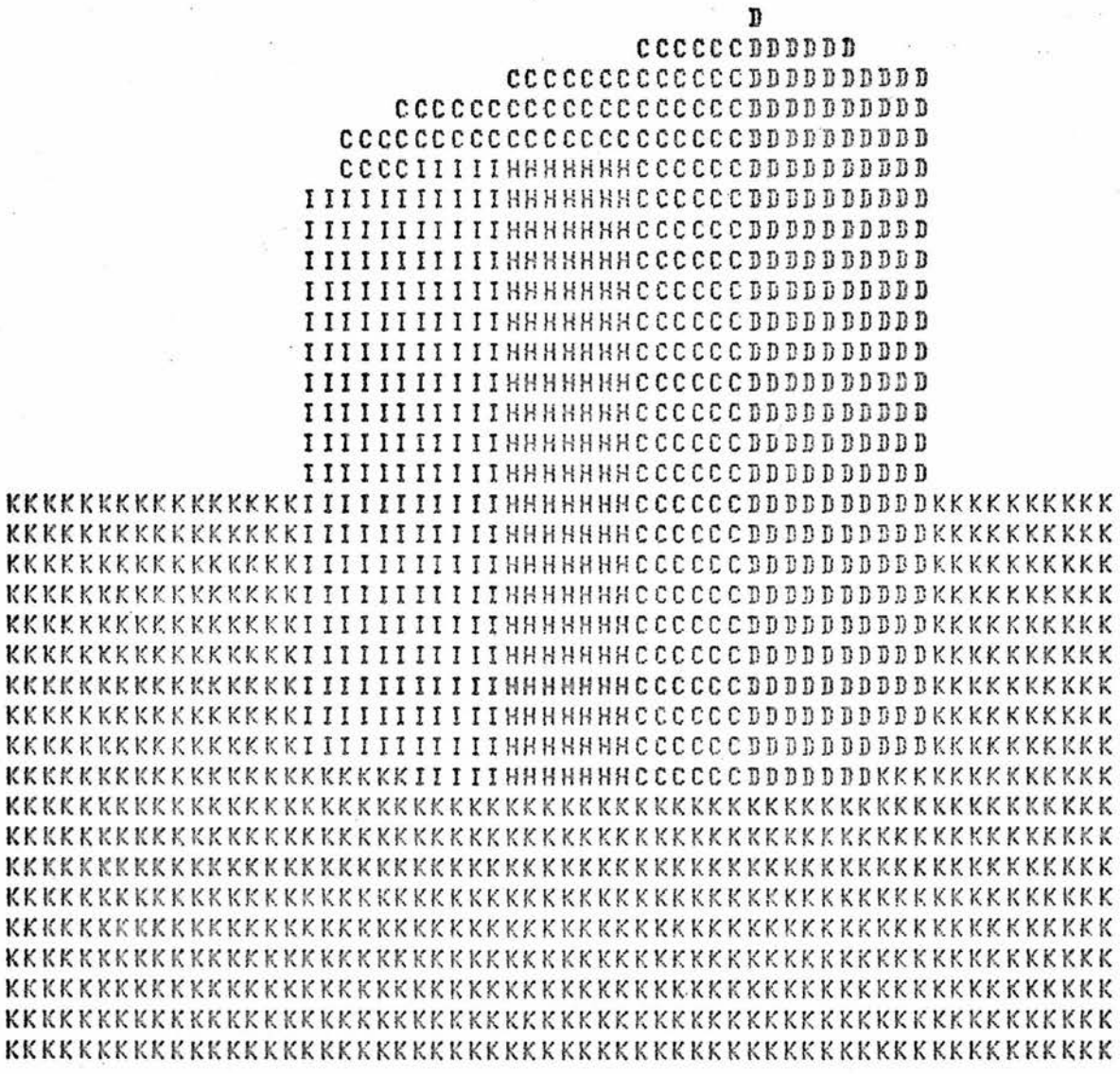
Display Command

Figure 51.

GRAPHIC MODELS

```
M = A.B.C.D.R + E.F.H.I.J + G;

DISPLAY: M, 2, 0, 6, 2, 0, 0, 7, 0, 5, 4, -7, 2, 40, 60;
```



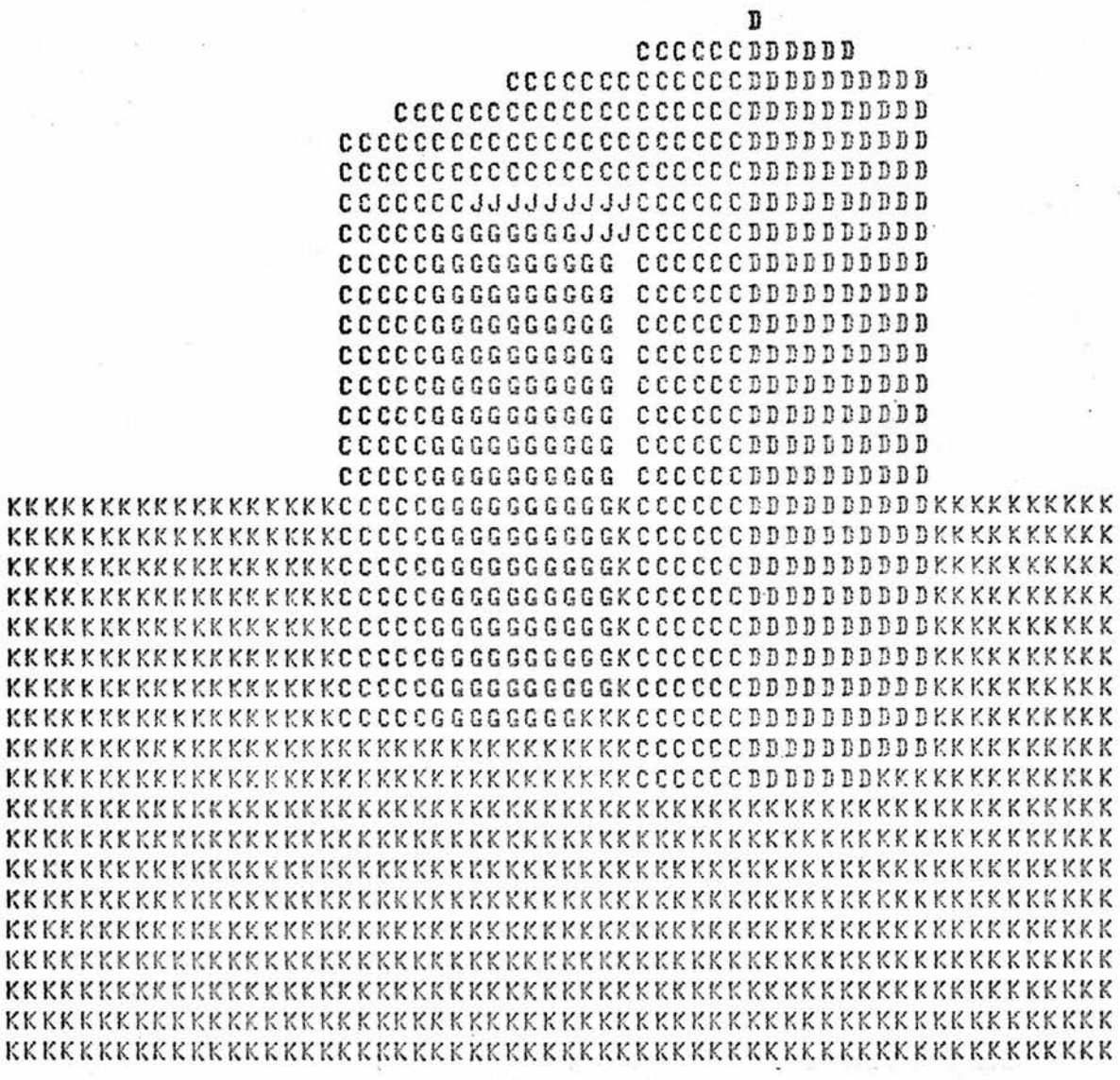
Large Box plus Small Box plus Ground Plane

Figure 52.

GRAPHIC MODELS

```
M = A.B.C.D.R.'(E.F.H.I.J) + G;

DISPLAY: M, 2, 0, 6, 2, 0, 0, 7, 0, 5, 4, -7, 2, 40, 60;
```



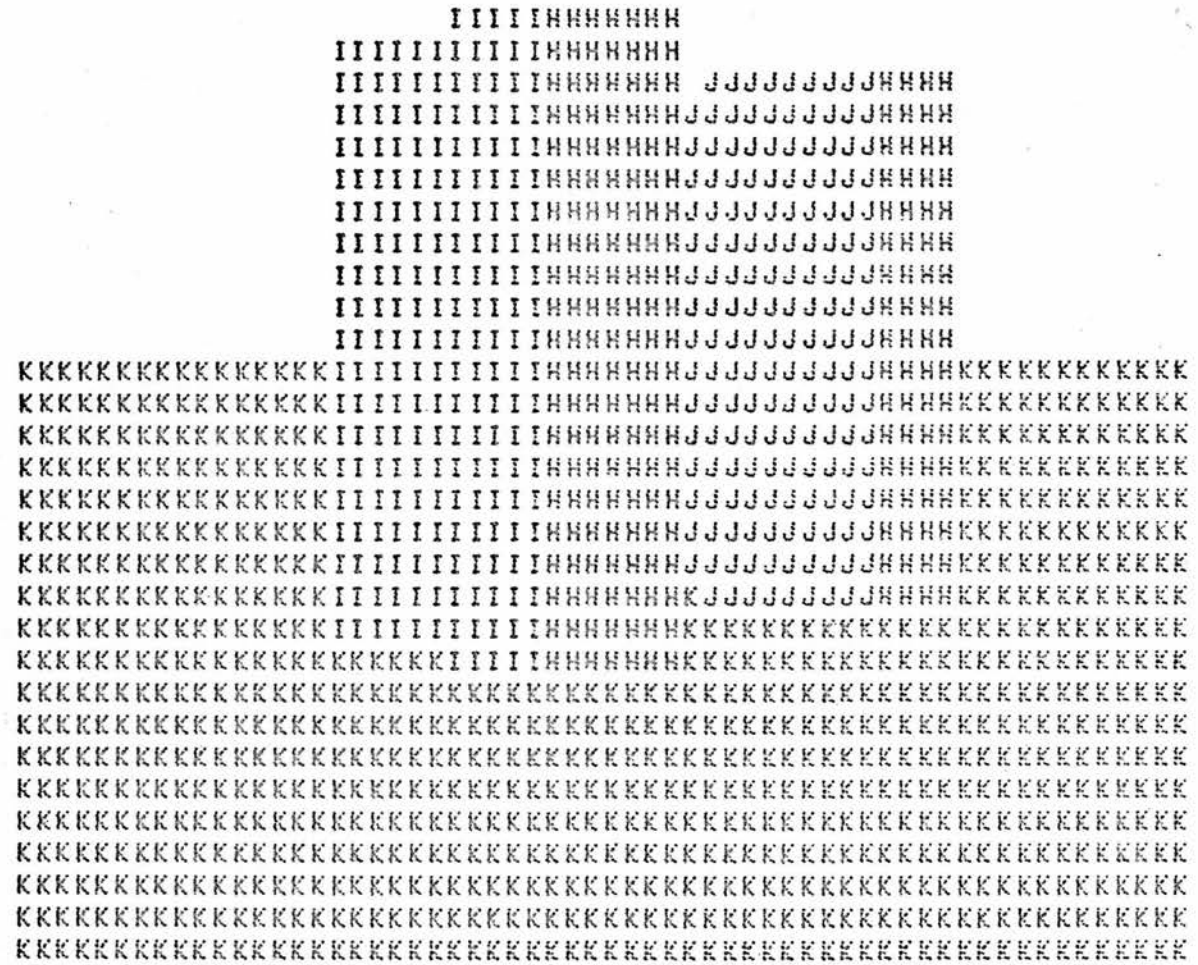
Large Box minus Small Box plus Ground Plane

Figure 53.

GRAPHIC MODELS

```
M = E.F.I.H.J .'(A.B.C.D.R) + G;

DISPLAY: M,2,0,6,2,0,0,7,0,5,4,-7,2,40,60;
```



Small Box minus Large Box plus Ground Plane

Figure 54.

Output created by this process can be seen in Figures 52, 53 and 54. In the first the volume is defined as the union of two boxes and a ground plane. In the second the smaller block is subtracted from the larger and the result added to the ground plane. These three displays were compiled from the same collection of plane definitions which were entered once at the beginning of the program run.[1]

Conclusions

In this Chapter two basic procedures for creating graphic output from the volume descriptions analysed in earlier Chapters have been discussed. Though the range of graphic effects which could be employed in displays was briefly examined, only a few of them have been used in these studies. In the case of line drawings some of the other effects have been discussed in the Chapter on the OBLIX program. In the case of the raster scan displays there is a much wider scope for further development, since only the basic display mechanism has so far been considered. The problem with this process is that so far only its software implementation has been attempted, and this is very slow and impractical to develop further in this form. The advantage which the process potentially offers is the facility to create displays directly from a volume data structure which can also act as a convenient input language. For this reason the basic ideas examined in this Chapter are extended in the next Section, and are taken as the starting point for a hardware processor which might make the operation more practical.

1. See Appendix C.

SECTION IV

HARDWARE POSSIBILITIES

CHAPTER 13

HARDWARE DEVELOPMENT: A SCHEMATIC MACHINE

HARDWARE DEVELOPMENT: A SCHEMATIC MACHINE

In the previous section most of the work was orientated towards the development of a software system. Though on one hand this meant starting with primitive operations established by other uses, it provided on the other, a way of linking new operations to many general purpose procedures which already existed. From this study several general points emerged. Whether the data were in the complex forms used to represent surfaces in the design of aircraft or ships, or the relatively simpler forms employed in map making, it was possible to define two overall classes of use. The first consisted of the analysis of spatial properties which only required output in a printed or language form. Properties such as surface areas, moments of inertia, the number of people living in a particular geographical zone, or even the construction cost for a design scheme could be calculated and printed out without including the spatial descriptions used. This meant that the spatial descriptions could be minimised and tailored to suit these specific internal uses. In contrast the second form of analysis demanded some form of graphic output in order to provide the system user with the information which he required, and this restricted the spatial models used to those which could also be used to create a display file.

The system had to provide the user with a reasonably simple input procedure. If data preparation was too complicated or time consuming it would tend to destroy the advantages of automation. The more common place the activity being replaced the more this was likely to be true. Where the control language related intuitively to the operations being carried out, the simpler and more error free the use of the system could

be. When using spatial data, this ease of use depends very heavily on being able to display the information in a graphic form. However, it was found that the input and output procedures were governed by primitive operations using point coordinates. Though for many applications these were suitable, it was shown in the last section that the consequence of using this form of representation for zones in space was that it was difficult to determine zones of overlap by any simple procedure. For geographical data the overlap of different properties in space is expected. In physical design work, where these properties are solid and void, the interpenetration of two solid objects is an error, and the fact that it can happen by accident is a major problem. In both cases it is necessary and important to define the regions of spatial overlap, and it is an operation which occurs frequently, and should therefore be implemented in the fastest form possible.

Boolean expressions were found to provide a relatively simple way of expressing zones of overlap, and they could also be used as a convenient way of entering spatial descriptions. However, it was a difficult matter to visualise what a particular expression meant without a display, though given a display it was easy to modify the spatial relationships shown using these expressions. The problem which this posed was that, if a language based on the boolean expressions were used for data input, the creation of a display using point based primitives required complex and extensive calculations. In the previous chapter it was shown that a spatial description existing as a boolean expression could be directly interpreted into a picture if new primitive operations could be provided in a display processor. The implications of providing such facilities at a hardware level are discussed more

fully in this section.

The first stage is to summarise the basic ideas which underlie this approach, and this can be done in the following way. The primitive interpolation operation which is implicitly applied to a stream of point coordinates to define a polygon boundary is replaced by a primitive overlap operation implicit in the use of boolean expressions.

The relationship between a point coordinate and a particular line co-ordinate is given by the expression:

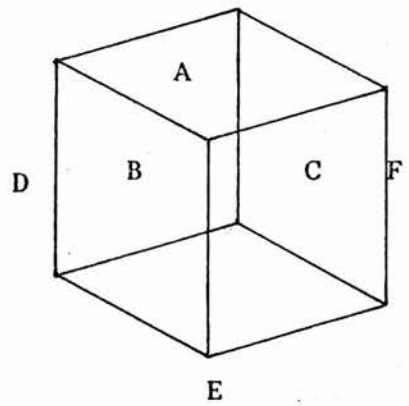
$$k = a.x + b.y + c$$

and similarly in the case of plane coordinates in three dimensional spaces by the expression:

$$k = a.x + b.y + c.z + d$$

Where a point lies on one side of the boundary defined by the line or plane then the value of k is positive, when it lies on the other side of the boundary then k is negative.

The wireframe drawing for the object A.B.C.D.E.F could thus take the form shown in Figure 1.



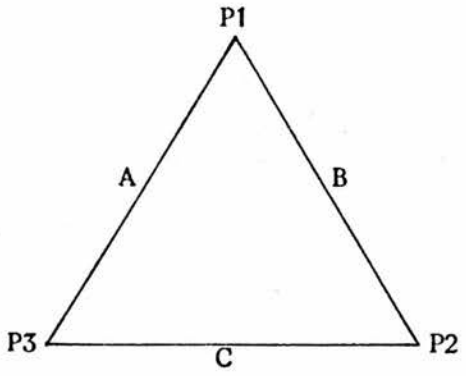
A.B.C.D.E.F

A Cube

Figure 1

In two dimensions the triangle shown in Figure 2 would be expressed as A.B.C.

HARDWARE DEVELOPMENT: A SCHEMATIC MACHINE

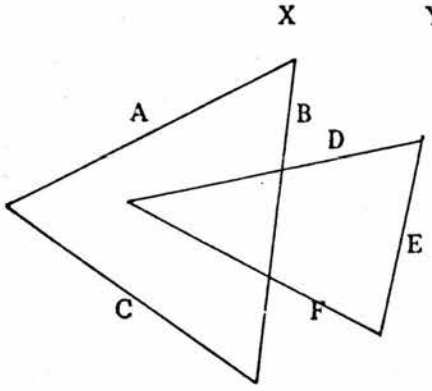


A.B.C P1 P2 P3

A Triangle

Figure 2

The overlap operation can be used to create new objects from previously defined objects as shown in Figure 3 where new zones can be defined as the result of different interactions between two overlapping triangles.



$X.Y = (A.B.C).(D.E.F)$

$X+Y = (A.B.C)+(D.E.F)$

$X.\overline{Y} = (A.B.C).\overline{(D.E.F)}$
$= (A.B.C.(\overline{D}+\overline{E}+\overline{F}))$

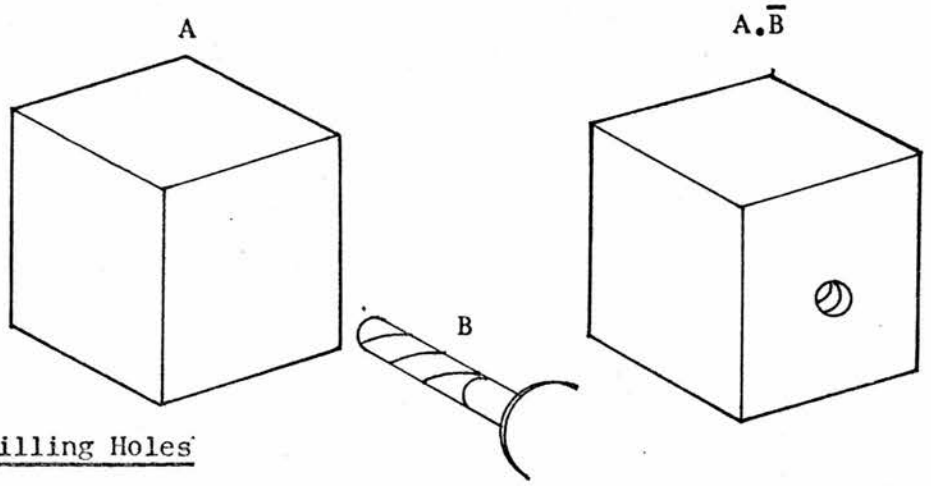
Complex Objects

Figure 3

This process is simple in the sense that it intuitively relates to the idea of adding and subtracting volumes as it occurs in practical construction and cutting operations. The manual process of drilling a hole shown in Figure 4, can be expressed as the boolean subtraction of two volumes.

The relationship between points and complex objects can be determined by a form of point in polygon procedure shown diagrammatically in

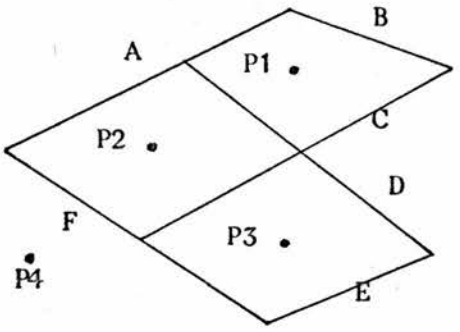
HARDWARE DEVELOPMENT: A SCHEMATIC MACHINE



Drilling Holes

Figure 4

Figure 5 .



Point in Polygon

Figure 5

	A.B.(C+D).E.F	
P1	T.T.(T+F).T.T	True
P2	T.T.(T+T).T.T	True
P3	T.T.(F+T).T.T	True
P4	T.T.(T+T).T.F	False

T: Inside is True
F: Inside is False

The polygon can be defined by the expression: A.B.(C+D).E.F
This expression can be expanded to give:

A.B.C.E.F + A.B.D.E.F

The two product phrases correspond to the two convex, overlapping polygons shown in Figure 5 . If for each point p_n the value:

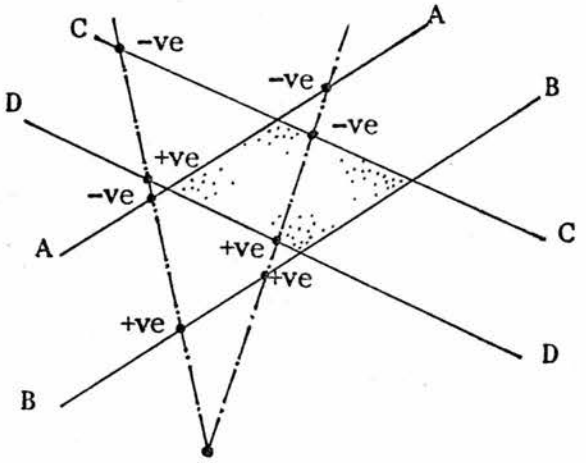
$$k_i^n = a_i \cdot x_n + b_i \cdot y_n + c_i$$

is evaluated for each line L_i, then the sign of k for each side of a convex polygon must be positive for the point to lie inside the polygon. For a polygon made up of several overlapping convex polygons the point

must lie within at least one, to be within the whole polygon. This operation can be applied in exactly the same form to polyhedra to give a point in volume text.

It was shown in the previous section that it is possible to define a more complex operation using boolean expressions for creating perspective displays of three dimensional scenes, with hidden areas automatically removed. The distances from the picture plane at which a particular viewing ray cuts the planes making up the volumes in a scene can be used with the structure of the boolean expression to indicate the visible plane at the point where the viewing ray pierces the picture plane.

Consider the two viewing rays R1 and R2 in relation to the convex volume A.B.C.D shown in Figure 6.



Conditions for Surface Visibility for a Convex Volume

Figure 6.

Where the distance from the picture plane to the plane A is a, and from B is b, etc., and where, if the viewing point lies inside the plane, the distance is considered to be a negative number whereas if the viewing

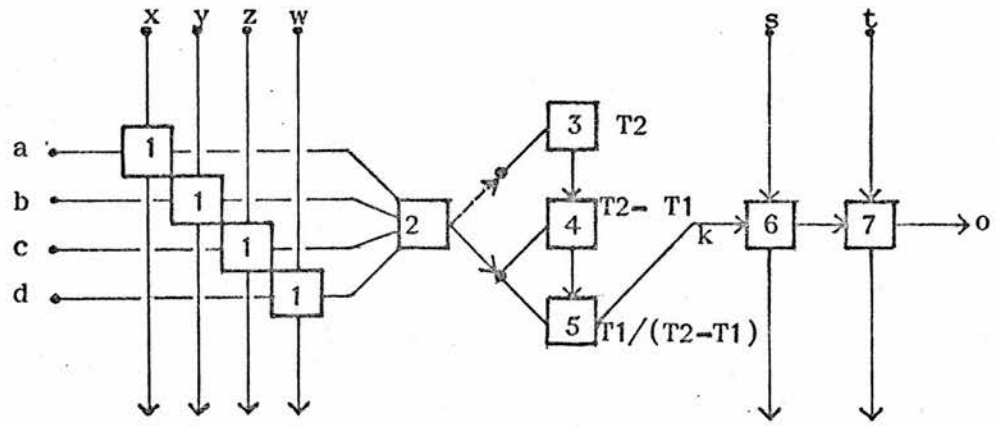
point lies outside the plane the distance is regarded as a positive number, then the following relationships define the visible plane:

> From the positive distances select the greatest, for example 'x';
>
> From the negative distances select the largest algebraically, for example 'y'.
>
> Then if $-y < x$ the viewing ray does not pass through the object
>
> If $-y > x$ then the visible ray is X.

Where more than one convex volume is potentially visible at a particular point on the picture plane then the surface which is at a minimum distance from the picture plane along the viewing ray will be the visible surface.

Each of these procedures has been simulated in software, and with the exception of the point in volume test can be carried out faster by methods using point-based data structures. The only advantage appears to be the convenience of the operating language. However, this approach provides a consistent approach to each of these operations, and though slow each of these procedures is remarkably simple. It was this fact and the independence of a large part of the individual processing tasks which suggested the possibility of using a parallel processor. Such a processor would consist of a series of modular units all capable of carrying out the same tasks. The basic scheme for one such module is shown in Figure 7 . The single unit is capable of carrying out the sequential process which has just been described. The more of these units included in a processor the faster the overall operation becomes.

Figure 7 shows the components of a machine which will carry out the



Basic Module: Parallel Processor

Figure 7

necessary operations for creating a raster scan display, and was developed from the procedure described in Chapter 12. These components are:

(1) Multiplier

(2) Adder

(3) Latch

(4) Adder

(5) Divider

(6) First Stage Selection

(7) Second Stage Selection

Creating a display is the most complicated of the series of operations which has been described. The other processes can be carried out using only a subset of this list of components. At this schematic stage of development it appeared possible using a machine made up from these units acting in parallel to carry out the following list of processes:

HARDWARE DEVELOPMENT: A SCHEMATIC MACHINE

(a) Matrix Multiplication

(b) Sort a List of Numbers into Order

(c) Point in Polygon Testing

(d) Point in Polyhedron Testing

(e) Raster Scan Display of Polygons

(f) Line shaded Polygon Displays

(g) Nearest Neighbour Polygon Display

(h) Shaded Perspective Display of Volumes

(i) Line Drawing Display of Volumes

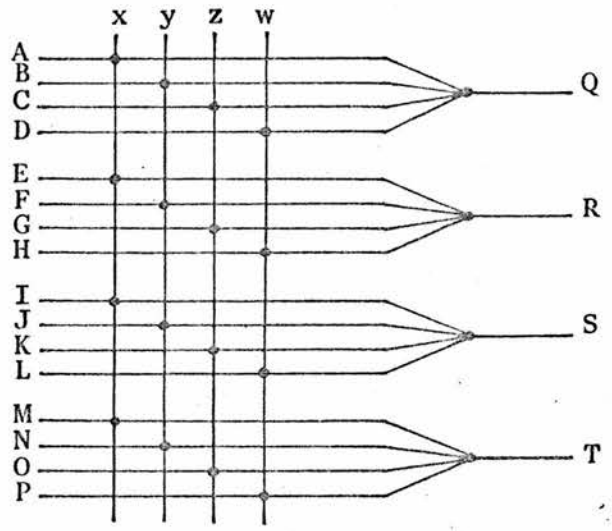
(j) Plan Section Displays of Volumes

The first stage in the investigation of this schematic machine was to examine the problems associated with the implementation of all these operations in one device.

Matrix Multiplication

Given an array of multipliers operating in parallel, matrix multiplication is merely a matter of supplying the correct values to the appropriate multiplier and then adding together the results to give the elements in the new matrix. An arrangement which will carry out the matrix multiplication necessary to transform point coordinates and plane coordinates is shown in Figure 8 . This is sufficient for scaling, translating, rotating, and creating perspective projections. It can be applied to data structures using both point coordinates and plane coordinates.

Where multiple transformations are required it is often necessary to create the resultant single transformation by multiplying together the



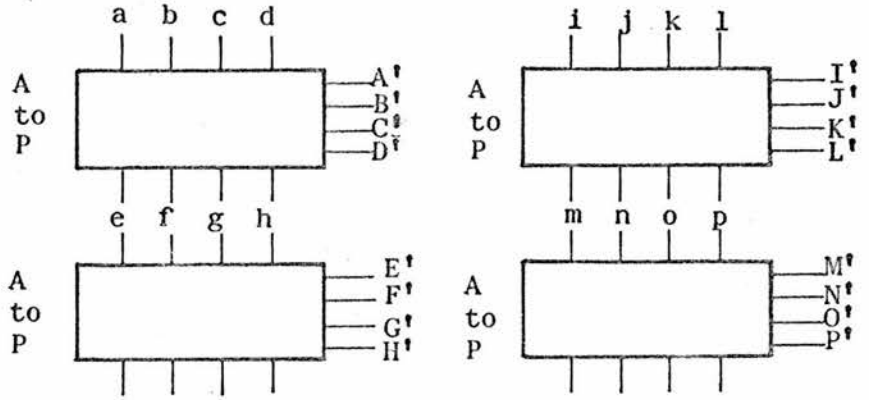
$$\begin{bmatrix} A,B,C,D \\ E.F.G.H \\ I.J.K.L \\ M.N.O.P \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = \begin{bmatrix} Q \\ R \\ S \\ T \end{bmatrix}$$

Matrix Multiplication for Coordinate Transformations

Figure 8

matrices representing the simple component transformations. This requires a more complex arrangement which is in effect four groups of the configuration shown in the previous case. Different values have to be introduced into the vertical lines for each group.

Only elements 1 and 2 in Figure 7 are used in this operation. This means that some mechanism for extracting the final results from between elements 2 and 3 will have to be included. It also means that the control structure must include appropriate switching arrangements to isolate the operation of the first two elements from the rest of the unit.



$$\begin{bmatrix} A,B,C,D \\ E,F,G,H \\ I,J,K,L \\ M,N,O.P \end{bmatrix} \cdot \begin{bmatrix} a.e.i.m \\ b,f,j,n \\ c,g,k,o \\ d,h,l,p \end{bmatrix} = \begin{bmatrix} A',B',C',D' \\ E',F',G',H' \\ I',J',K',L' \\ M',N',O',P' \end{bmatrix}$$

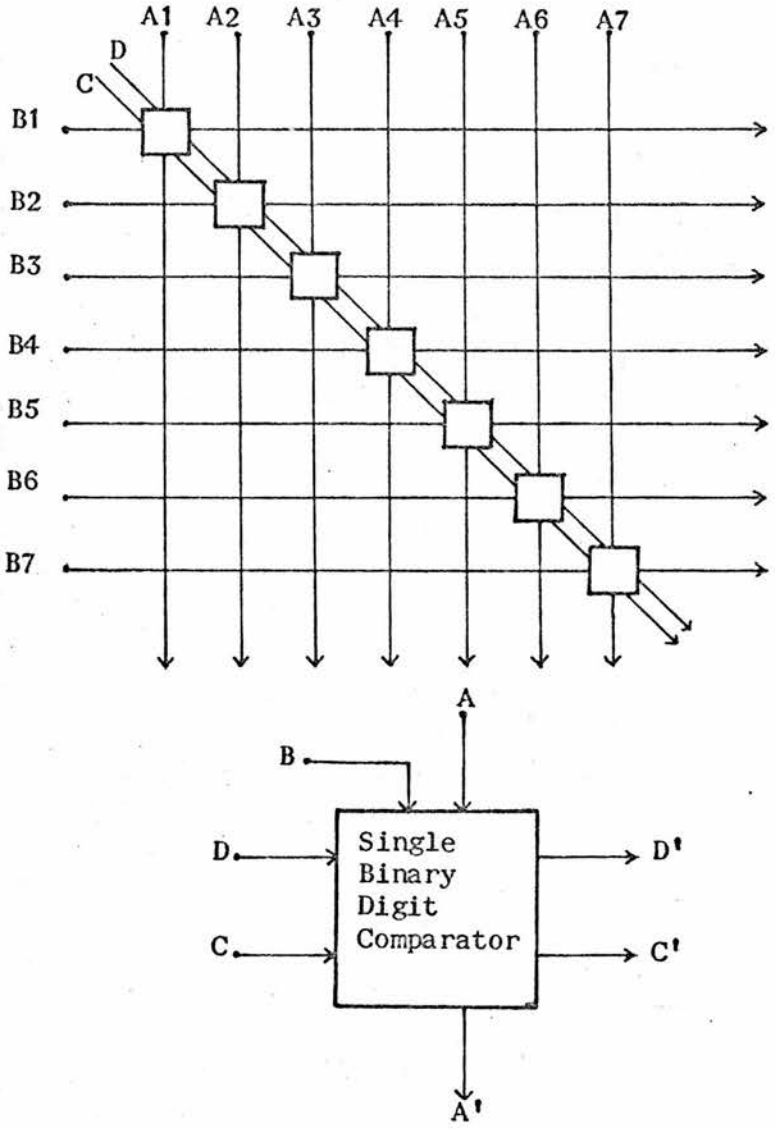
Matrix Multiplication for Multiple Transformations

Figure 9

Sorting a List of Numbers into Order

The second operation which these units could be used for was to order a list of numbers into either ascending or descending order. The basic operation is to locate the largest or the smallest number and pass it to the output device. This means that if this operation is repeated, then in 'n' steps a list of 'n' numbers will be output in their natural order. The original concept was for element 6 to receive two digital signals, one horizontally from the unit in which it belongs and the second vertically, from the units above it. The function of element 6 was to pass down to units below it the larger of the two numbers it received as inputs. This element also had to produce a logic signal to indicate

whether the signal which it received from its own unit was the largest number so far encountered. A schematic structure for an element which could perform this operation is shown in Figure 10.



Comparison Element for Sorting Numbers

Figure 10

In Figure 10 'A' represents the largest number so far found by other

units passed down from above, and B represents the number entering from the present unit. If the two numbers A and B are processed from left to right, then leading zeros can be ignored: the first number to provide a 1 being the largest number. If there is a tie, then the first number to have a 1 where the other number has a $\emptyset$ will be the larger number. Until the superiority of one of the inputs has been established, the output from each binary digit unit can equal either of them since they will be identical. Once the decision has been made, then whatever the relationship established by subsequent bit comparisons, the output must be from the larger number. This requires a left to right communication between the digit comparison cells. The minimum linkage seemed to require at least two lines, C and D in Figure 10.

As a design exercise the single cell used to compare two digits was analysed to establish a possible configuration of and or or gates which could be used to carry out this selection process. A Karnaugh map showing the structure of such a circuit is given in Figure 11.

A'

AB \ CD	00	01	11	10
00	0	0	0	0
01	1	0	1	1
11	1	1	1	1
10	1	1	0	1

C'

AB \ CD	00	01	11	10
00	0	0	1	.
01	1	0	1	1
11	.	0	1	.
10	0	0	1	0

D'

AB \ CD	00	01	11	10
00	0	1	1	0
01	1	1	1	1
11	0	1	1	0
10	1	1	1	1

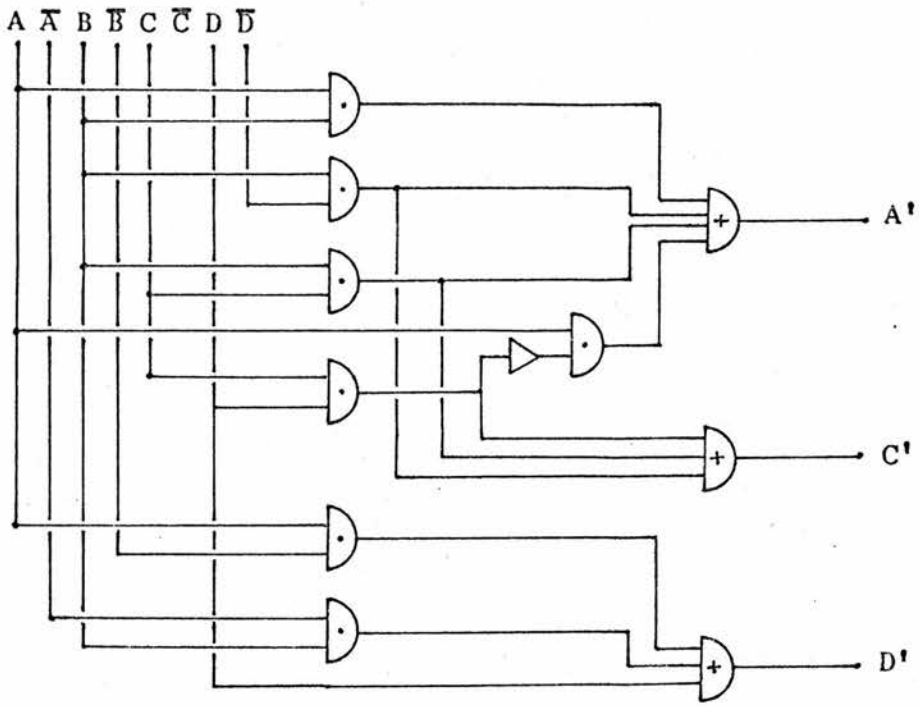
$A' = B.\overline{D} + B.C + A.\overline{C} + A.\overline{D} + A.B$

$C' = B.\overline{D} + C.D + B.C$

$D' = A.\overline{B} + \overline{A}.B + D$

Karnaugh Map for Single Digit Comparator

Figure 11.



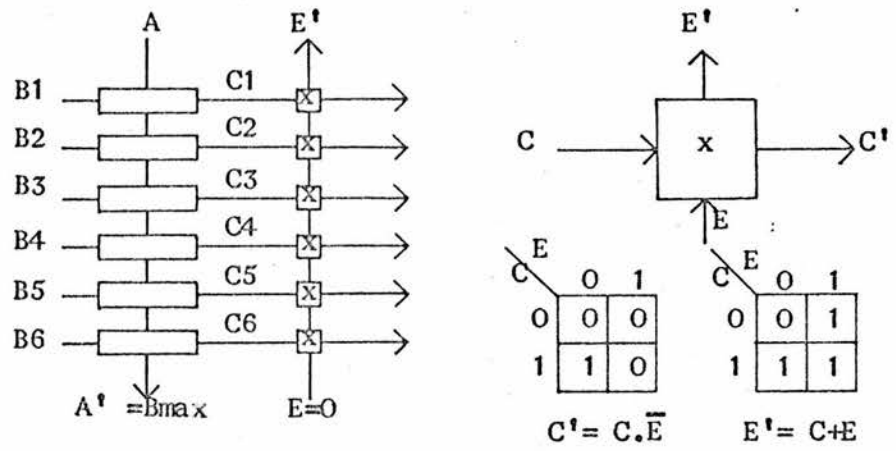
A Possible Circuit for the Comparator Cell

Figure 12

Figure 12 shows a possible implementation of a circuit which will carry out the operation defined in the Karnaugh Map in Figure 11.

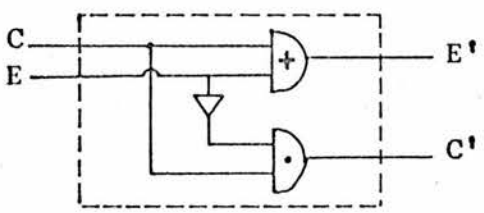
If a list of values B1, B2, B3, ... is arranged in ascending order, so that B1 < B2 < B3 ... then though the correct maximum value will be output at the bottom of the list as A', all the logic signals C' will be indicating maximum values.

This makes it necessary to introduce a further element which corrects these outputs when a new maximum supersedes an existing one. One approach to this solution is shown in Figure 12, where an element X receives a signal from below if a new maximum is found. The circuit for X is shown in Figure 14.



Locating the Register Containing the Maximum Value

Figure 13



Unit for Indicating the Maximum Value

Figure 14

The element which selects the largest of a list of values must be designed to handle negative as well as positive integers. This depends on the representation used for negative numbers. The original choice because of the selection process needed in the display operation was to use a sign bit and a positive value. An alternative is to use a two's complement representation which simplifies certain arithmetic operations.

+5	0 0000101	positive number
-5	1 0000101	negative number
-5	1 1111011	negative 2's complement

Using the first representation for negative numbers meant that the basic cell in the comparison element had to be modified so that when it was presented with two positive numbers it selected the larger, when presented with a positive and a negative number it selected the positive number, and when presented with two negative numbers it selected the one nearest zero. A comparison element which will carry out this function is defined by the Karnaugh Maps in Figure 15
A sign bit comparison element must be employed with this unit to initialise the values of C and D passed to the first digit comparison elements.

A'

AB \ CD	00	01	11	10
00	0	0	0	0
01	1	0	1	0
11	1	1	1	1
10	1	1	0	0

C'

AB \ CD	00	01	11	10
00	.	0	1	.
01	0	0	1	1
11	.	0	1	.
10	1	0	1	0

D'

AB \ CD	00	01	11	10
00	0	1	1	0
01	1	1	1	1
11	0	1	1	0
10	1	1	1	1

$$A' = B.\overline{C}.\overline{D} + C.A + A.B + B.C.D$$
$$C' = \overline{C}.\overline{D}.\overline{B} + B.C + C.D$$
$$D' = A.\overline{B} + \overline{A}.B + D$$

Comparison of Both Positive and Negative Numbers

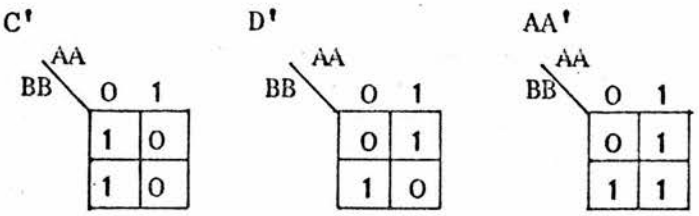
Figure 15

To enable the element defined in Figure 15 to operate correctly the

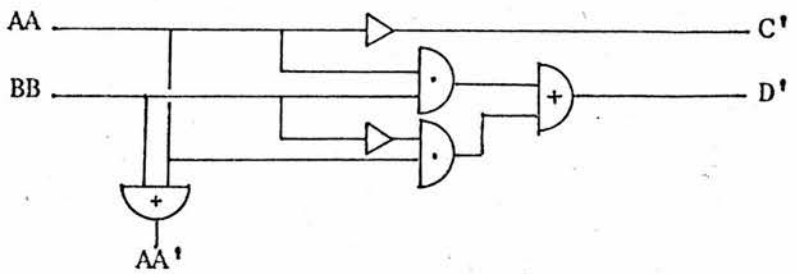
values of C and D have to be initialised by comparing the sign bits of the two numbers being compared. The combination of values which C and D can be given are:

D=0	C=1	Allow the smallest number to pass through
	C=0	Allow the largest number to pass through
D=1	C=1	Allow B to pass through
	C=0	Allow A to pass through

A sign bit comparison element which will set these values correctly is shown in Figure 16.



$AA' = BB + AA; \quad C' = \overline{AA}; \quad D' = BB.\overline{AA} + \overline{BB}.AA$



Sign Bit Comparison

Figure 16.

With this arrangement it was found to be possible, by changing the way in which C and D depended on the sign bits to make the circuit represented in Figure 15 select either the largest number or the smallest number. If an extra input E is provided to the sign bit

Comparison Unit

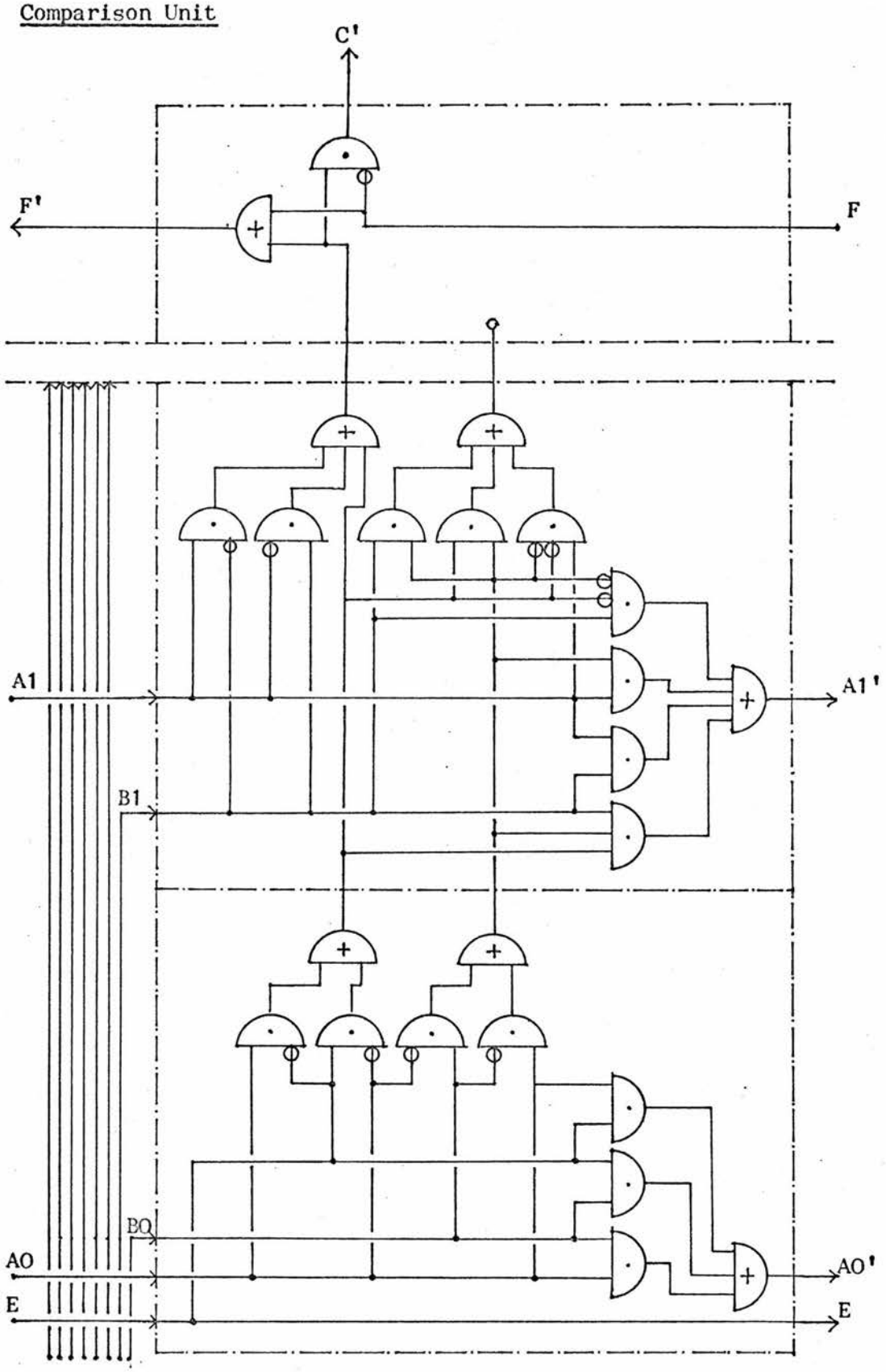


Figure 17

comparison element then when E = 1 the maximum value will be found, while when E = 0 the minimum value will be found.

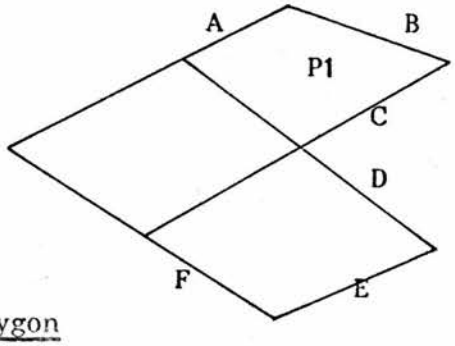
Though this arrangement can produce the ordered list of values, which was the original aim, it can be seen that this circuit is not acting in parallel in the sense which was originally intended. A better solution to this part of the machine was found later when the display operation was considered in more detail.

Point in Polygon and Point in Polyhedron Testing

It was possible to consider Point in Polygon testing and Point in Polyhedron testing as the same process at this stage in the analysis. The first step in this process was to calculate a value k for each plane, where

$$k = a.x + b.y + c.z + d$$

and this could be done using elements 1 and 2 in each unit. The second step of the operation depends on the sign of the values of k for the various planes. Consider the polygon shown in Figure 18.



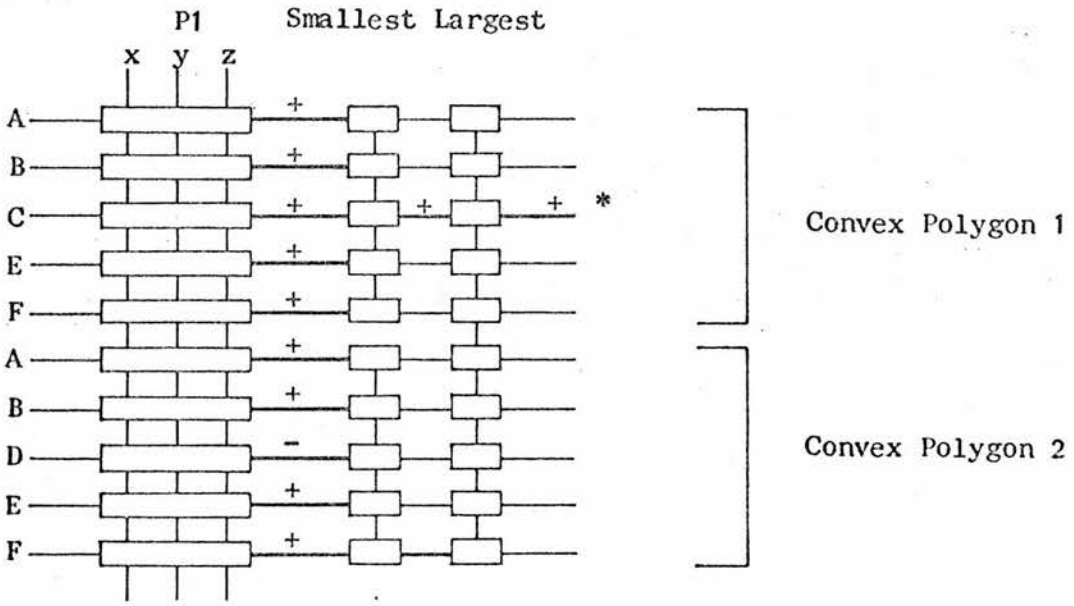
$$A.B.(C+D).E.F$$
$$= A.B.C.E.F + A.B.D.E.F$$

Test Polygon

Figure 18

If the expanded expression representing this polygon A.B.C.E.F. + A.B.D.E.F is placed in the machine by locating the corresponding plane

coefficients in this order in consecutive units, and the values of the point coordinate are passed down vertically from unit to unit, then the process for testing whether this point is inside the polygon or not is shown in Figure 19.



Testing Point in Polygon

Figure 19

It can be seen that the two levels in the expanded boolean expression are represented by the two levels of the selection circuit. In the first level only those planes in a product phrase of the expression are compared with each other. At this level all the values of k must be positive for the point to lie within the convex polygon represented by a product. At the second level only one value has to be positive for the test to be successful. This operation can be achieved by selecting the smallest value at the first level, a negative result indicating a local failure, while at the second level the largest value must be

selected. If a positive value exists it will be located in this way, so indicating that the point lies within the total polygon.

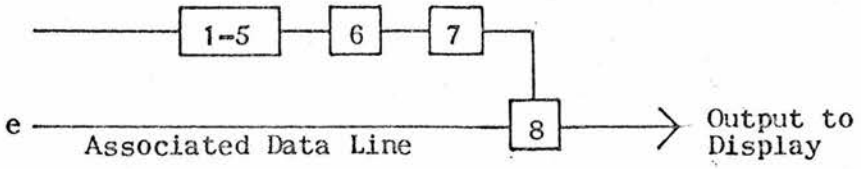
In the example shown in Figure 19 the value of z and w would be 1. It can be seen that for planes the same process can be applied, the only difference being that instead of the coefficients a, b, c, ø used for lines four values a, b, c, d are entered into each unit, and a three-dimensional coordinate x, y, z, 1 is introduced vertically.

The point in polygon and the point in polyhedron tests use elements 1, 2, 6, and 7 of the unit shown in Figure 7 . This requires the control switching to be able to by-pass elements 5, 4, and 3. Other switching required by this operation can be seen in the separation of the first stage comparison elements into groups corresponding to the product phrases of the boolean expression. It can be seen that this grouping will depend on the expression and must be set up when the polygon or polyhedron description is entered into the machine.

Raster Scan Display of Polygons

A simple extension of the point in polygon test can be used to generate a raster scan display of a polygon. By feeding in the coordinates of points from a raster into the machine vertically, and displaying the two value results: either positive or negative, as bright and dull a picture of the polygon will result. This limits the display capability to either one polygon or to disjoint polygons. However, if the output value of positive and negative for a particular polygon is used to control a switch which selects a value associated with each polygon, then it becomes possible to create a much more useful form of display.

This means extending the original unit to include an associated data line as shown in Figure 20 . In this case the data line is associated with each line or plane unit, which means that each line forming the boundary of a particular polygon will have to be associated with the same value. The significance of this point will become apparent when perspective displays are considered, where this apparent redundancy finds a useful application.



Selecting Associated Data Values

Figure 20

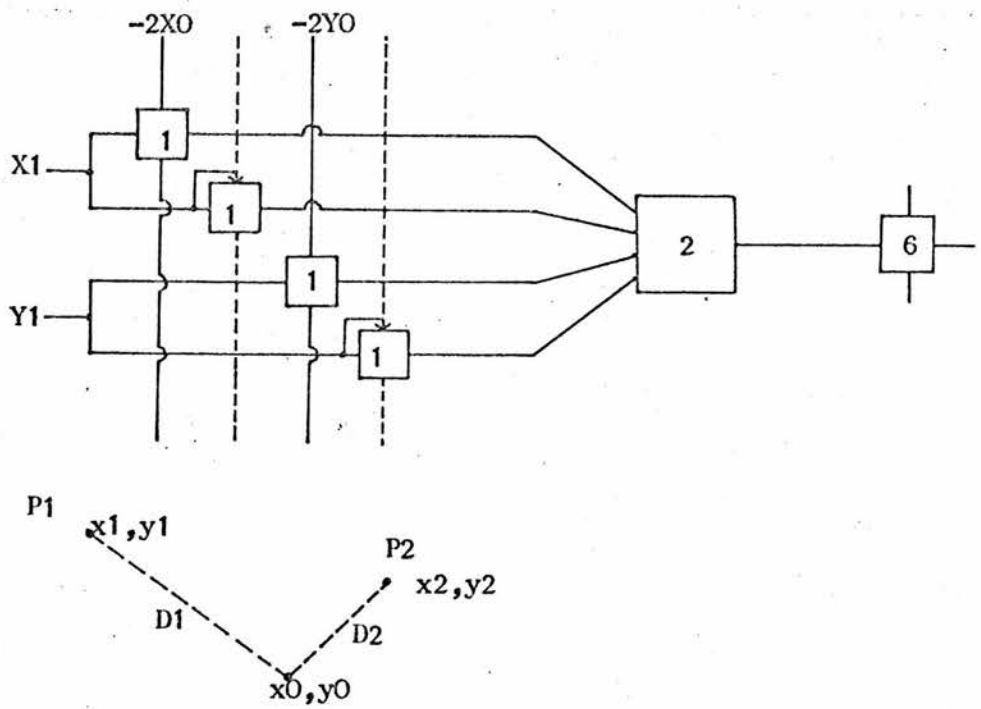
Line Shaded Displays of Polygons

In creating raster scan displays of polygons the different associated data values would be converted into different intensities or different colours. An alternative approach is to create line shading of different types to distinguish separate polygons and represent the values within them. This can be done by using this machine as a line clipping or scissoring device. If a line generator is used to create a stream of points along a line then these points can be tested using the point in polygon operation as before, by selecting the appropriate pattern of lines for each polygon a shaded polygon display can be created.

Nearest Neighbour Polygon Displays

As a by-product of the investigation into polygon displays it was found

that nearest-neighbour polygons could be generated from a collection of point coordinates with only a minor modification to the original layout of a unit. Where the value displayed at a point in a display depends on the value of the nearest data point, as in proximal mapping techniques, then the arrangement shown in Figure 21 will automatically generate the appropriate output. This technique again selects the associated data value to be displayed as with previous polygon display processes.



Circuit to Create Nearest Neighbour Polygons

Figure 21

If in Figure 21 point P_0 is the current display point and P_1 and P_2 are the two data points, then the two distances from the display point to these data points will be D1 and D2 respectively. These distances can be calculated from the respective coordinates of the points, so that

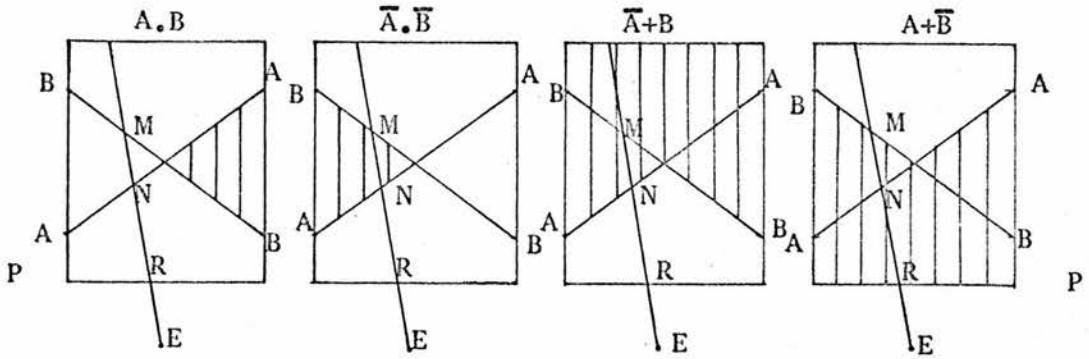
$$D1 = X1^2 + XO^2 + Y1^2 + YO^2 - 2.X1.XO - 2.Y1.YO$$

However, both XO^2 and YO^2 will occur in all calculations of D so they can be ignored. The result is the circuit shown in Figure 21 where the unit giving the minimum value of k switches through its associated data, and where

$$k = X1^2 + Y1^2 - 2.X1.XO - 2.Y1.YO$$

Shaded Perspective Display of Volumes

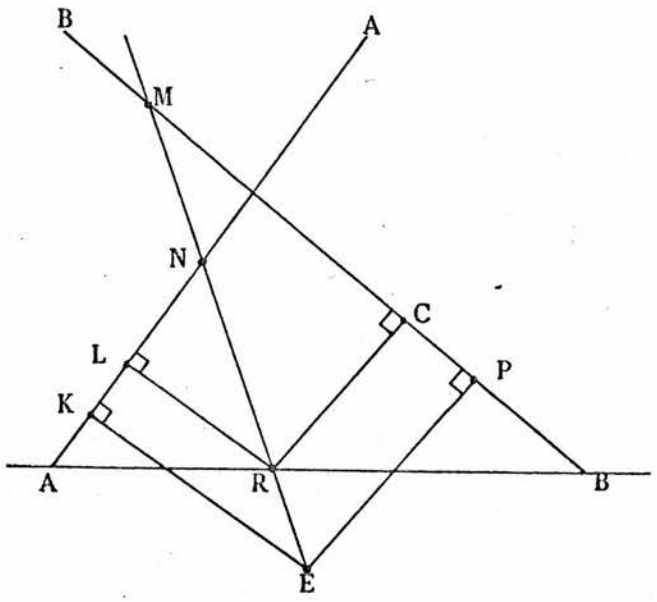
The remaining collection of applications for these units uses all the elements outlined in Figure 7 and are related to the display of three dimensional scenes. It was on the process described in Chapter 12, developed to produce perspective displays that the design of these units was based. The original process is the creation of a raster scan display of volumes as a perspective projection, with hidden areas removed. The basic relationships used in this operation are shown in the diagrams in Figure 22 where sections through simple objects are created by the plane taken through the viewing point and the raster line, PP.



Basic Relationships for Volume Display

Figure 22.

The initial problem is to determine the sizes of RM and RN at each point in the raster line in order to decide whether RM $>$ RN or RM $<$ RN. A direct approach to this calculation is shown in Figure 23 where two planes AA and BB are compared.



Comparing Two Planes

Figure 23

In Figure 23 the following geometrical relationships hold:

$$\frac{RM}{EM} = \frac{RO}{EP}$$ Similar Triangles ROM and EPM

$$RM = \frac{RO.(RM+RE)}{EP}$$

$$RM.EP - RM.RO = RO.RE$$

$$RM = \frac{RO.RE}{(EP-RO)}$$

Similarly based on the Similar Triangles RLN and EKN

$$RN = \frac{RE.RL}{(EK-RL)}$$

Expressing these relationships using point and plane coordinates

$$T_{1b} = a_b \cdot x_r + b_b \cdot y_r + c_b \cdot z_r + d_b$$

$$T_{2b} = a_b \cdot x_e + b_b \cdot y_e + c_b \cdot z_e + d_b$$

$$S = \sqrt{a_b^2 + b_b^2 + c_b^2}$$

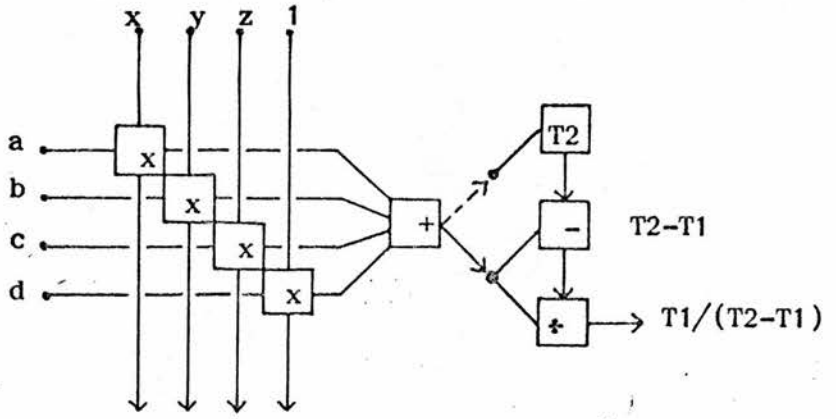
$$RO = T_{1b} / S$$

$$EP = T_{2b} / S$$

Since for a given raster point, RE is a constant for all planes it can be ignored in the comparison of RM and RN.

$$RM = T_{1b} / (T_{2b} - T_{1b})$$

The calculation of this value can be carried out for each plane by the circuit in Figure 24



Circuit for Volume Displays

Figure 24

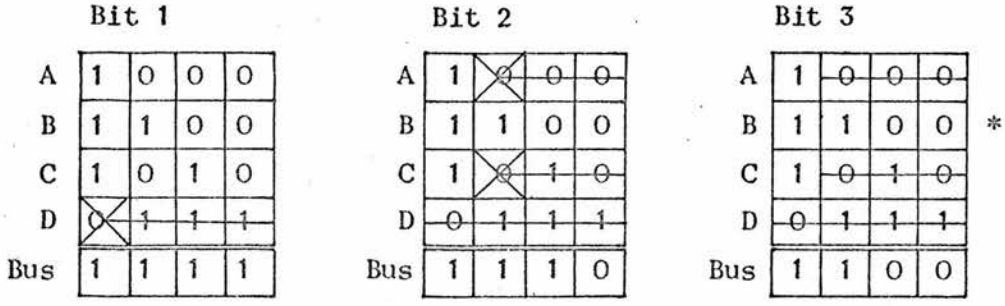
It is possible to initialise the value of T2 for each plane, using the circuit shown in Figure 24 by entering the coordinate of the viewing point, (vertically) instead of the raster point coordinates. The

raster point coordinates can then be entered to create the stream of values for each picture frame, which is passed to the comparison elements.

The comparison operation required to create a display turned out to be more complex than was first thought. The first stage of the comparison required both the maximum positive and the minimum negative value to be found for each product phrase in the boolean expression defining the scene. Two possibilities were investigated. The first employed a third vertical comparison where only back planes of convex volumes were compared, and the value selected in this process was then compared with the output from a similar comparison carried out for the front planes. The second approach attempted to extend the circuit described previously, to handle both positive and negative comparisons using one set of vertical comparison lines. In the final analysis these two approaches produced very similar results.

The drawback to the original comparison technique was that the process was sequential in that the largest value so far found was carried down for comparison with values found further down the list. An alternative way of carrying out this operation which increased the 'parallelness' of the process, and consequently reduced the overall time required to carry it out could be implemented in the following way. The bit values from the numbers produced by each unit were wire ored together on the bus lines. The value produced by each unit was then compared with the value on this bus. For each unit the values were processed as before starting with the most significant bit place and proceeding sequentially

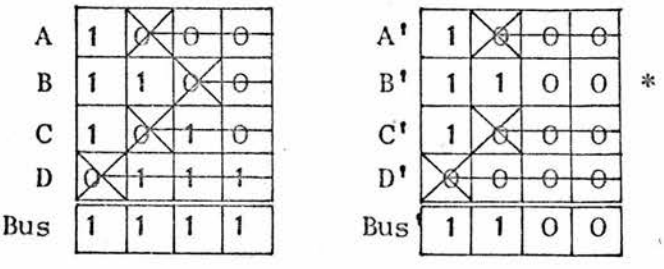
to the least significant place. As long as the bit value from the unit was the same as the value on the bus line no action was taken. As soon as they were found to be different then subsequent bit values output to less significant lines in the bus were switched off. This operation is shown diagrammatically in Figure 25.



Comparison Alternative I

Figure 25.

In Figure 25 it can be seen that the final outcome of this comparison is to output B as the largest value. It can be seen also, that the speed of the process depends on the time it takes for later bus lines to readjust to a stable value when an input value is switched off. A second alternative to this idea is shown in Figure 26.



Comparison Alternative II

Figure 26

In this alternative approach the values on the first bus are left as they are. The bit comparisons are carried out as before sequentially from the left. However, as long as the plane-unit bit values are the same as the corresponding bus bit values, a value of 1 is output to a second bus, as soon as they differ then subsequent outputs to the second bus are Ø. If the same process is then carried out using the second bus, the final output will be the largest value. Though this appears to introduce a secondary operation, it has the advantage that it can be pipelined.

Once it became clear that an appropriate comparison device could be constructed, it became necessary to examine in greater detail, the nature of the values which would be compared. In general terms this was the problem of choosing the most efficient form of scaling and coding for the data to be used in the process. The layout for the display processor described in this Chapter was developed directly from the software developed in the previous Chapter. As design work on the parallel implementation of the same operation advanced, it became apparent that several simplifications were possible. These modifications and the reasons for them are the subject of the next Chapter. In this Chapter the primary interest is in the list of potential uses to which this kind of processor could be applied.

The shaded perspective display of volumes can be achieved using the circuit shown in Figure 24 along with two levels of comparison. As the first level of comparison visible surfaces for convex volumes are located, from the product phrases of the boolean expression defining

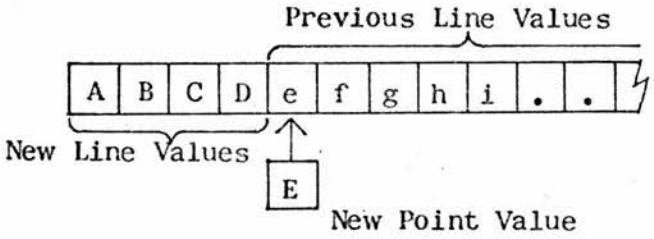
the whole scene. At the second level competition between surfaces of convex volumes is resolved to give the single visible surface at a particular point in the display. The selection of the visible plane does not indicate what the display data should be; this can vary from the illumination of the surface calculated from the plane's orientation, to the simple classificational coding which gives each plane a different grey-tone or colour. This process is similar to the way in which the parallel processor was used to switch through associated data for the polygon displays.

Line Drawing Displays of Volumes

There are two ways in which this processor can be used to generate line drawings of volumes. The first is akin to the process described previously for polygon shading. Lines are created on the picture plane but are only drawn in when the current point of the line corresponds with a particular plane being visible. This makes it possible to create a line shaded drawing using the same scanning procedure which has just been described. The advantage of being able to do this is that line drawings as a hardcopy record of photographic grey-scale displays can be created for simple and cheap drawing devices. Edges can be drawn using this technique, but they are more difficult than area shading lines because they may pass outside the 'inside' section of a particular facet. For this reason it is necessary to have edges associated with both the facets they separate, in other words the dual naming discussed in earlier chapters, to ensure that the edge lines are not incorrectly masked out. Even with this precaution edges which form the outer boundary of a volume viewed from a particular point may still present

difficulties. To draw edges using this technique requires an edge-based data structure to exist. This may not be possible if the input of volume descriptions is by means of plane coordinates and boolean expressions.

An alternative approach to the production of line drawings can be developed, based on the comparison of neighbouring points in a raster. This process is akin to the zone boundary definition algorithm developed in the OBLIX routine. If a line buffer is used to store output values, so that not only the values in the present raster line are known but also the previous raster line, then points lying in edge lines can be defined by the comparison process shown in Figure 27.



If D ≠ E or e ≠ E then Output = 1 A Line Point
Otherwise Output = 0 Not a Line Point

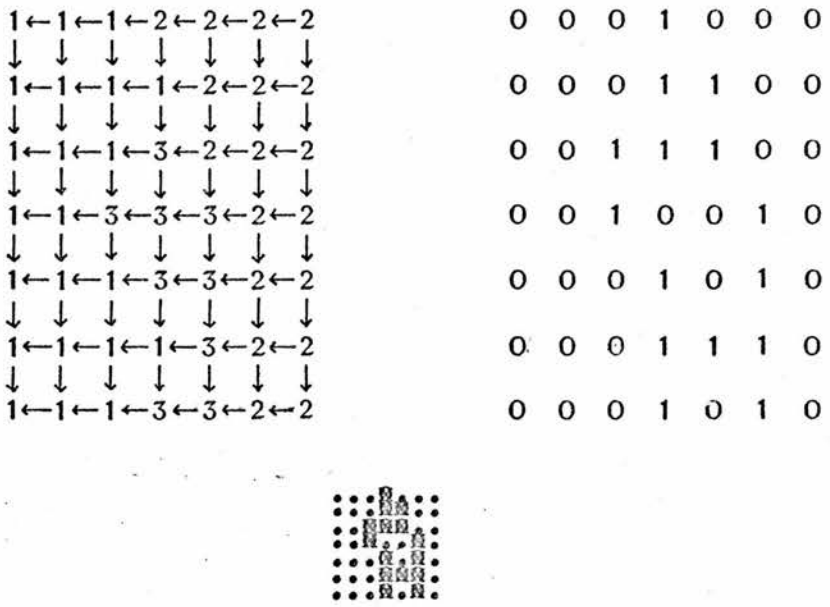
Creation of Line Displays Using Raster Values

Figure 27

The effect of this procedure is shown in the diagrams in Figure 28.

This basic technique can be modified in a variety of ways, but the basic process is the simple comparison method shown above. For example, where a very fine raster of points is used it may prove necessary to create a double line of points to represent a strong enough line in the display.

HARDWARE DEVELOPMENT: A SCHEMATIC MACHINE



The Creation of Line Displays from Grid Data

Figure 28.

Plan Section Displays of Volumes

In the previously described display techniques there are two situations where a point in the display raster may not be representing a visible surface. The first is where the point represents open space, in other words the equivalent of the open sky. The second is where the point represents the internal point of some volume. In both cases some value must be shown on the display to indicate which of these two cases has occurred. It is from the second of these two situations that a different form of display can be generated. If internal points are shown as black, and all other points are shown as white then a plan section is created. This by itself, however, is a fairly basic form of picture. If the complement of the boolean expression is used to create a display, then what were 'outside' spaces will become 'inside' spaces, and vice versa.

If instead of using associated data for each plane which describes its surface characteristics, associated data is used which describes the material from which the volume is made, then it is possible to convert a view of say, the inside of a room into a cross sectional plan which indicates the construction of the walls, furniture, etc. At the present stage of design it would appear that this is a very simple change to create, and it could be implemented as two modes for the display processor to be operating within.

Conclusions

The initial analysis for each of the processes described in the list on page 440 shows that they could all be implemented in some form using the original modular unit in a parallel processor. In the next two chapters the original design ideas are investigated more carefully and a more detailed specification begins to emerge.

CHAPTER 14

A PARALLEL DISPLAY PROCESSOR

A PARALLEL DISPLAY PROCESSOR

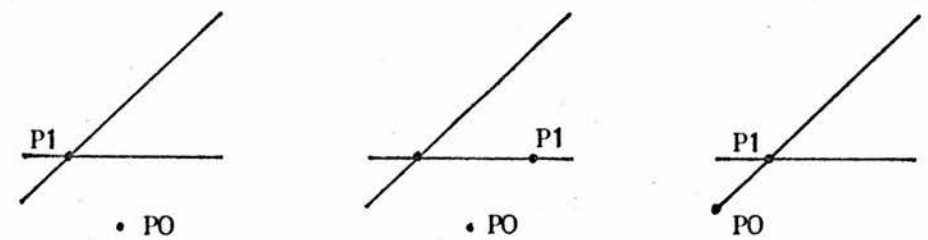
In this chapter the emphasis is placed on the implementation of the display operations described in Chapter 13. In practice, this consisted of a study of the volume displays rather than the polygon displays because once the more complicated problems have been solved for the former it appears to be relatively easy to then implement the latter.

The algorithm used in Chapter 12 to create a display was based on the value of the expression:

$$K = T1 / (T2-T1)$$

for different points on the display plane. For simple cases, this expression gave values of K which were positive for front facing planes, in other words where the viewing point lay 'outside' the plane; and a negative value for K for back facing planes where the viewing point lay 'inside' the plane. A series of special cases made the software implementation of this process unsuitable for direct conversion into hardware, and the first stage was to examine these cases so that some way of avoiding them could be found.

Figure 1 shows the three cases where the simple circuit discussed in the previous chapter will need some form of special provision to create the acceptable output.



Difficulties in the Calculation of K

Figure 1

A PARALLEL DISPLAY PROCESSOR

The simplest case occurs where T1 = 0, in which case the output: K = 0 is not difficult to create. Where T1 = T2 the divisor becomes zero, and the value of K must be set to the maximum number which the linking lines can be made to hold. A similar result is required in the case where T1 = T2 = 0. The real difficulty in these cases is to determine the sign of K when the maximum value is being generated. This point is returned to later.

A second line of attack was to simplify the calculation of the value of K which was passed to the comparison processes. This appeared to be necessary because though operations seemed few in number when considered in the context of sequential processing, the demand for 4n multipliers and n division units, to process n planes in a parallel process, demanded a very large machine to handle even relatively simple scenes.

The distance from the raster point P1 to a plane A along a viewing ray from a viewing point P_O: Da is given by the relationship

$$Da = c.T1 \;/\; (TO-T1)$$

The operation of the comparison stage of the process depends on the ordering of the values of D:

$$Da < Db < Dc < Dd \quad \ldots\ldots$$

Expanding this relationship gives:

$$\frac{c.T1a}{(TOa-T1a)} < \frac{c.T1b}{(TOb-T1b)} < \frac{c.T1c}{(TOc-T1c)} < \cdots\cdots$$

Inverting:

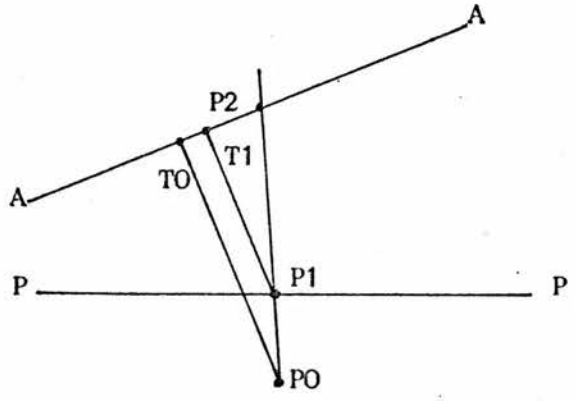
$$\frac{TOa-T1a}{T1a} > \frac{TOb-T1b}{T1b} > \frac{TOc-T1c}{T1c} > \cdots\cdots$$

$$\frac{TOa}{T1a} - 1 > \frac{TOb}{T1b} - 1 > \frac{TOc}{T1c} - 1 > \cdots\cdots$$

$$\frac{TOa}{TOa} < \frac{T1b}{TOb} < \frac{T1c}{TOc} < \cdots\cdots$$

This means that the original calculation to produce T1/(TO-T1) can be reduced to T1/TO, if the various conditions which were used to decide whether the viewing point was inside or outside a plane could still be used in the same way.

The range of values which the entity T1/TO can take is easier to handle than the range of values created by the previous expression.



Section Through a Raster Line

Figure 2

In a section through the picture plane PP along a raster line shown in Figure 2 the following relationships hold. If P2 is in front of the picture plane, in other words on the opposite side to the viewing point, then T1 and TO give the inequality: $0<T1/TO<1$. If the sign

of K is being used to indicate whether a plane is a front or back plane, then it is necessary to define K by the relationship:

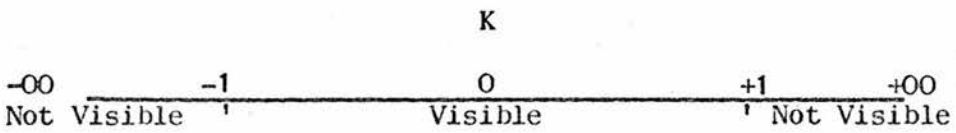
$$K = \frac{T1}{|T0|}$$

In this case the range of values for K, if P2 lies in front of the picture plane, lies between -1 and +1. If P2 lies between P0 and P1 on the viewing ray then: $0 \geqslant T1/T0 \geqslant -\infty$ and $+\infty > K > -\infty$. If P2 lies behind the eye, P0, then the value of K will lie in the range -1 to $-\infty$ or +1 to $+\infty$. Where the intersection of the plane and the viewing ray moves to infinity then $T1 \longrightarrow T0$ and the value of $K \longrightarrow \bar{+}1$. The limiting case is where the viewing ray becomes parallel to the plane A. If the intersection point of the plane and the viewing ray lies behind the eye then the value of K will either be greater than 1 or less than -1.

At the comparison stage, if the value of K is -1 or less, then the raster point is inside the volume and the visible plane still has to be established: if the value of K is +1 then the raster point lies outside the volume, and no plane in the convex volume containing this plane will be visible. Consequently, except for the condition where P2 lies between P1 and P0 a very simple sorting strategy results. In the case where P2 lies between P1 and P0 it is still possible for the value of K to lie between +1 and -1 but for the plane to be invisible. There is, however, one relationship which makes it possible to remove this problem. When P2 lies between P1 and P0 then the signs of T1 and T0 will be opposite; consequently, when this occurs, the value of K can automatically be given the value of +1 or -1. The sign of K is made the same as T1. The algorithm to create K is thus:

```
If (Sign(TO) = Sign (T1)) then K = T1/ Absolute(TO)
        Else if (T1<0) then K = -1
        Else if (T1<0) then K = 1
End.
```

The application of this procedure to all planes gives the range of values for K shown in Figure 3



Value Range for K

Figure 3

Having successfully removed a subtraction from the previous process, it was then found that each calculation of K for a plane at successive raster points did not require the division circuit if the input data was correctly prepared for each frame of the display. This meant that some of the calculation for each plane could be carried out once per frame and there was consequently more time to carry this out and it could be done sequentially.

The input data representing a plane have so far been in the form of a vector (a, b, c, d) being the coefficients of the plane's equation. These coefficients have been scaled so that the product of these coefficients with the coordinate of a point (x, y, z) to give $a.x + b.y + c.z + d$, provides the perpendicular distance from the point to the plane. The expression T1/TO can be rewritten:

as $$\frac{a.x_1+b.y_1+c.z_1+d}{a.x_0+b.y_0+c.z_0+d}$$

or $$\frac{a.v_1}{TO} + \frac{b.y_1}{TO} + \frac{c.z_1}{TO} + \frac{d}{TO}$$

However TO is only calculated once per frame, so the following values need only be calculated once per frame:

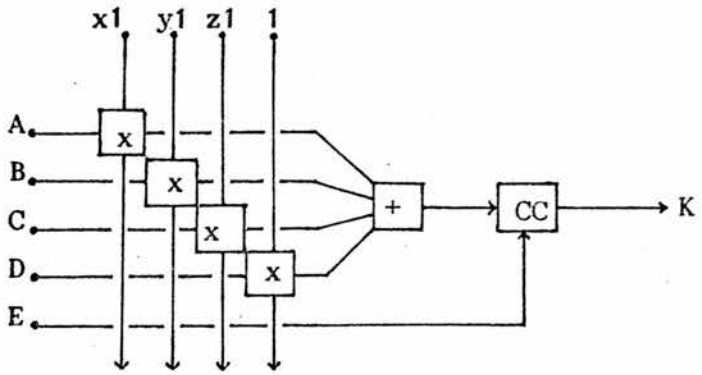
$A = a/TO$

$B = b/TO$

$C = c/TO$

$D = d/TO$

Since $A.x_1+B.y_1+C.z_1+D = T1/TO$, it means that the division operation can be performed for each plane once, for each new picture displayed, instead of once for each point on the raster. This simplifies the display circuit to the layout shown in Figure 4.



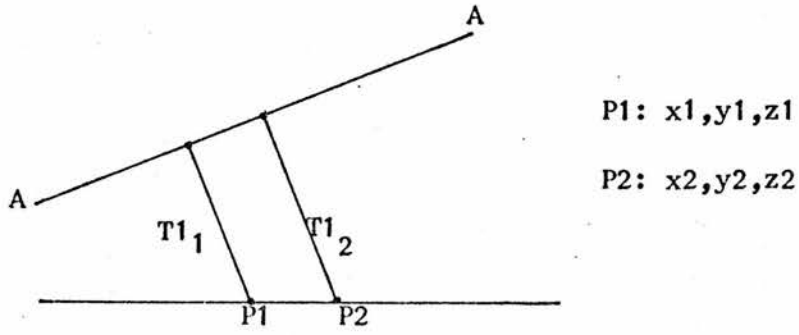
Display Circuit without Division

Figure 4

The operation of the element 'CC' is to correct the output if the sign of T1 is opposite to the sign of TO. This requires the original sign of TO to be entered as data on the input line named E in Figure 4.

Once it had been realised that certain parts of the calculation carried out by the original processing module need not be carried out

further simplifications to the parallel processor became possible. The regular spacing of the points in the display raster made it possible to remove the multiplication operation from the display process. Consider the two raster points P1 and P2 in Figure 5.



The Relationships between Raster Points

Figure 5.

In moving from P1 to P2 the values of x_1, y_1, z_1 are changed by a constant increment dx, dy, dz to give x_2, y_2, z_2.

$$T1_1 = a.x_1 + b.y_1 + c.z_1 + d$$

$$T1_2 = a.x_2 + b.y_2 + c.z_2 + d$$

However, the second relationship can also be expressed as

$$T1_2 = a.(x_1+dx) + b.(y_1+dy) + c.(z_1+dz) + d$$

By Subtraction:

$$dT = T1_2 - T1_1 = a.dx + b.dy + c.dz$$

This means that within a raster row all values of T1 for each point can be obtained by adding dT to the value of T1 for the previous point. The value of T1 for the first point in a row can in a similar way be calculated by adding a DT to the first value in the previous row. Where

$$DT = a.Dx + b.Dy + c.Dz$$

and Dx, Dy, Dz are the changes in the coordinates when moving from one raster point to the one vertically below it.

A plane can thus be represented by four values: A, B, C, D where

A = dT

B = DT

C = T1

D = TO

As they stand these values will still require a division for each raster point. If this approach is combined with the previous approach then these four input variables can be given the following values:

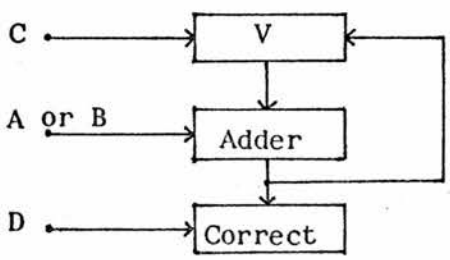
A = dT/TO

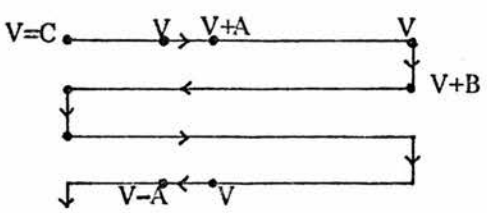
B = DT/TO

C = T1/TO

D = Sign(TO)

A circuit which will create the required values for K for each plane can be constructed using a single adder as shown in Figure 6.





The Order in which Raster Points are generated.

Display Circuit Based on an Adder

Figure 6.

This process was simulated as a sequential operation using the FORTRAN routine given below. The volumes used in this exercise are shown in Figure 7 . To save programming effort the volume descriptions are built into the routine as DATA statements, and the output is created as a raster of identifying numbers. Since there are less than ten surfaces which will be visible from any viewing point, this means the display can use single digits for each raster point and the table of numbers can be read as a picture.

Data Generation

```
      DIMENSION A(12),B(12),C(12),D(12),E(12),F(12),G(12)
      DIMENSION H(12),IR(12),IP(12),NM(60),P(12)

      DATA A/-1,1,1,-1,0,0,-1,1,1,-1,0,0/
      DATA B/1,1,-1,-1,0,0,1,1,-1,-1,0,0/
      DATA C/0,0,0,0,-1,1,0,0,0,0,-1,1/
      DATA D/-4.9497,-12.0208,2.1213,6.3640,0.0,-6.0,-6.346
     1,-10.6065,0.7071,7.7782,1.0,-5.0/
      DATA IR/0,0,0,0,0,1,0,0,0,0,0,1/
      DATA IP/9,1,2,3,0,0,4,5,6,7,0,0/
      DATA DX/0.1/,DY/0/,DZ/0/,EX/0/,EY/0/,EZ/-0.16666/
      DATA X1/1/,Y1/5/,Z1/6/,XO/4/,YO/2/,ZO/3/

      N=12
      IX=0
      M=60
      I=36
      DO 5 J=1,N
      A(J)+A(J)*0.7071
    5 B(J)=B(J)*0.7071
      DO 10 J=1,N
      E(J)=A(J)*DX+B(J)*DY+C(J)*DZ
      F(J)=A(J)*EX+B(J)*EY+C(J)*EZ
      G(J)=A(J)*X1+B(J)*Y1+C(J)*Z1+D(J)
      H(J)=A(J)*XO+B(J)*YO+C(J)*ZO+D(J)
      IF(H(J).EQ.0)GO TO 10
      S=H(J)
      IF(S.LT.0)S=-S
      E(J)=E(J)/S
      F(J)=F(J)/S
      G(J)=G(J)/S
   10 CONTINUE
```

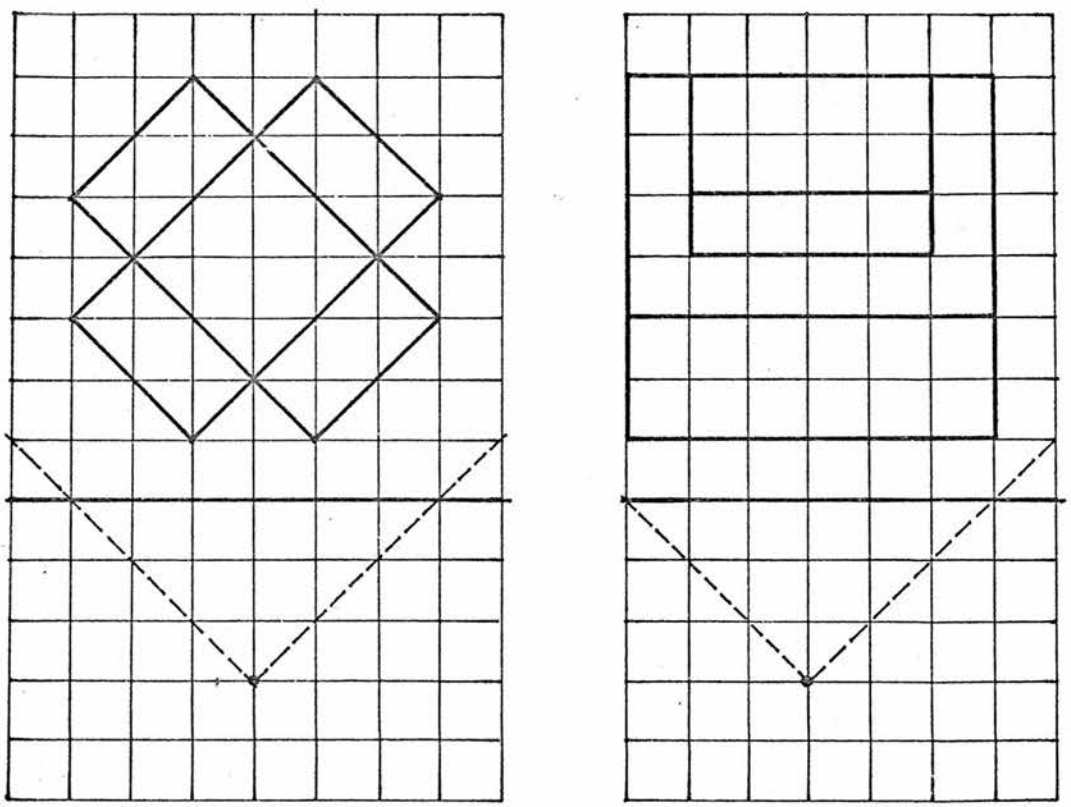
Display Generation

```
      DO 150 L=1,I
      DO 140 K=1,M
      Q=9999999999
      T=0
      U=-9999999999
      DO 130 J=1,N
      IF(K.NE.1)GO TO 15
      G(J)=G(J)+F(J)
      P(J)=G(J)
   15 CONTINUE
      P(J)=P(J)+E(J)
      S=P(J)
      IF(H(J).EQ.0)GO TO 40
      IF(H(J).LT.0)GO TO 20
      IF(S.LT.0)GO TO 40
      GO TO 30
   20 IF(S.GT.0)GO TO 40
   30 IF(S.GT.1)S=1
      IF(S.LT.-1)S=-1
      GO TO 50
   40 IF(S.GT.0)S=1
      IF(S.LT.0)S=-1
   50 IF(S.LT.0)GO TO 70
      IF(T-S)60,60,70
   60 T=S
      IV=IP(J)
      GO TO 90
   70 IF(U-S)80,80,90
   80 U=S
      IW=IP(J)
   90 IF(IR(J).EQ.0)GO TO 130
      IF(T+U)100,120,120
  100 IF(T-Q)110,120,120
  110 Q=T
      IX=IV
  120 T=0
      U=-9999999999
  130 CONTINUE
      NM(K)=IX
      IF(Q.EQ.0)NM(K)=10
      IF(Q.GE.0.999999)NM(K)=0
      IX=0
  140 CONTINUE
      WRITE(6,1000)(NM(KK),KK=1,M)
 1000 FORMAT(6011)
  150 CONTINUE
      STOP
      END
```

A PARALLEL DISPLAY PROCESSOR

The first section of this program is merely an ad hoc construction to set up the data for the second section which actually creates the display.

This program led to the clarification of two issues. The first was related to the removal of special cases, the second to the form of the comparison and sorting operations. In this routine it was still necessary to handle the case where TO = 0, or where TO was so small that it was effectively zero, as a special case. When the situation which demanded TO to be zero was considered in association with the value range of K which could affect the display, and the resolution of the display, it became clear that a uniform approach to all situations could be set up which did not require special provision to be made for relatively rare conditions.



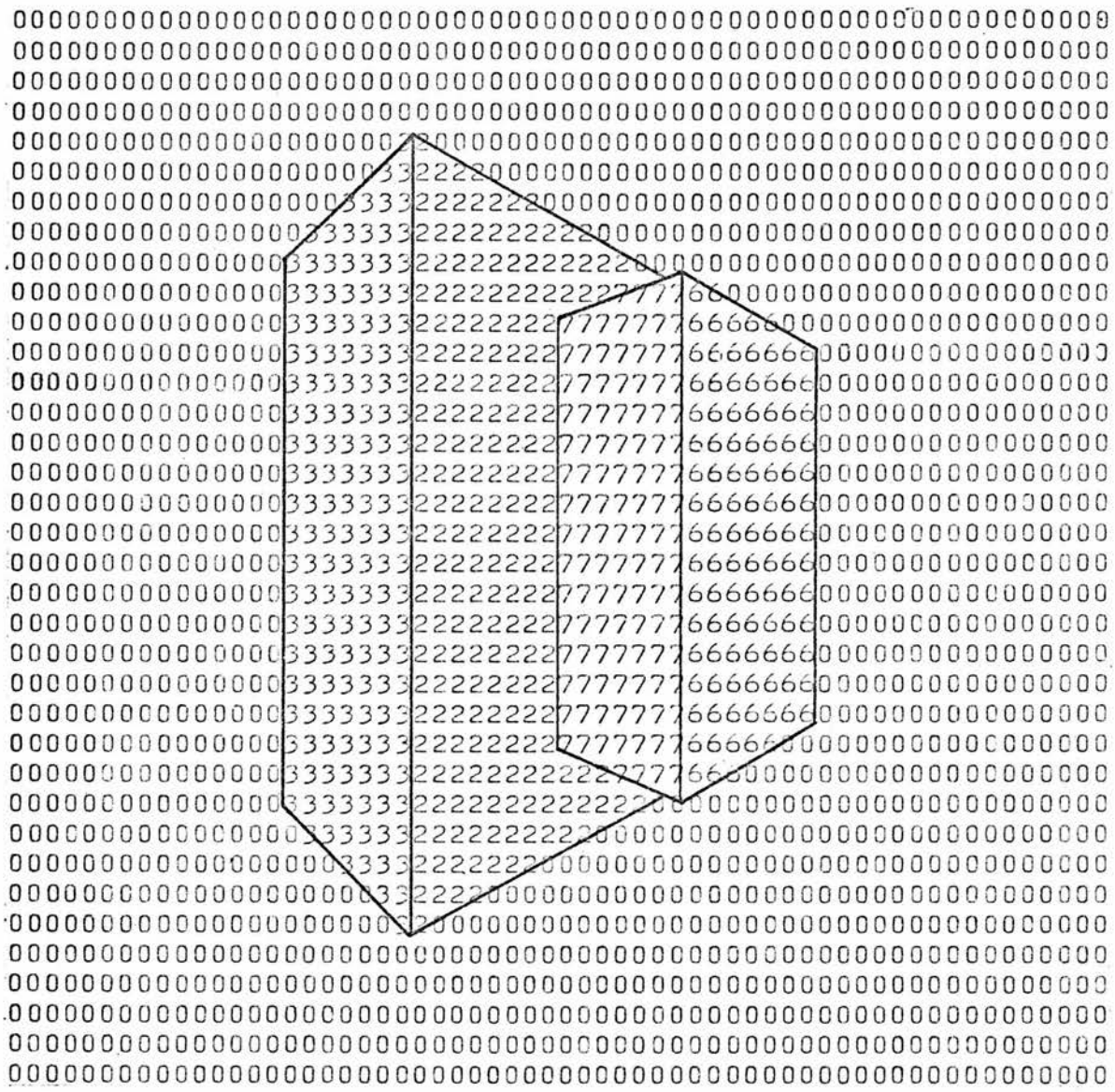
Test Data for Display Simulation

Figure 7

The second issue was concerned with the sorting operation described in the previous chapter. A fairly complex arrangement seemed to be necessary to provide the next visible surface, which made use of the order of all the K values being generated by the collection of plane units acting in parallel. In the sequential operation it was necessary to use the simple comparison of two numbers, or the equivalent operation of subtraction linked to a sign test. It became clear that the same simple comparison could be carried out in the parallel processing units because the frequency of multiple comparisons needed would be small. This issue will be returned to later because there are obvious complications which depend on the 'coherence' of the scenes being displayed. These problems are relatively less important because they are concerned with picture accuracy rather than creating an initial stable image.

In the program used to create the display shown in Figure 8 the values of all the variables associated with the calculation of K and the comparison operations, were held as floating point numbers. This was done in order to avoid as far as possible the problems which are linked with the use of very small or very large numbers when there is a limited number 'size'. The aim in creating the limited range of values for K between -1 and +1 was to allow a fixed point decimal representation of numbers to be used, since this would simplify the arithmetic operations used in the plane processor units. The effect of using fixed point values for the display operation was to make the process equivalent to that which would be employed using a three dimensional grid. It is possible to rearrange the position of the elements being displayed so that they occupy the unit cube shown in Figure 9.

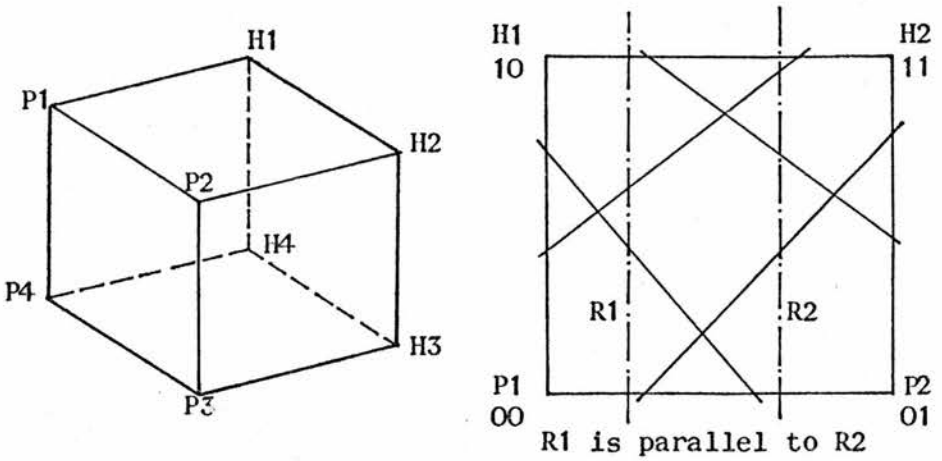
A PARALLEL DISPLAY PROCESSOR



Test Output from Sequential Simulation

Figure 8

The transformation which occurs when changing from K=T1/(TO-T1) to K=T1/TO changes values from the range O to OO to the range O to 1. This transformation is the equivalent of the perspective transformation which is applied to point coordinates, being applied to plane co-



The Display Box

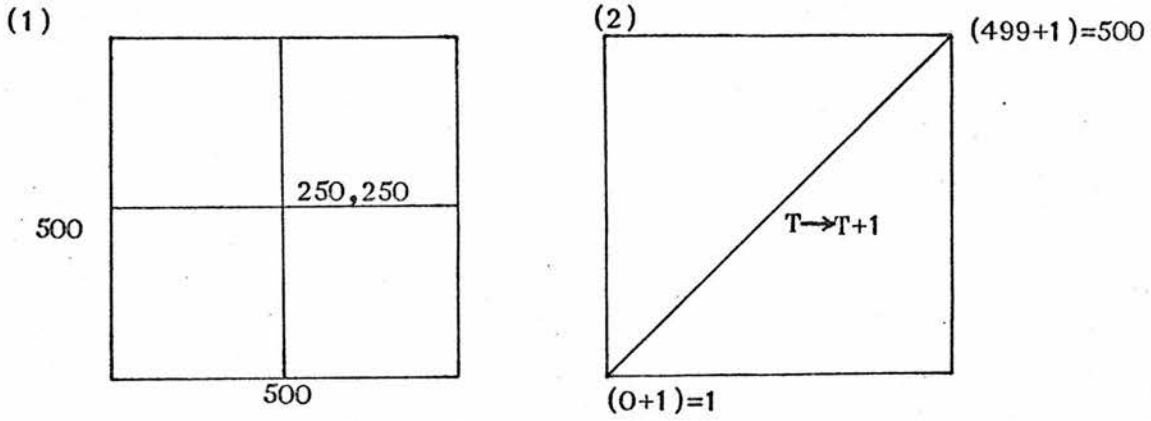
Figure 9

ordinates. Planes in the original space are transformed to planes in the picture box shown in Figure 9 but the viewing rays which in the original space meet at the eye, are transformed into parallel lines perpendicular to the front face of the picture box - the picture plane.

If the values of K and the X and Y positions of the raster point on the picture plane are taken as the transformed coordinates of a point on a plane, then it can be seen that the fixed point representation of these values will lie on a regular grid within the picture box in Figure 9 Starting with a predefined picture resolution it becomes possible to specify the accuracy with which the fixed point representation of variables must be given, in order to represent any distinguishable change in a plane.

When analysing this problem it is easier to start with the grid in the picture box and then specify the variables T, dT, and DT necessary to define a plane by the process of cumulative addition, than to start

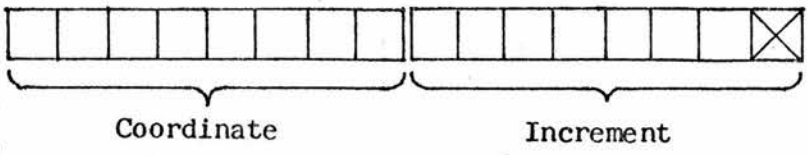
with the plane equation. Consider a unit cube with its sides divided into 500 strips in three directions to give a three-dimensional grid. Any point on the grid can be defined by a coordinate made up from three eight bit numbers.



Defining a Surface by Cumulative Addition

Figure 10

The problem is to define the range of sizes which the increments dT and DT must take, in order to create different surfaces within the grid. Consider a section shown in Figure 10 where the x increment is dT. If T is initially $\emptyset$ and dT is one unit, in other words 1/500th of the side of the unit cube, then if dT is added to T for each step taken along the raster line PP, the line generated will lie at 45^{o} to the Picture Plane, as shown in diagram 2 in Figure 10. To obtain a line which subtends a smaller angle with the picture plane, requires a smaller increment for dT. The limiting case must be where there is only one step in the z direction during the whole raster scan. This requires an increment which is 1/500 of the vertical, z unit in other words 1/250000th of the side of the unit cube. This requires the size of the fixed point coordinate to be extended by 7 bits, as shown in Figure 11.

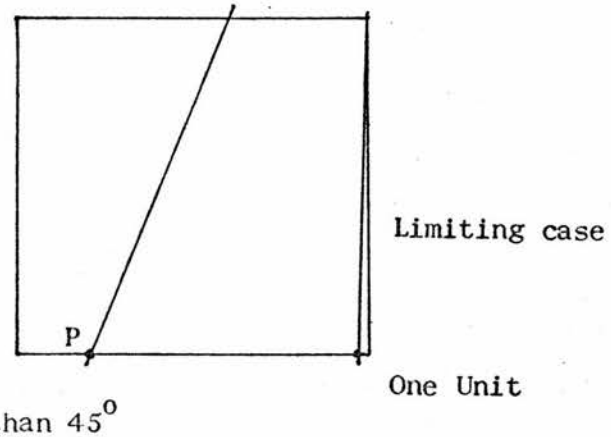


Word Size for Planes between 0^{o} and 45^{o} with the Picture Plane

Figure 11

If a 1 is added to the fifteenth place in Figure 11 then within 500 increments the carries will affect the 8th place, so stepping the surface back by one unit. This makes it possible to handle the range of plane positions which it is possible to represent in the range 0^{o} to 45^{o} with the picture plane.

A similar problem occurs for planes in the range 45^{o} to 90^{o} with the picture plane. Consider the situation shown in Figure 12.

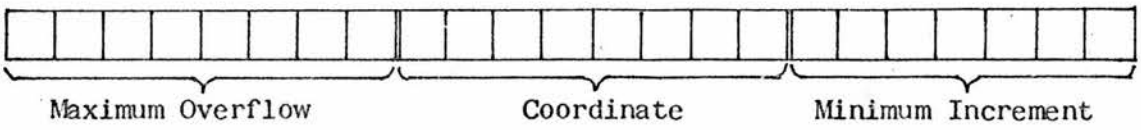


Planes Steeper than 45^{o}

Figure 12.

The problem in this case is to locate the position of the point P, where the plane appears within the picture box. By a similar argument to the one used above it is possible to show that another eight places are required in front of the coordinate shown in Figure 11 if the

same incrementing process is used to define the plane position. In this case the plane does not exist as a visible surface where the z coordinate is greater than the original eight bit number. The maximum word size required for a picture box of resolution 500^3 is shown in Figure 13.



Word Size for Planes in a Picture Box 500 units Cubed.

Figure 13.

Though the range of values which has to be stored for this level in picture resolution requires 24 bits, the maximum range for the incrementing values can be contained within 16 bits. If the increment is the maximum size which can be represented by 16 bits then if the original value of T is Ø then one increment will take the plane from the front of picture box to the rear of the box. This gives the nearest representation of a plane perpendicular to the picture plane which is possible using a raster scan display, so any increase in the size of the increment will not be registered in the display. In a similar way the minimum value which needs to be passed to the comparison elements is the number represented by the central group of eight bits. The definition of the picture will be improved in certain rare conditions if a larger number of bits is used in this comparison stage, but this concerns the faithfulness of the internal picture to the input data rather than its consistency.

This completes the analysis of the first stage of the display unit, using digital techniques. The next stage was the development of the comparison process. The ideas behind this part of the work were developed from attempts to modify the display operation so that it could be carried out using analogue rather than digital circuitry. These experiments are presented in the next chapter where an alternative implementation of the operations examined in this chapter are explored.

CHAPTER 15

PARALLEL PROCESSOR USING ANALOGUE SIGNALS

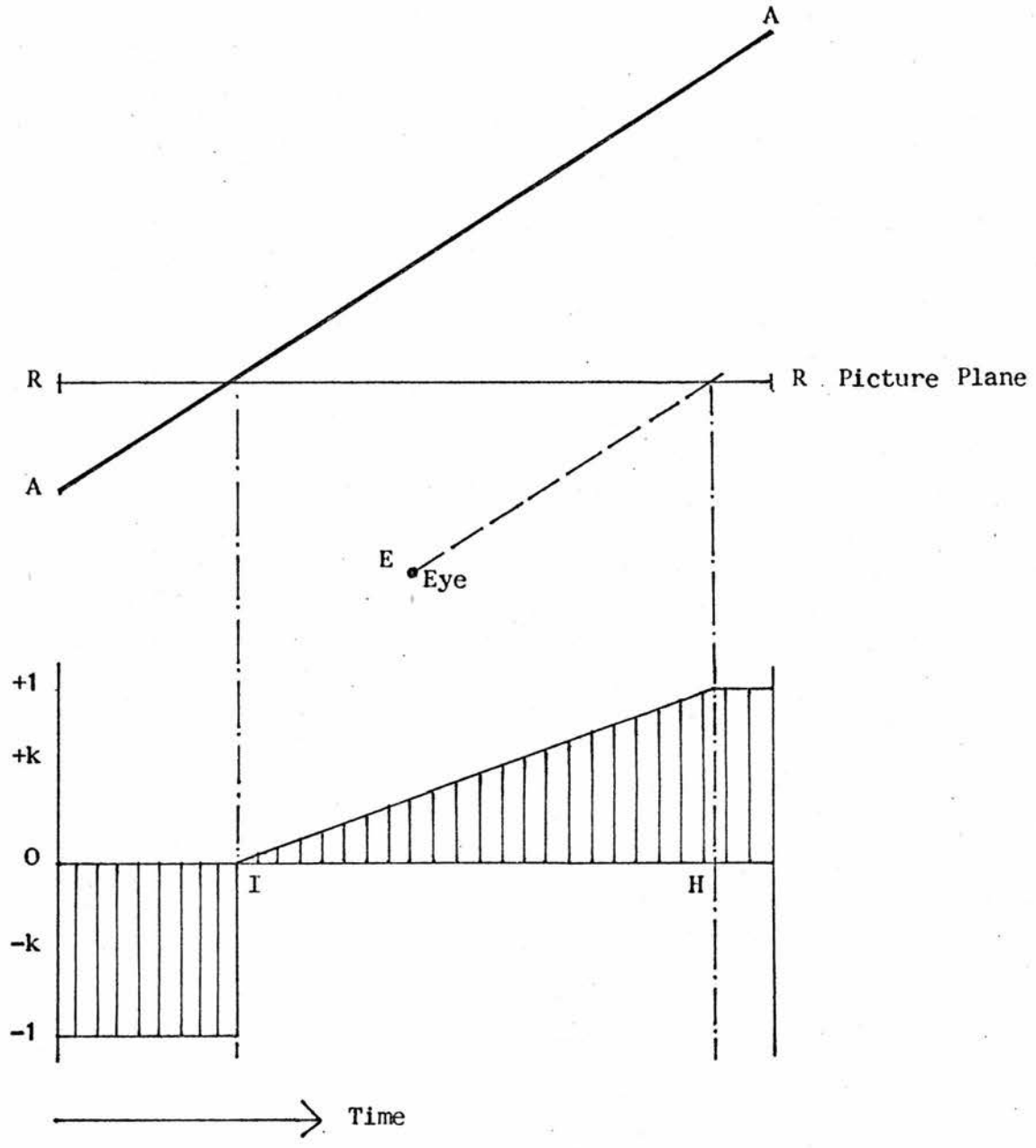
PARALLEL PROCESSOR USING ANALOGUE SIGNALS

The creation of a raster scan display using digital operations required the display to be considered as a set of discrete points. If the output device were a T.V. monitor then the display is in fact made up from a set of lines. To create a raster line which is a continuously varying signal means the evaluation of the functions so far developed for discrete points as a continuous operation. To see whether an analogue-digital process offered any advantages in carrying out this process the operations discussed in the previous chapter were restructured for implementation on an hybrid computer.

The problem of restructuring initially seemed to revolve round the constraint that all analogue values had to be represented in the voltage range -1 to +1 volts. If the input data again taken as the plane equation coefficients (a, b, c, d), then though a, b, and c, being direction cosines of the planes orientation, will be less than 1 the value of d can be in the range OO to -OO. If the goal is to create K where K=T1/TO, subject to the correction on the sign of TO, then the value of K of interest falls in the range -1 to +1. Consequently, if a way of creating K from the plane coefficients can be evolved, then it would appear that the first stage of the display operation can be implemented using analogue circuitry.

In figure 1 the general relationships which have to be taken into account for a plane are shown. In diagram 1 the relationship of the plane in space, to the picture plane, and to the viewing point are shown. In diagram 2, vertically below, the signal K which matches the different positions on the raster line in the picture plane is shown for this plane.

PARALLEL PROCESSOR USING ANALOGUE SIGNALS



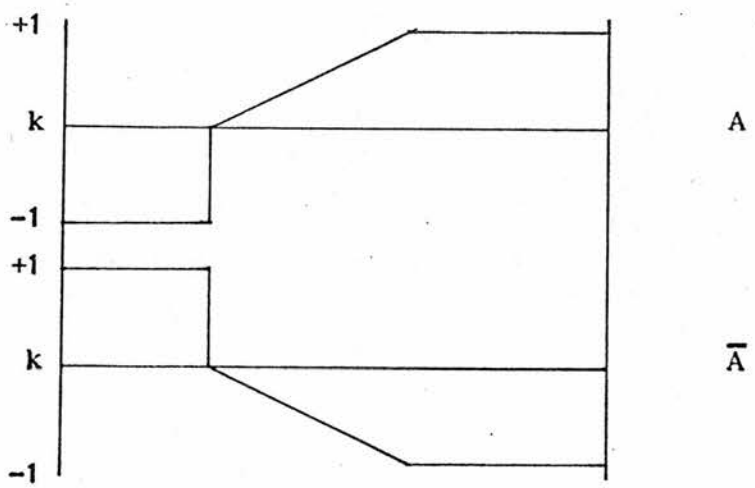
The General Relationship Between a Plane and Signal K

Figure 1

Given a plane A, a viewing point E, picture plane P and raster line RR, Figure 1 shows the relation of the signal K with the time taken in scanning the scene. The value at I will be zero because T1 is O. The value at H will be 1 because T1 = TO, since EH is parallel to AA. The value to the left of I in the diagram will be negative. Since the sign

of T1 and TO are opposite in this section then K is given the corrected value of -1. To the right of H the value of K would be greater than 1 but since for the comparison stage all values greater than or equal to 1 are treated the same the signal can be truncated as 1.

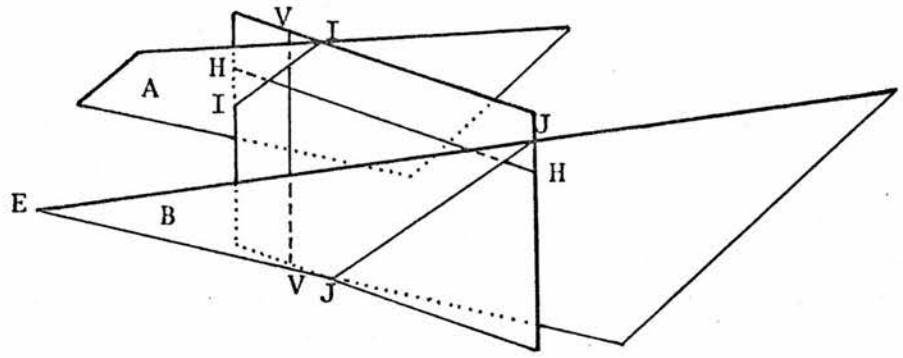
If a plane and its complement are considered, then the two raster line signals will be mirror images on the time axis, as shown in Figure 2



Signals for a Plane and its Complement

Figure 2.

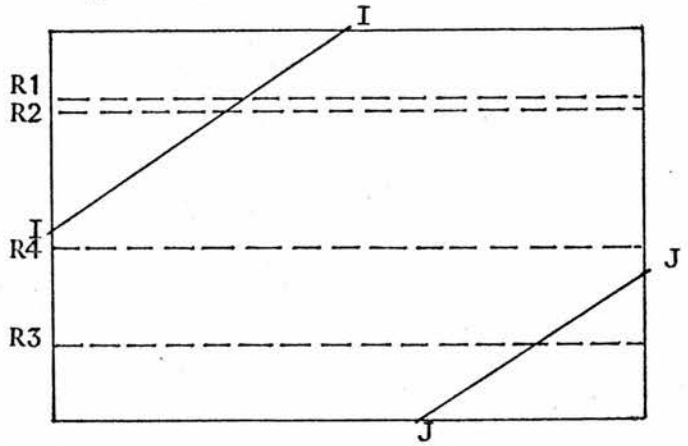
The 'shape' of the signal would be similar if a vertical raster line were used. The relationship between a vertical scan and a horizontal scan are shown in Figure 3 . The displayed plane is A, and B is the parallel plane which passes through the viewing point E. HH represents the horizontal raster line, VV the vertical raster line, II is the intersection of the plane A with the picture plane, and JJ is the corresponding intersection line of the plane B with the picture plane.



Three Dimensional Relationship of the Signal K and Plane A

Figure 3

In Figure 3 line II is the locus of all the zero values for raster line signals whatever their direction across the picture plane. Similarly, the line JJ - the horizon line in perspective construction - is the locus of the +1 values for the raster line signals, where the planes are positive relative to the viewing point. Looking at the picture plane from E, the lines II and JJ are parallel to each other, as shown in Figure 4.

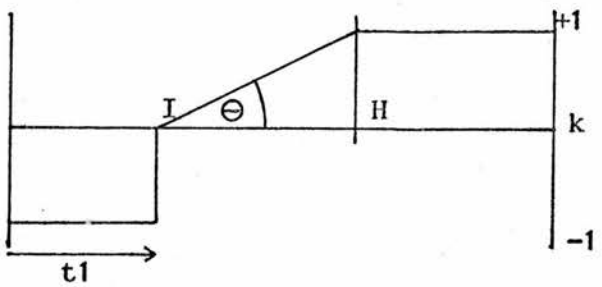


The Display Screen

Figure 4.

PARALLEL PROCESSOR USING ANALOGUE SIGNALS

Assuming the plane is positive relative to the viewing point, then the basic signal for R1 and R2 in Figure 4 will be of the form shown in Figure 5.



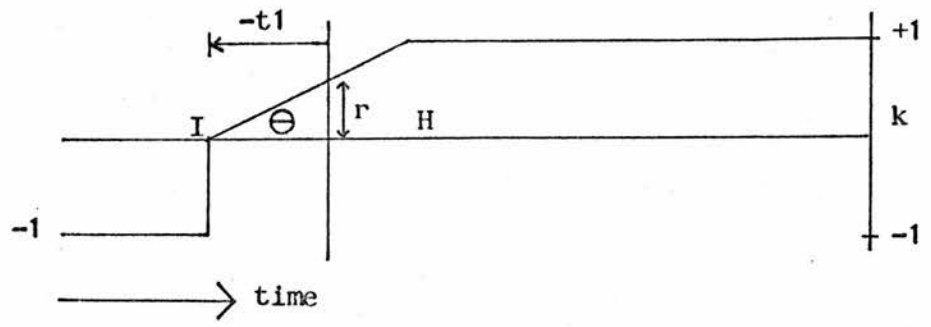
Basic Raster Signal

Figure 5.

Such a signal is not difficult to generate. For a time t1 a steady signal of -1 volts is output. The output is then switched to an integrator initialised to start at 0 volts and with parameters set up to give an increase in the output voltage matching the slope Θ. When the signal reaches +1 volts the output from the integrator is limited to this value.

Since the raster lines are assumed to be parallel, the distance IH will be constant. Moving from raster line R1 to R2 consequently requires the size of t1 to be modified by the appropriate amount. The inadequacy of this simple scheme becomes apparent when raster lines R3 and R4 are considered. To create the correct signal for these two lines would require t1 to become negative. The picture frame has been moved in relation to the original signal which still in theory retains the same shape as shown in Figure 5.

It is still possible to use an integrator to generate this signal if it



Modified Raster Signal

Figure 6

is initialised to give the value of r as its starting output. If tl is given the value 0 then the integrator will switch on immediately at the beginning of the raster line. The problem consequently becomes that of setting up the values of tl, r, and Θ from the plane coefficients (a, b, c, d). To simplify this problem, the eye was assumed to be at the origin of the coordinate axes used to define the coefficients of the plane. The direction of viewing was taken along the y axis, and the picture plane taken parallel to the YZ axes plane, at a distance e in front of the eye.

These relationships are shown in Figure 7 . Diagram 1 shows the three dimensional relationships, and diagram 2 shows the display frame.

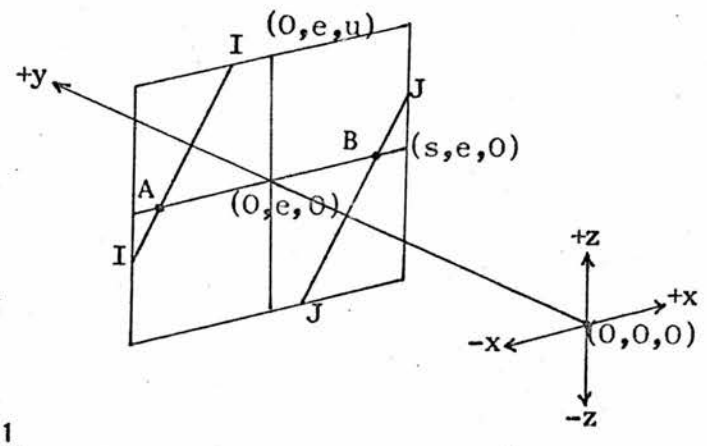


Diagram 1

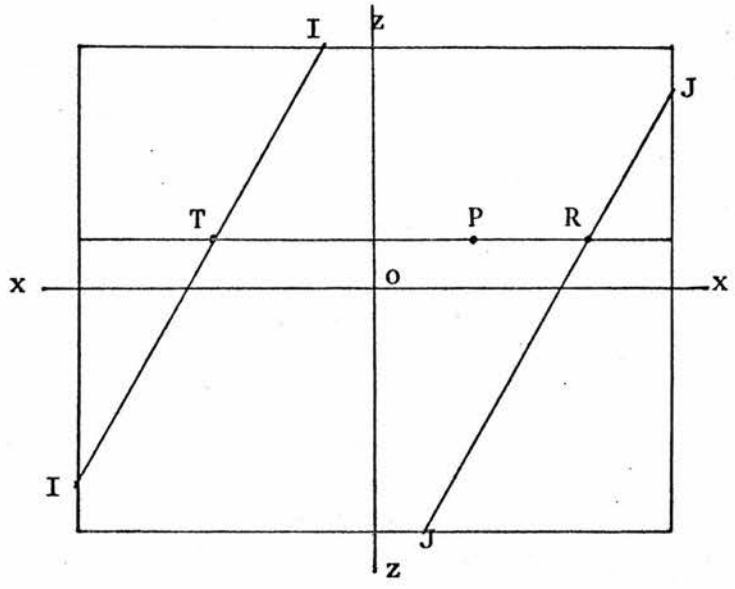


Diagram 2

Calculating K from Plane Coefficients

Figure 7

In Figure 7 the following relationships hold:

Line II: $0 = a.x + c.z + b.e + d$

Line JJ: $0 = a.x + c.z + b.e$

At point A, the intersection of line II and the X axis, $z = 0$ therefore the coordinate of A is $((-b.e-d)/a, 0)$.

At point B, the intersection of the line JJ and the X axis the coordinate of B is $((-b.e)/a,0)$.

The distance between A and B along the X axis is therefore:

$-b.e/a + b.e/a + d/a = d/a$.

In diagram 2, for a raster line parallel to the X axis such that $z = zl$, at a point P whose coordinate is (xl,zl).

The coordinate of T is $((-c.zl-b.e-d)/a, zl)$.

The distance from P to T is $xl - (-c.zl-b.e-d)/a$.

Multiplying by a gives (a.xl + c.zl + b.e +d)/a.

The value of the K signal at T is O, the value of the K signal at R is 1. By simple proportion, the value at P will be TJ/TP which gives

$$K_p = (a.xl + c.zl + b.e)/d + 1$$

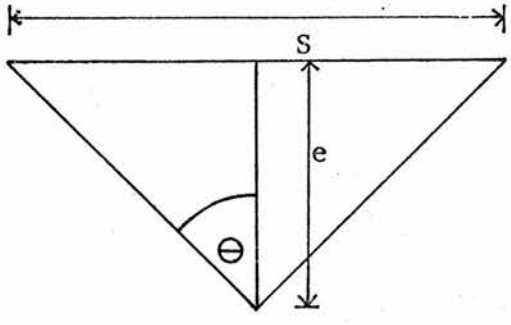
Consequently, the value at any point (x,z) on the picture plane is given by the expression

$$V = (a.x + c.z + b.e)/d + 1$$

The value of V is only of interest in the range -1 to +1. Though the values of a, b, and c will be less than 1, the values of x, z, e and d will depend on the size of the picture frame and the relative location of the plane. If the width of the picture frame is given as s then all these distances can be scaled by s, to give:

$$V = (a.(x/s) + c.(z/s) + b.(e/s))/(d/s) + 1$$

This means that, since the origin of the X and Z axes is at the centre of the picture frame, the values of (x/s) and (z/s) for a square frame will be in the range +½ to -½. The value of (e/s) depends on the angle which the picture frame subtends at the eye.



$$\frac{2e}{S} = \text{Cot } \Theta$$

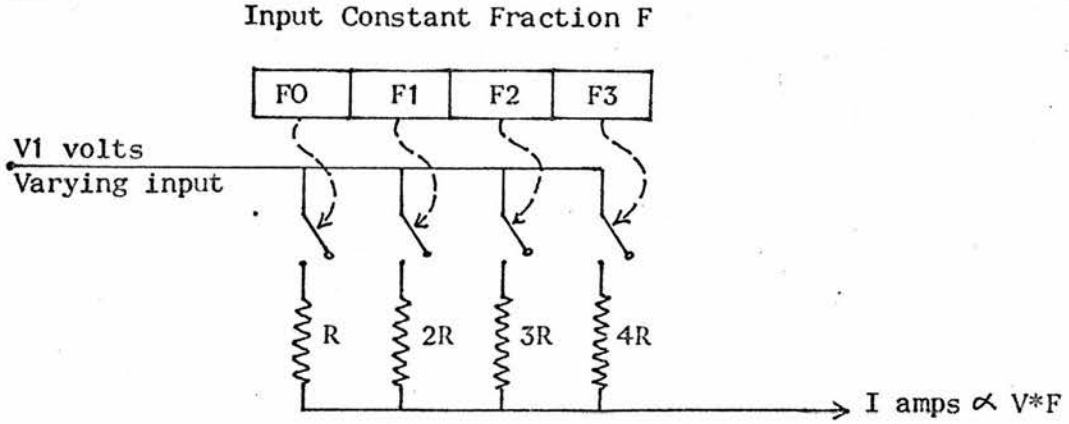
The Angle which the Picture Frame Subtends at the Eye

Figure 8

PARALLEL PROCESSOR USING ANALOGUE SIGNALS

Since the maximum value of Θ will rarely be greater than 45^{o} or less than 30^{o}, the Cot of these angles lie between 1.0 and 1.7 respectively, the value of e/s will usually be in the range 0.5 to 0.85. This means that all the values in the expression can be expressed in the analogue range of voltages except d/s.

The expression for V can be evaluated in several stages. If a, c, x/s, z/s, b.e/s, and s/d are created digitally, then subsequent calculations will be made up from the operations: two multiplies, a three number addition, another multiply, and finally a two number addition. In calculating a.x/s, a will be a constant during the processing for a single picture frame, whereas x/s will vary continuously as the device scans each consecutive raster line. This operation is very conveniently carried out using a Multiplying Digital to Analogue Converter, (MDAC). This circuit shown schematically in Figure 9 allows the digital constant to be continuously multiplied by a varying analogue value.

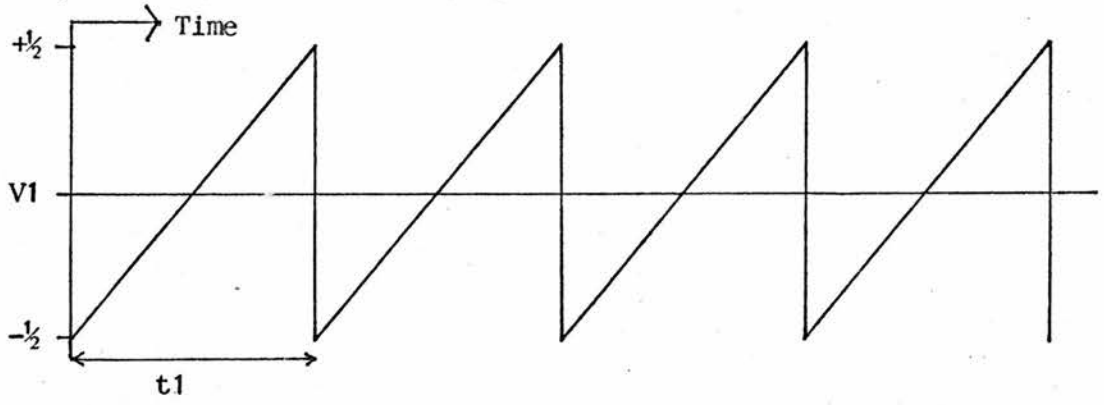


Multiplying Digital Analogue Converter

Figure 9

If the value of V1 represents x, then for a single raster line V1 will

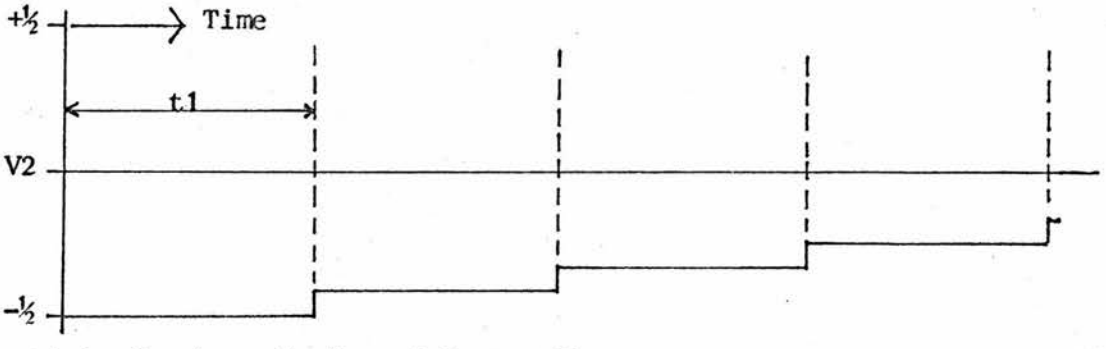
be a ramp, and for several lines will be a saw toothed signal, shown in Figure 10 . If the value of a is set up as the digital fraction F, then the output I is given by the relationship: I = 2F.V1/R. In other words, the output current is proportional to the value a.x.



Signal to Create a Raster Scan

Figure 10.

A similar arrangement will give the value of c.(z/s), except that in this case, the value of z must be held constant while the x value is traversing the screen. This requires a stepped ramp as input for z, shown in Figure 11.

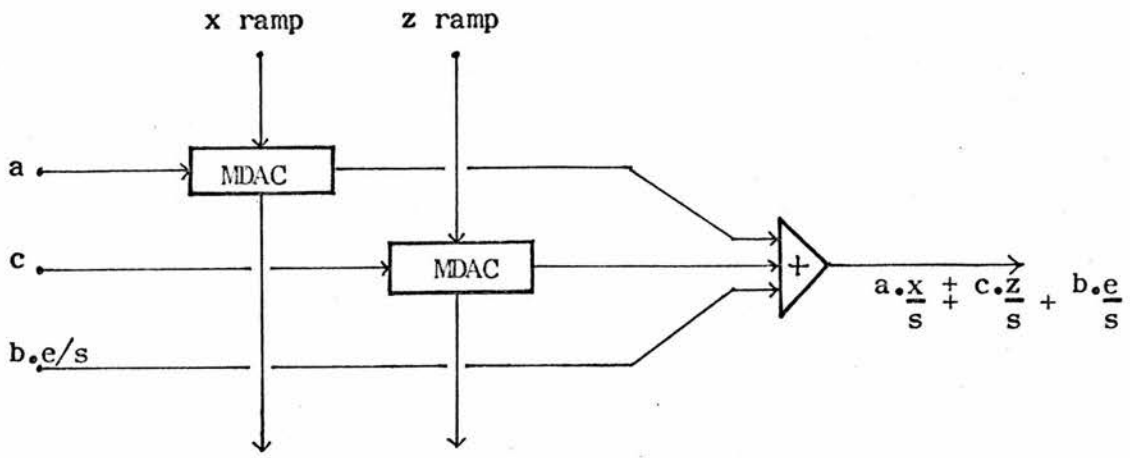


Signal to Create a Series of Raster Lines

Figure 11.

A circuit which will create the first stage of the calculation of V is shown in Figure 12.

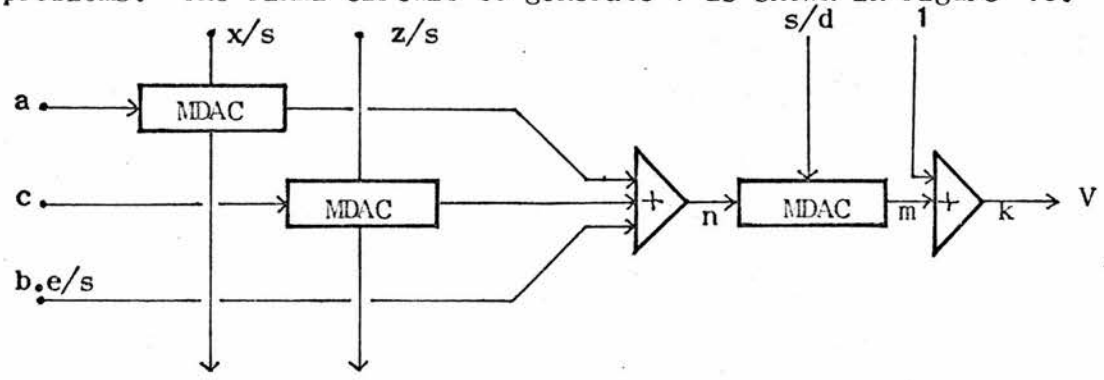
PARALLEL PROCESSOR USING ANALOGUE SIGNALS



Calculating V Stage One

Figure 12

The next stage is to multiply the output of the first stage by s/d and then subtract 1. At present this is assumed to be possible using another MDAC, though the range of values s/d can take may well create problems. The final circuit to generate V is shown in Figure 13.

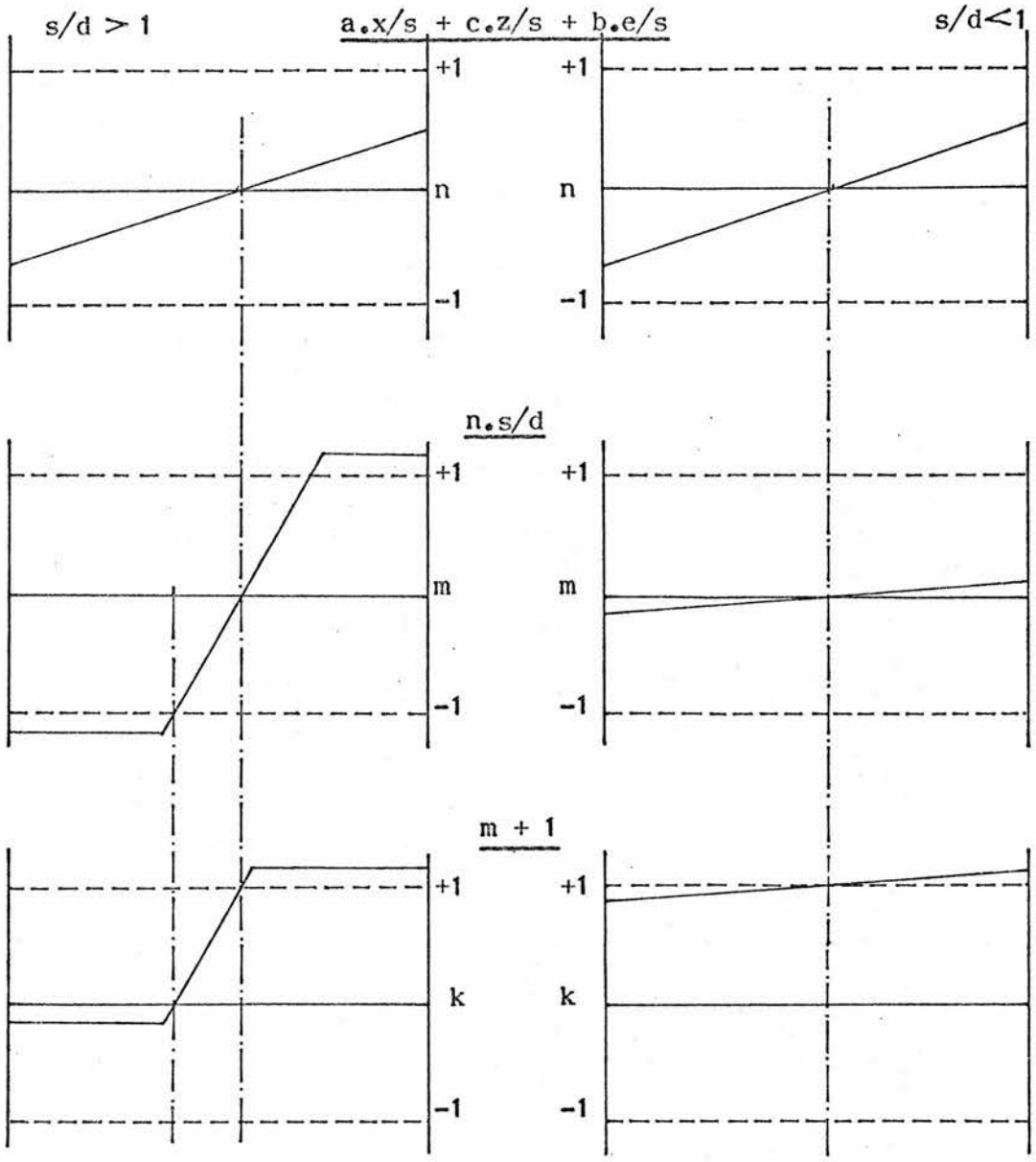


Circuit to Calculate V Stage I and II

Figure 13

The various modifications to the input signals to create V are shown in Figure 14.

PARALLEL PROCESSOR USING ANALOGUE SIGNALS



Generation of the Signal V

Figure 14

The first stages of this processing unit were simulated using a hybrid computer. The circuit used is described in more detail in Appendix A. The output from two simulated input conditions are shown in Figures 15. and 16 . The three stages in the creation of V: n,m, and k are

PARALLEL PROCESSOR USING ANALOGUE SIGNALS

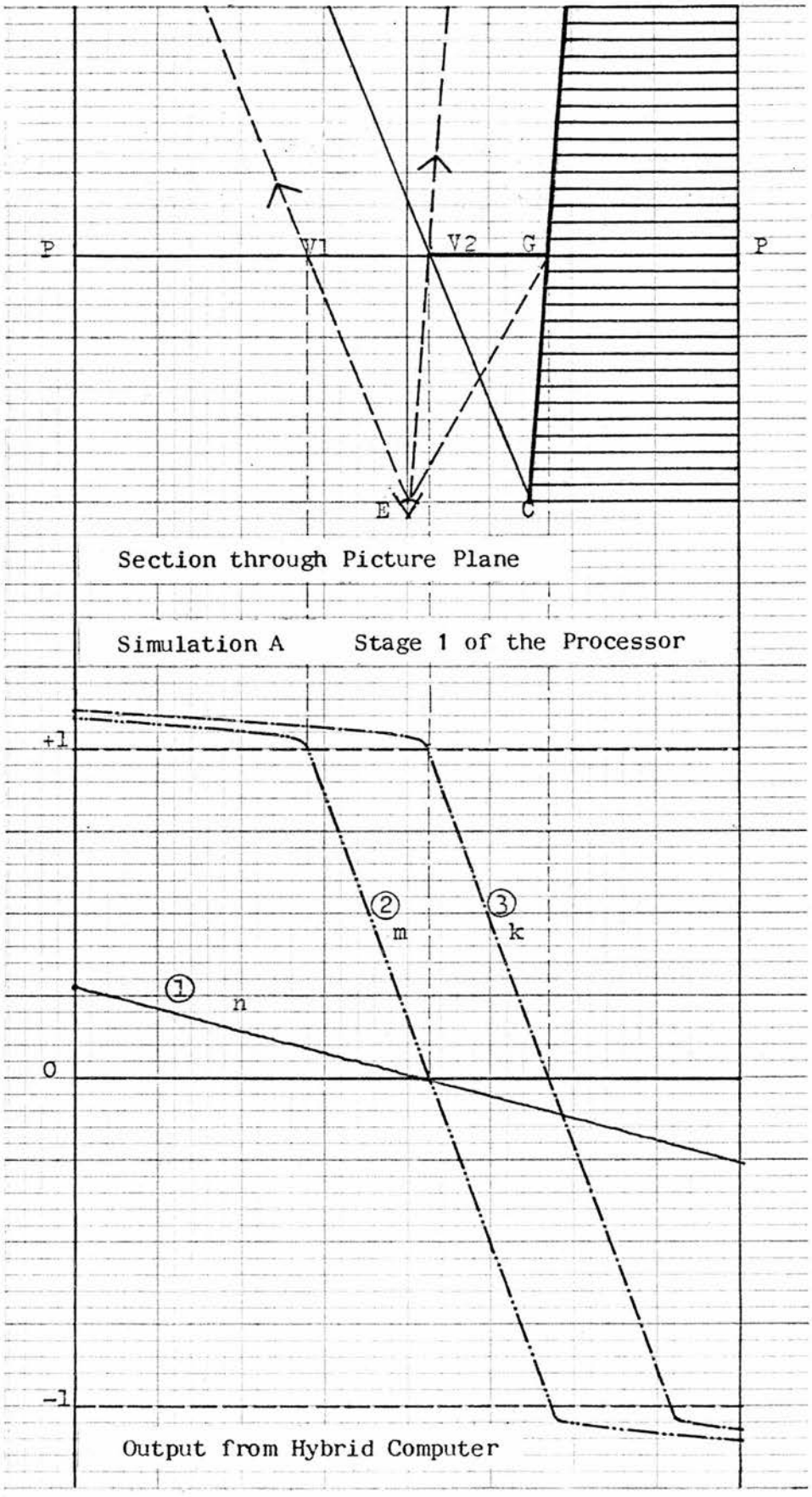


Figure 15

PARALLEL PROCESSOR USING ANALOGUE SIGNALS

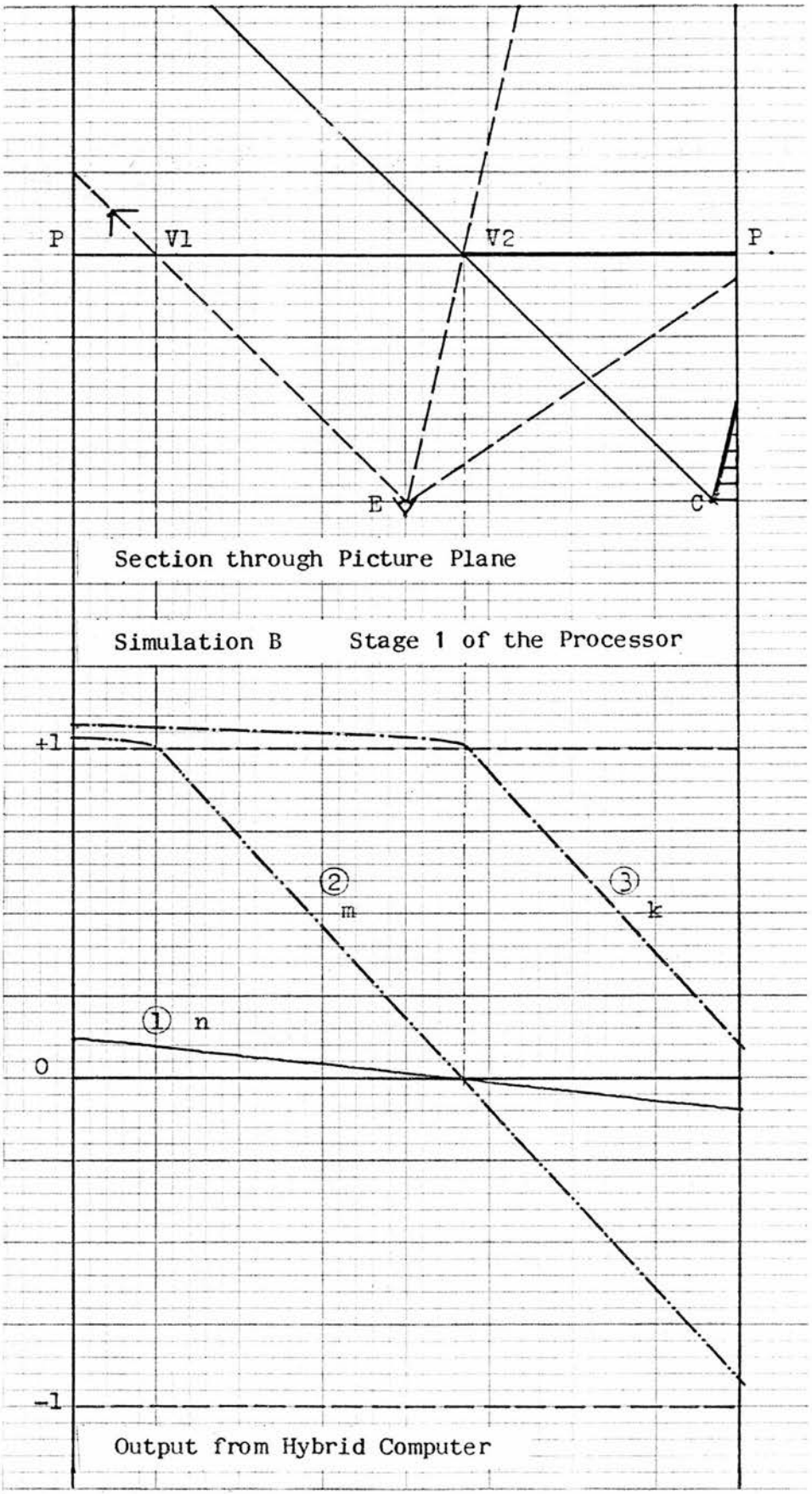
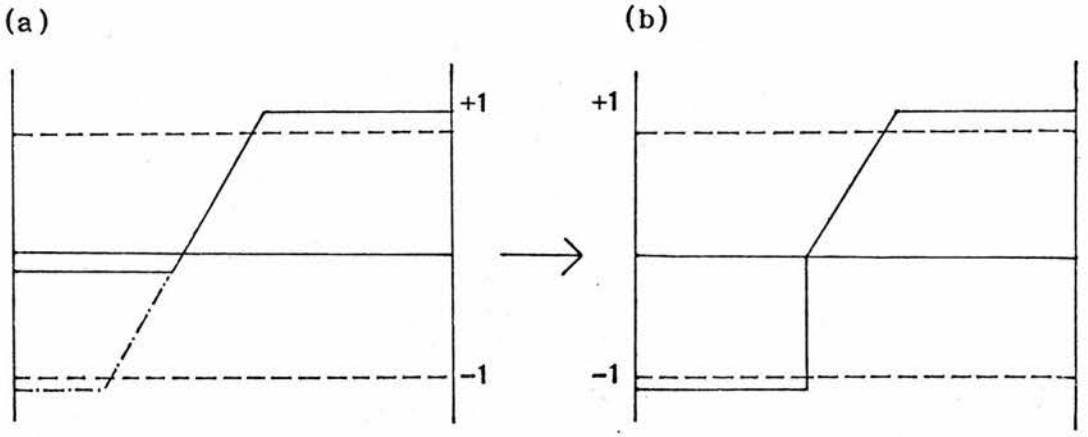


Figure 16

labelled and can be referenced to Figure 14. The lower diagrams in Figures 15 and 16 show the corresponding output from the machine's plotter, the upper diagrams show a reconstruction of the relationships between the eye, the picture plane and the displayed plane, which would have created these signals. In Figure 15 V2G represents the projection of the plane on the picture plane, V1 represents the position of a parallel plane which passes through the position of the eye, and PP and E represent the picture plane and the eye respectively.

Once the signal V had been created, the next stage was to carry out the correction procedure where the signs of T1 and TO are opposite. In such cases the FORTRAN simulation program converted the value of V to ± 1. In Figure 17 this correction is shown by the conversion of signal (a) to signal (b).

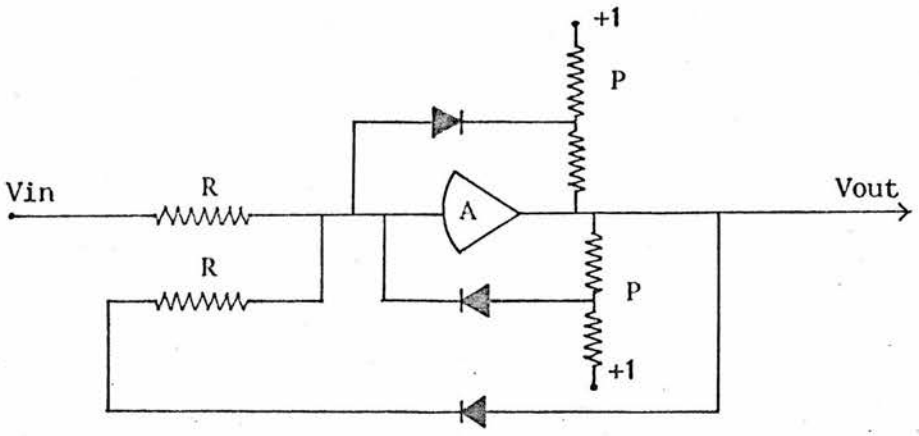


Correction of Signal V

Figure 17

A circuit was constructed to carry out this operation and its general structure is shown in Figure 18. The relationship between Vin and Vout necessary to create the modified signal is shown in the output from the

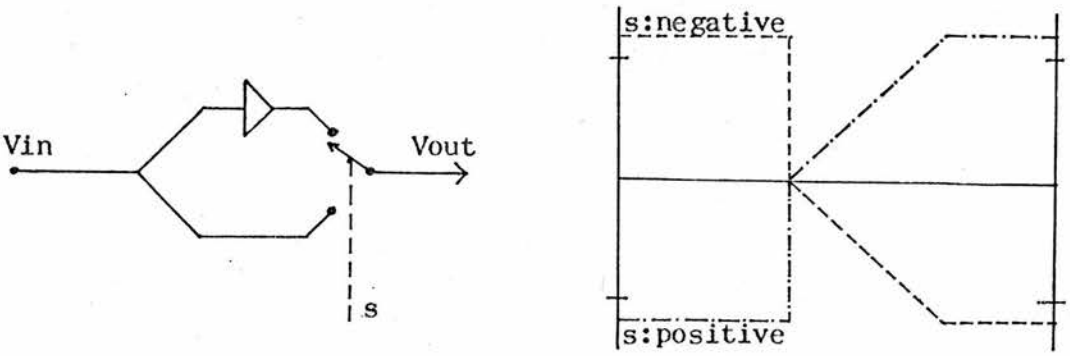
hybrid computer shown in Simulation C in Figure 20. The conversion shown in Figure 17, however, assumes that the plane is positive



Circuit to Correct Positive Plane Signals

Figure 18

relative to the viewing point. If the plane is in fact negative relative to the viewing point then the signal V has to be inverted. Initially this would be achieved using the externally controlled switch shown in Figure 19.



Creating Negative Plane Signals

Figure 19

At a later stage in the design such an arrangement would have to be

PARALLEL PROCESSOR USING ANALOGUE SIGNALS

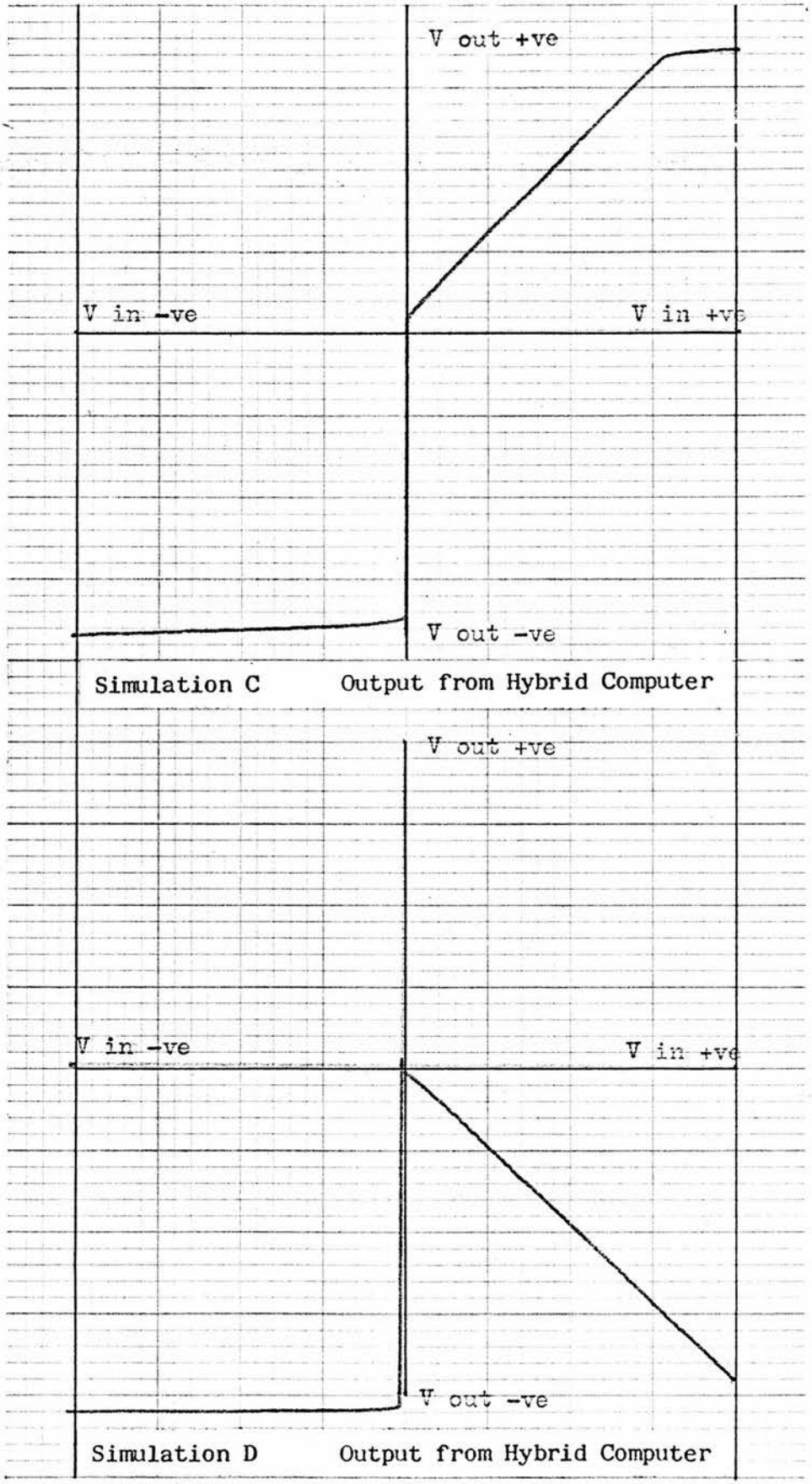
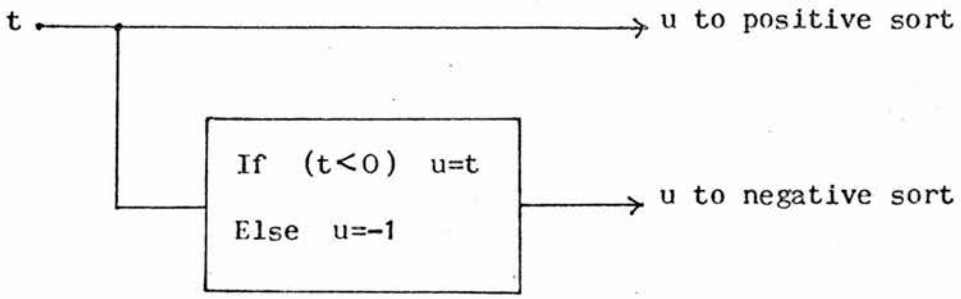


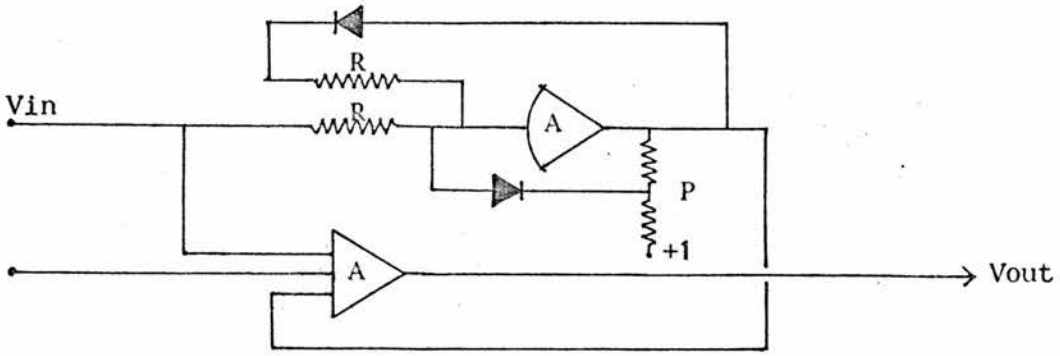
Figure 20

reconsidered, because if real time movement is attempted in the display, then a positive plane may well become negative. This change would require an internally controlled switching operation. Once the positive, or the inverted negative signal had been created it had to then be passed to the negative and positive sorting circuits. Since both these circuits can be set up to seek maximum values it is possible to pass both positive and negative signals to the positive sort, but only the negative signals can be passed to the negative sorting circuit. This means that a selection circuit has to be included in the input line to the negative sort as shown in Figure 21.



Selection of Values to be Passed to the Negative Sort.

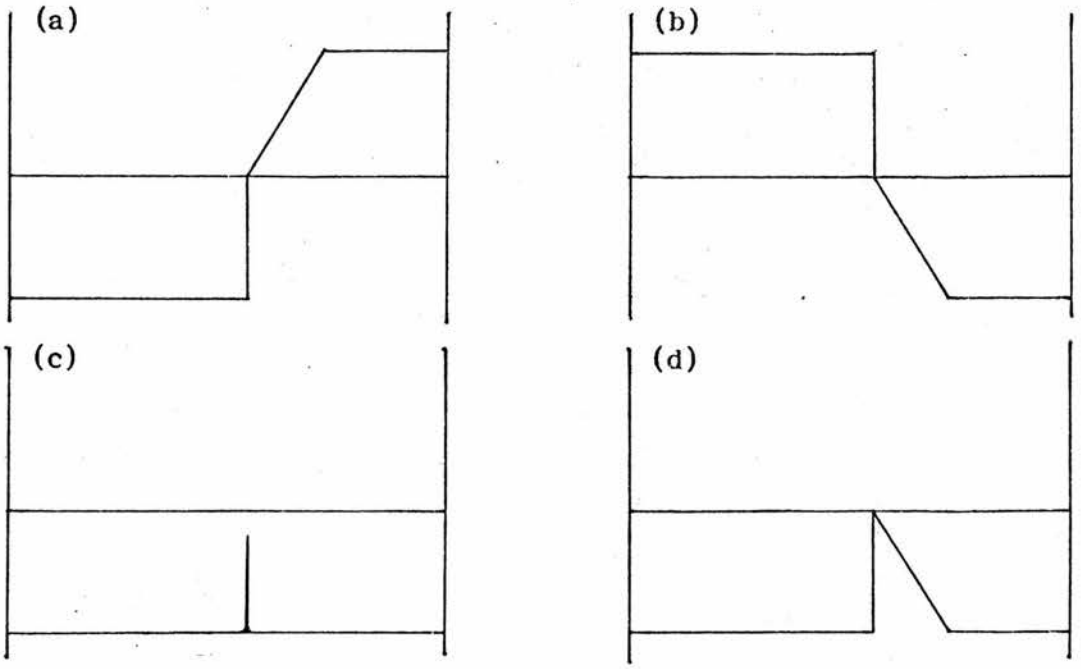
Figure 21



Circuit to Select Negative Values

Figure 22

The relationship between Vin and Vout for the circuit shown in Figure 22 is given in Figure 20 in Simulation D. The effect of using this circuit is shown in the diagrams in Figure 23 . (a) and (b) represent the two forms of input signal expressed against time, (c) and (d) show the output signals which correspond to them and which would be passed to the negative sorting circuit. It can be seen that signal (c) should have been a constant value of -1 units, but it was impossible to prevent the spike shown in the diagram from occurring.

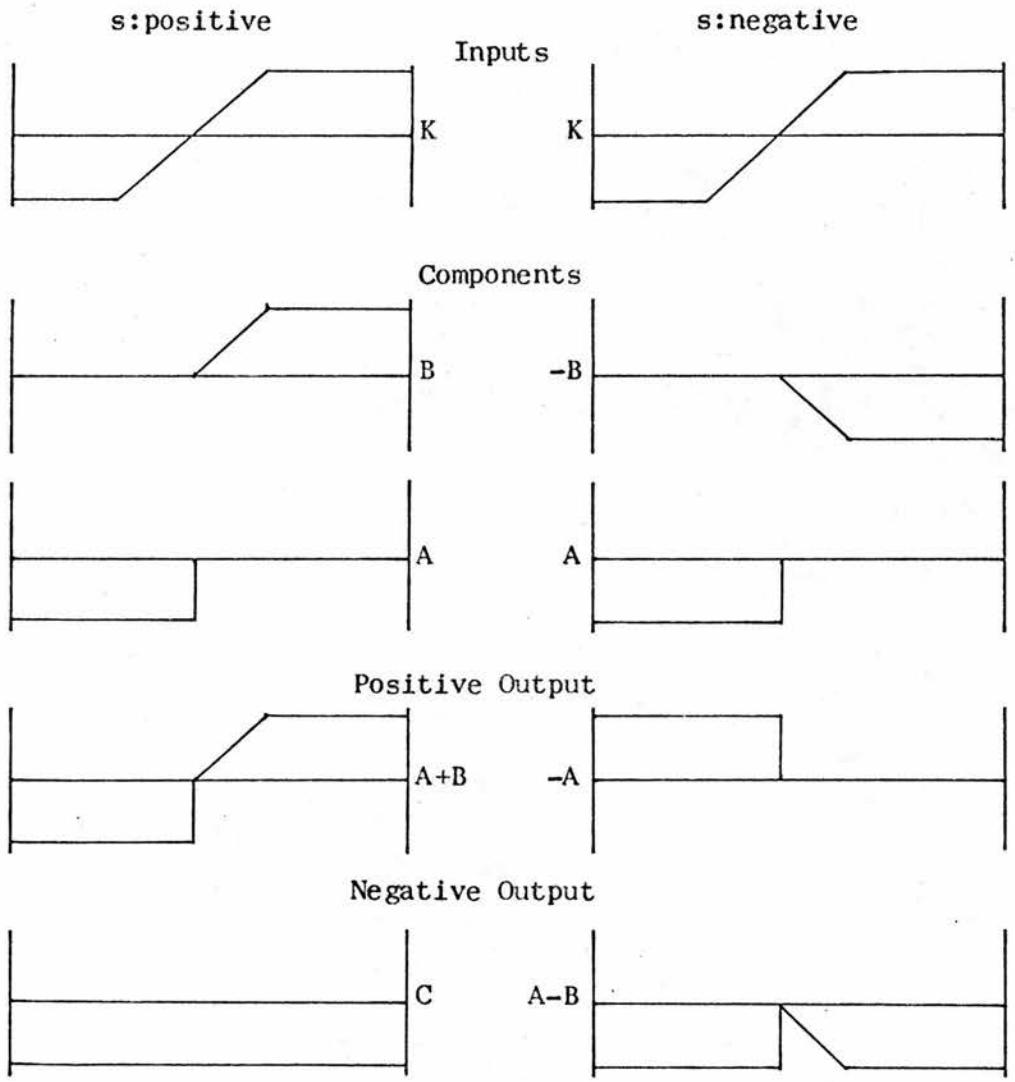


Signals Passed to the Negative Sort

Figure 23.

The first stage of this investigation was an 'additive' design exercise, and the next stage was to review the whole arrangement. Taken as a whole it was clear that the layout of the circuits used was not the simplest way of obtaining the required outputs, but the experienced gained in this exploration had defined the problem more clearly. This was to have appropriate signals entering both the positive and negative sorting

circuits for each plane unit. In other words when the input signal was positive some value still had to be passed to the negative sort, which would allow the correct outputs to be obtained from the subsequent sorting operations. An attempt to rationalise what had been done was carried out using the diagrams shown in Figure 24.

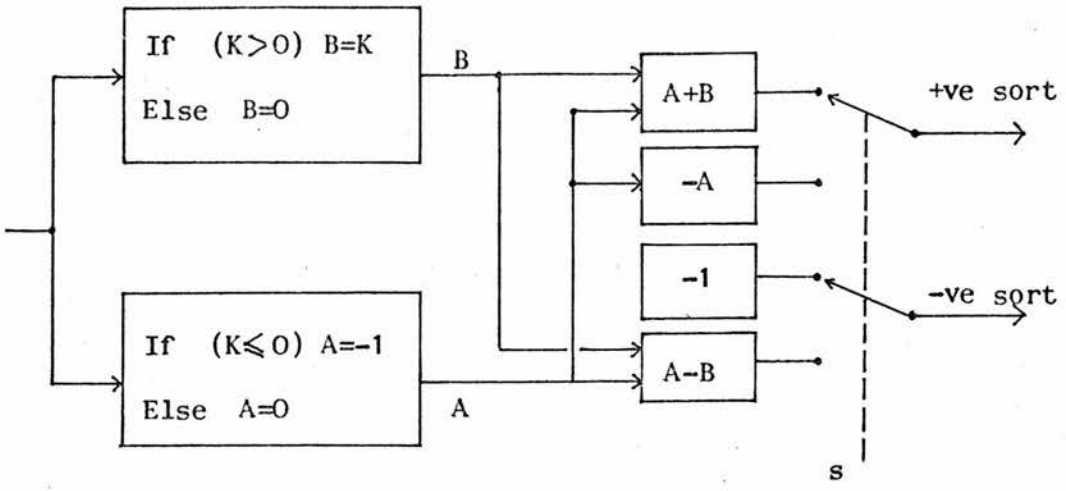


Positive and Negative Signal Components

Figure 24

A simpler schematic circuit for creating these outputs is shown in Figure 25.

PARALLEL PROCESSOR USING ANALOGUE SIGNALS



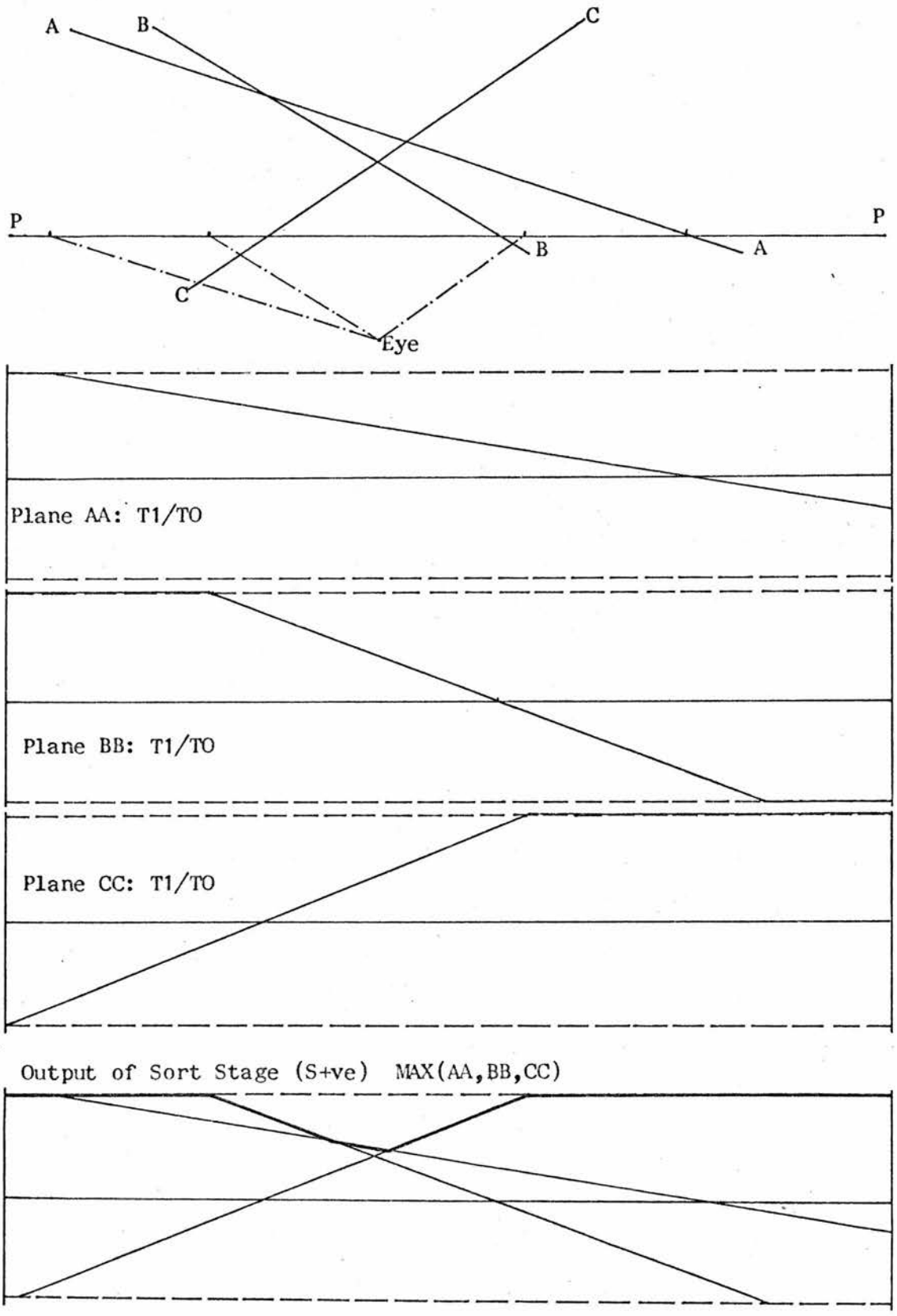
Creating Inputs to the Sorting Circuits

Figure 25

Even this alternative created difficulties. The sudden switching from -1 to 0 or from +1 to 0 in all the output signals except for C in Figure 24 , was difficult to implement accurately at the high speeds required. Consequently the whole procedure was re-examined at a more basic level.

The most important simplification came from distinguishing between positive and negative signs used to indicate front and back planes, and positive and negative signs which resulted in the position of the plane relative to the viewing point and raster point. In Figure 24 it can be seen that for a positive plane the positive values of V correspond to sections of the surface in front of the picture plane, while negative values correspond to sections of the plane on the same side of the picture plane as the eye. The difficulties which led to the need for the correction circuit resulted from these signs being reversed for a negative plane. In Figure 26 the intersection of three positive or front planes are considered and the required output signal for them shown.

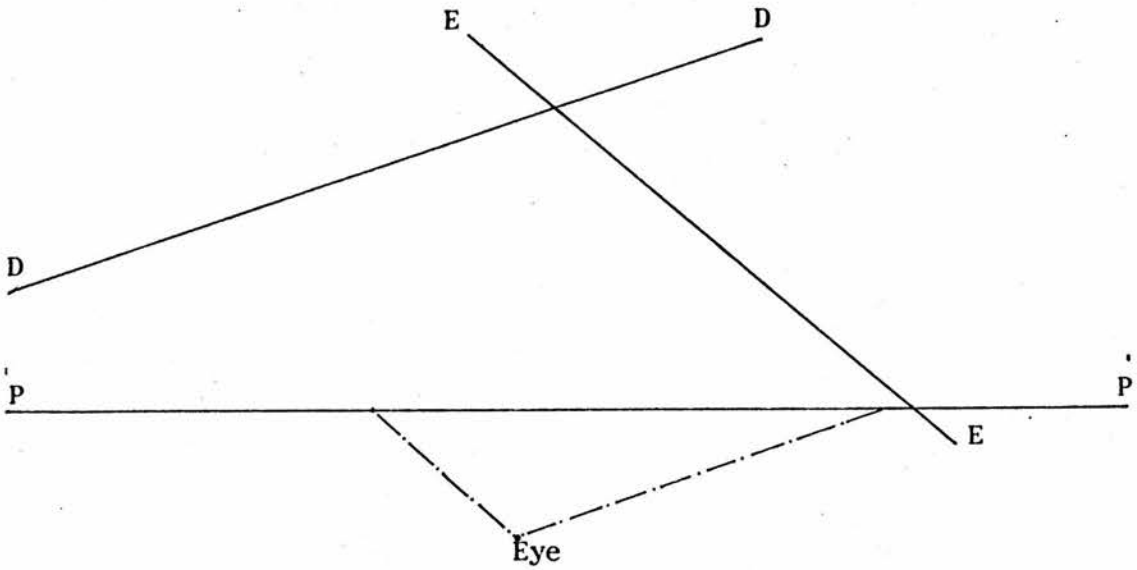
A PARALLEL PROCESSOR USING ANALOGUE SIGNALS

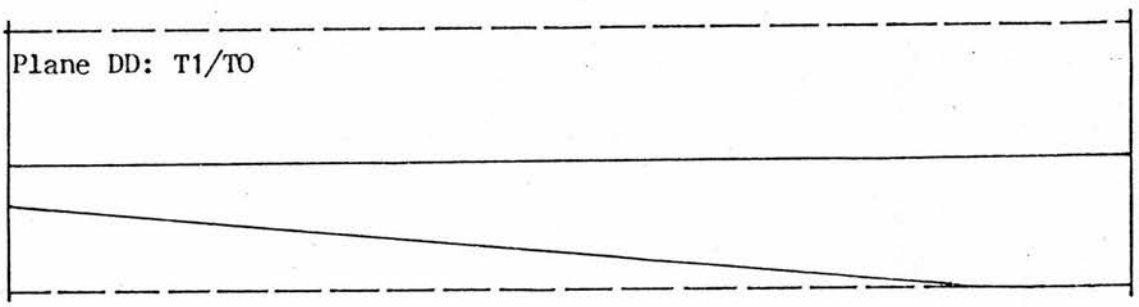


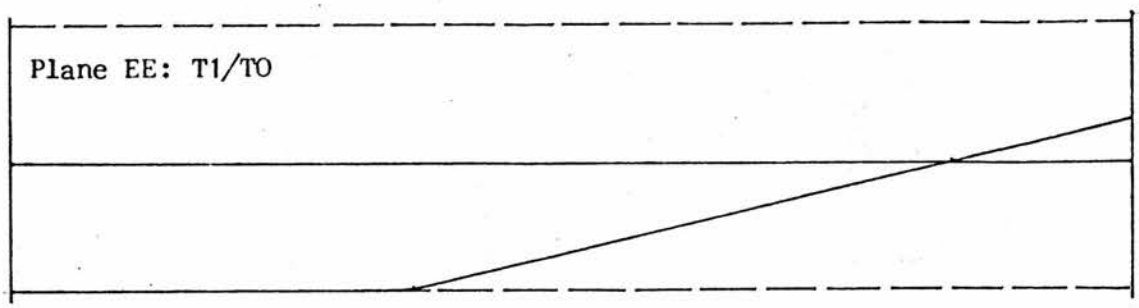
Front Planes

Figure 26

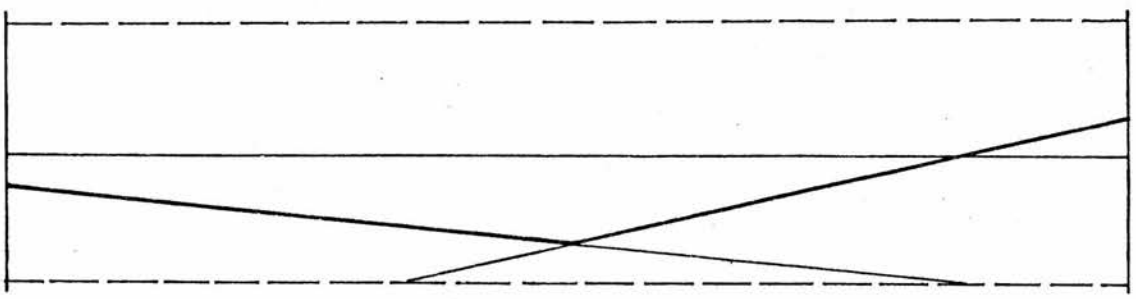
A PARALLEL PROCESSOR USING ANALOGUE SIGNALS







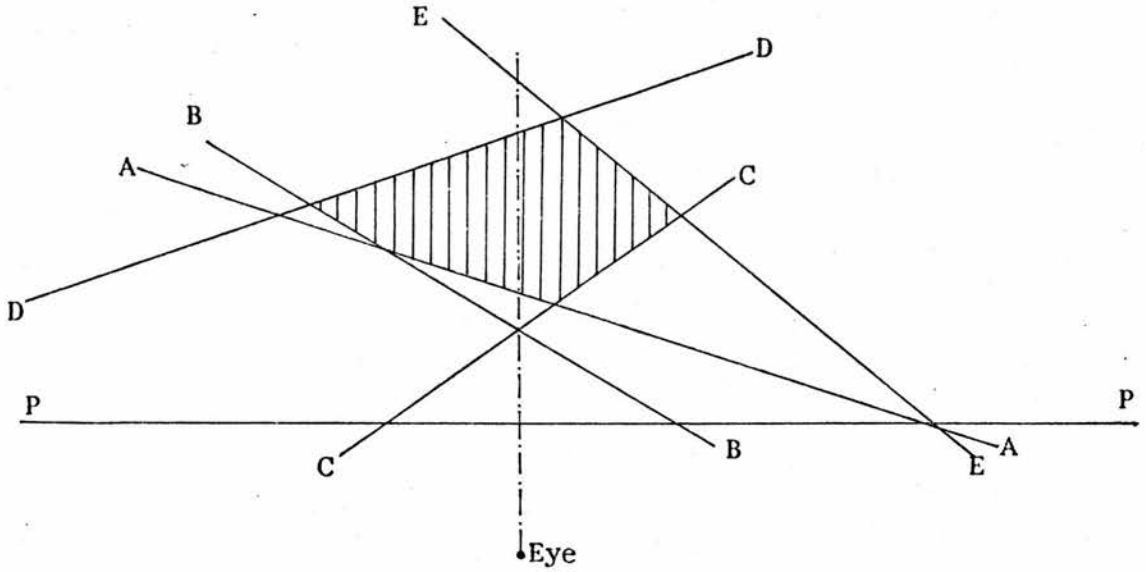
Output of Sort Stage (S-ve) MAX(-DD,-EE)

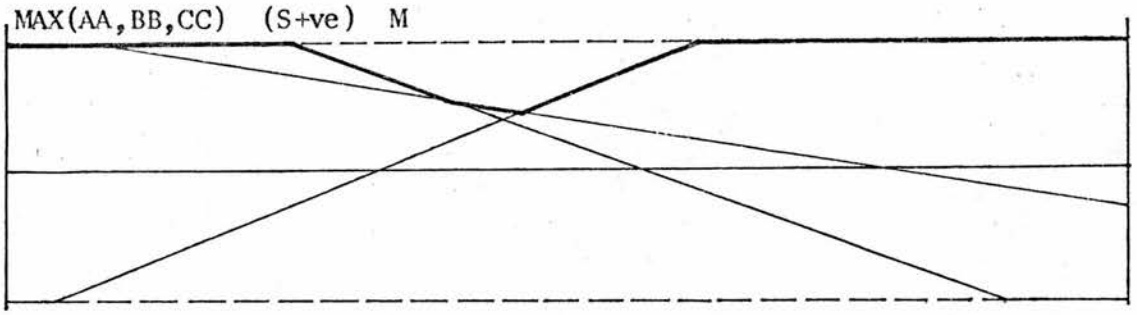


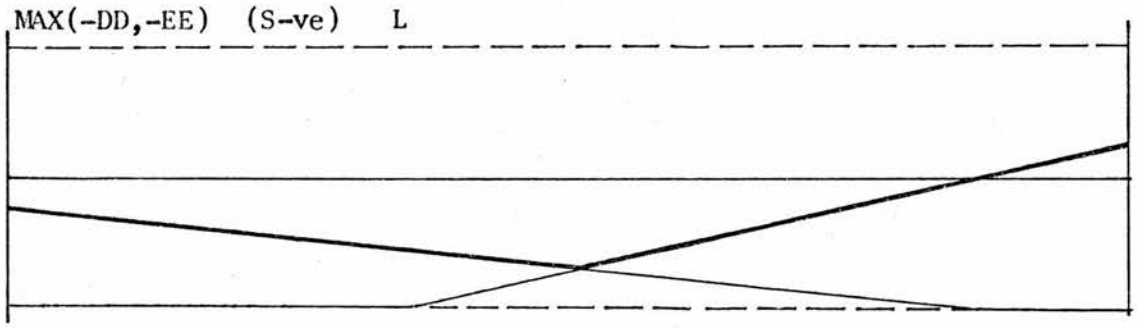
Back Planes

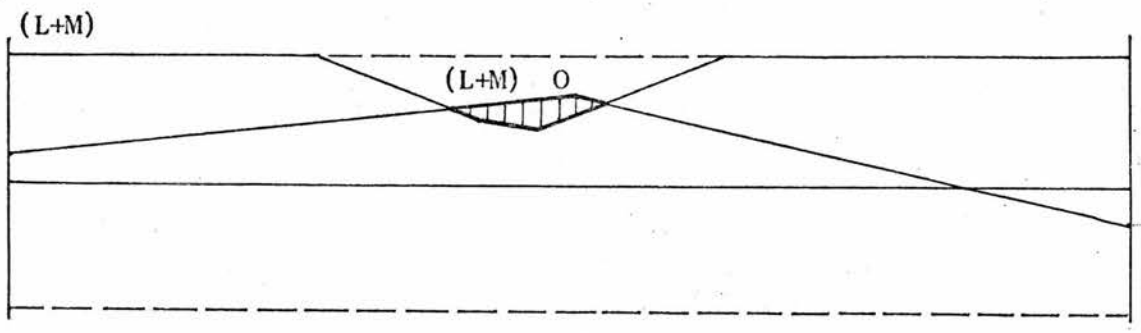
Figure 27

A PARALLEL PROCESSOR USING ANALOGUE SIGNALS







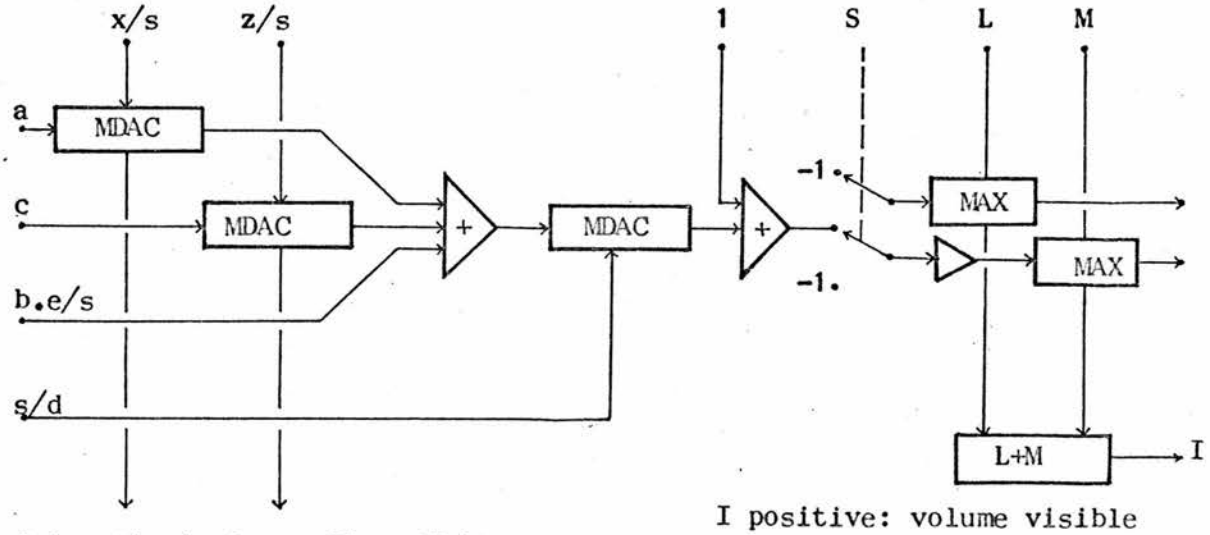


Merging Front and Back Planes to Define an Object

Figure 28

A PARALLEL PROCESSOR USING ANALOGUE SIGNALS

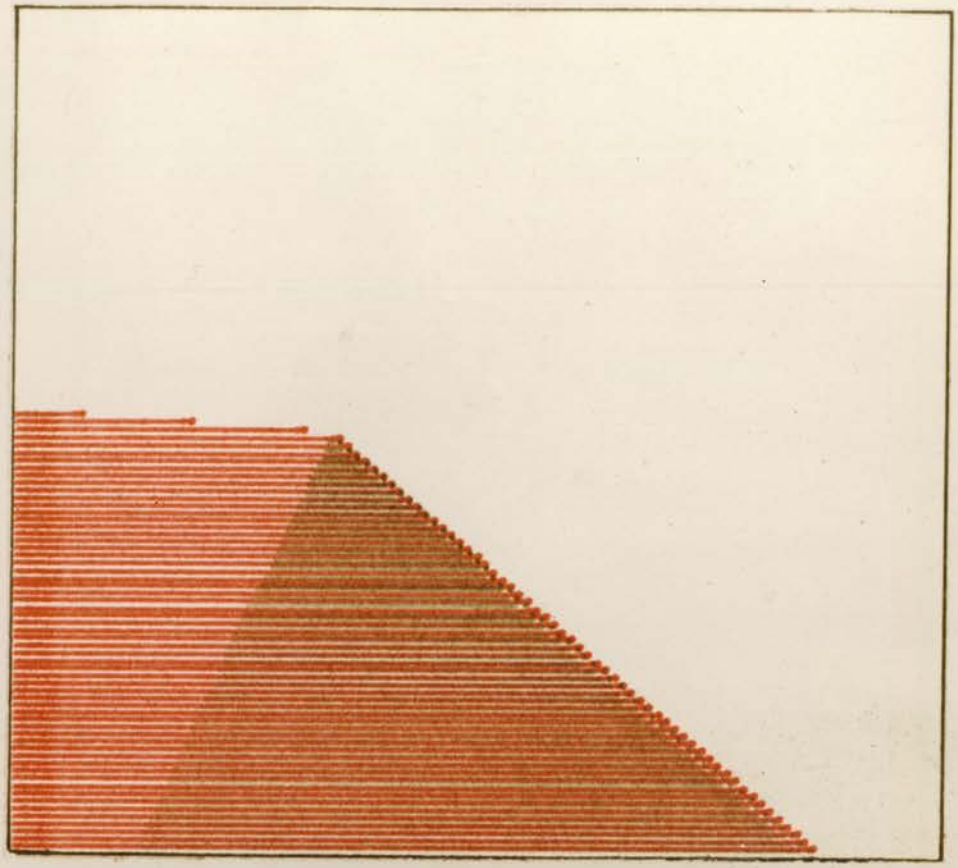
The three planes AA, BB, and CC in Figure 26 may be represented by the three 'plane templates' shown below them. The output value can then be created from these templates by simply selecting the surface which lies furthest in a perpendicular distance from the picture plane. In Figure 27 the two back planes are treated in the same way except that the templates are treated as negative. The results from these two operations are the two signals Max(AA,BB,CC) and Max(-DD,-EE). These two composite templates represent the front and back surfaces of a volume of intersection of AA, BB, CC, DD, and EE. In Figure 28 the result of overlaying them on the same side of the picture plane is shown. If the signal for the front planes is called L and the signal for the back planes is called M then the arithmetic sum L+M for the values at any point on the display surface can be used to define the visibility of the object. Where L+M is positive then the object is visible, where L+M is negative then the object is invisible. This approach results in a simplification of the plane unit to the arrangement shown in Figure 29.

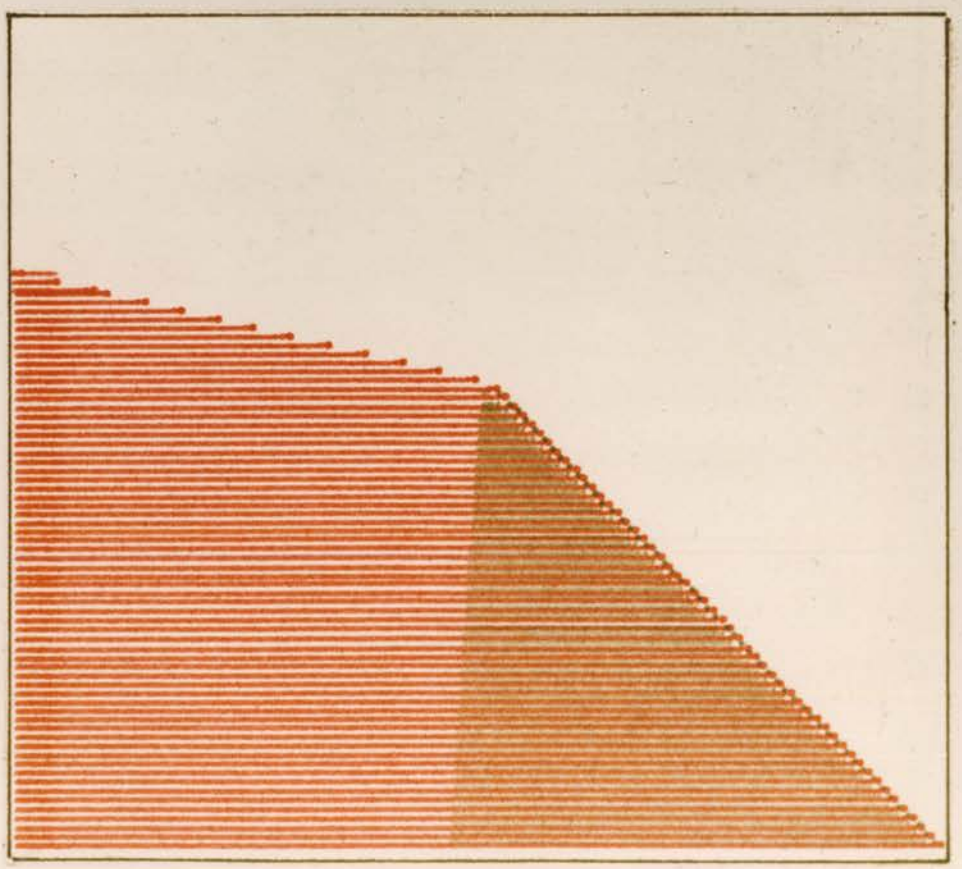


Schematic Analogue Plane Unit

I positive: volume visible
I negative: volume invisible

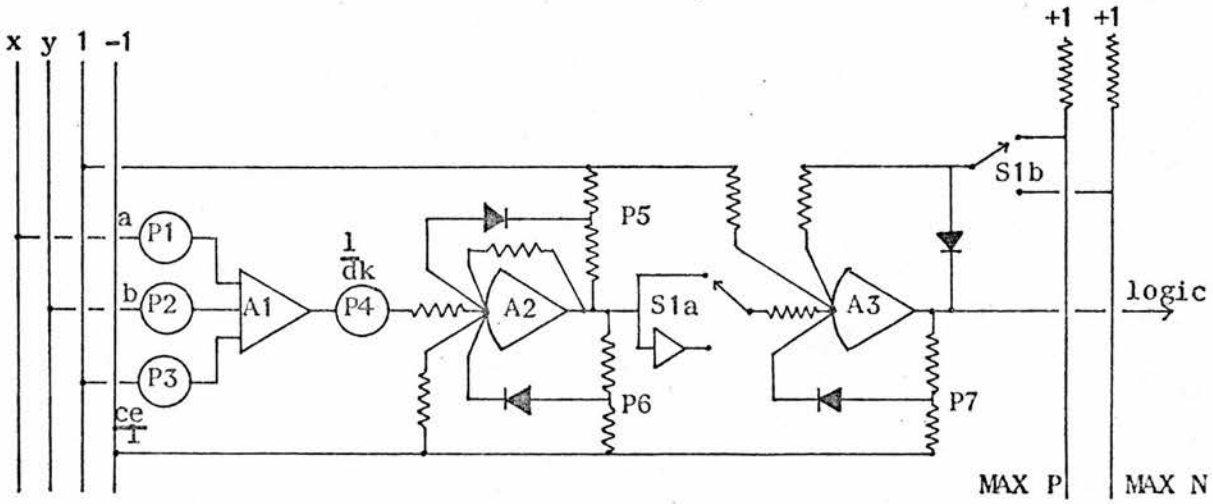
Figure 29.





A PARALLEL PROCESSOR USING ANALOGUE SIGNALS

It was the circuit shown in Figure 29 which was simulated on the hybrid computer. A brief description of this work is presented in Appendix A, by J. Downie who built the necessary circuits. The resulting display was limited to the intersection of two planes. Though the display was created on a cathode ray tube, the hardcopy given in Plate 3 was produced on a small plotting device.



Analogue Plane Unit: Simulation

Figure 30

The circuit used to simulate one plane unit is shown in Figure 30 The Multiplying Digital to Analogue Converters were replaced by potentiometers so the input variables were not externally controlled by digital values. However, the potentiometers allowed the form of the displayed objects to be adjusted in real time, and the two examples shown in Plate 3 are the result of two different settings for the potentiometers.

Conclusions

Two lines of development became apparent as a result of this experiment

using a hybrid computer. The first depended on the use of the MDAC's and consisted of a simple way of generating the movement of objects being displayed. The second consisted of a way of creating displays of objects with curved surfaces, by using curved templates in the operations shown in Figure 28.

It was assumed, when this analogue section of the display processor was being designed that the rotation and translation of the scene or components of the scene relative to the position of the viewing point, would be performed between frames. This would be done by changing the parameters used for defining each plane. What this assumption means in practice is that between frames all the inputs A, B, C ..... would have to be recalculated even if the only change is that the viewing point is moved to position x_1, y_1, z_1 and that the planes are still the same relative to the previous origin. This seemed impractical, at best it means that if the values of A, B, C, ... are calculated during one frame in readiness for the next, then there must be a parallel set of storage spaces for the new parameters, while the old ones are being used, which can be switched to the appropriate plane unit at the end of the current frame.

The representation of a plane in this part of the machine consists of the digital coefficients: a, b, c, d. The transformations of these coefficients for rotation and translation of the plane about the origin can be given by the following matrix operations:

Rotation about the x axis by an angle θ

$$[a',b',c',d'] = [a,b,c,d] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

so that: $[a',b',c',d'] \equiv [a,\ \cos\theta.b+\sin\theta.c,\ \cos\theta.c-\sin\theta.b,\ d]$

Rotation about the y axis by an angle α

$$[a',b',c',d'] = [a,b,c,d] \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

so that: $[a',b',c',d'] \equiv [\cos\alpha.a-\sin\alpha.c,\ b,\ \sin\alpha.a+\cos\alpha.c,\ d]$

Rotation about the z axis by an angle β

$$[a',b',c',d'] = [a,b,c,d] \begin{bmatrix} \cos\beta & -\sin\beta & 0 & 0 \\ \sin\beta & \cos\beta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

so that $[a',b',c',d'] \equiv [\cos\beta.a+\sin\beta.b,\ \cos\beta.b-\sin\beta.a,\ c,\ d]$

Translation of the origin by dx, dy, dz

$$[a',b',c',d'] = [a,b,c,d] \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

so that: $[a',b',c',d'] \equiv [a,\ b,\ c,\ d+a.dx+b.dy+c.dz]$

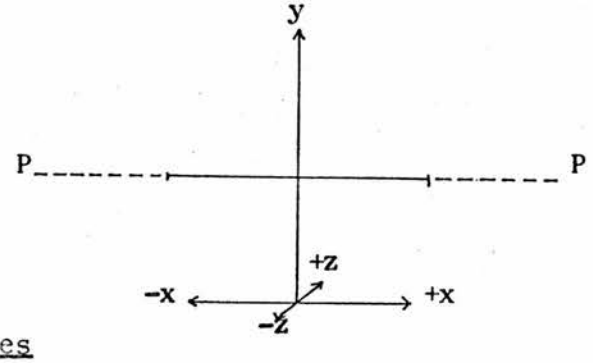
Where a transformation is required which is a multiple of any of these separate transformations, it can be represented by one matrix by multiplying together the appropriate component matrixes. Each of the transformations has an inverse created by putting $-\theta$, .. or -dx,.. into the relevant matrix form. It can be seen that during the display

of one frame the digital operation:

$$[a',b',c',d'] = [a,b,c,d] \cdot \begin{bmatrix} & \\ & T \\ & \end{bmatrix}$$

must be performed on all the plane coefficients making up the boolean expression model of the scene.

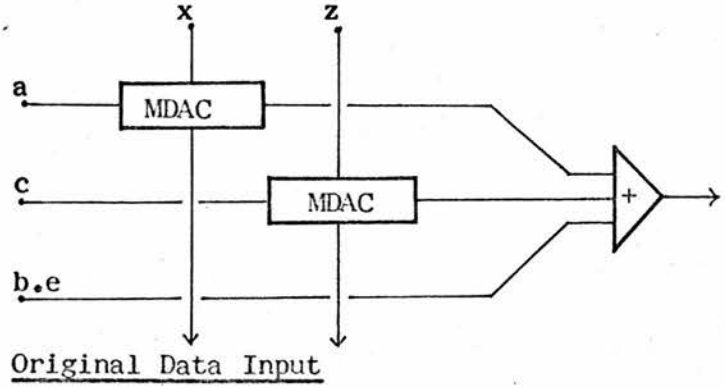
The form taken by the final version of the experimental machine makes it possible to simplify this task. The first stage of this simplification can be demonstrated using the diagram in Figure 31



Display Axes

Figure 31.

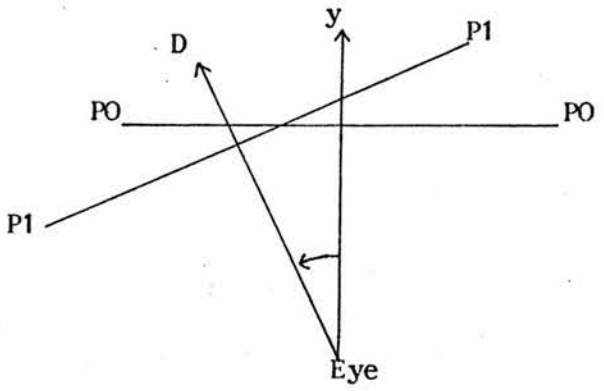
The original simplifying assumption which was made, was that the eye should always be at the origin and that the picture plane be parallel to the XZ axes plane, which resulted in the circuit shown in Figure 32



Original Data Input

Figure 32.

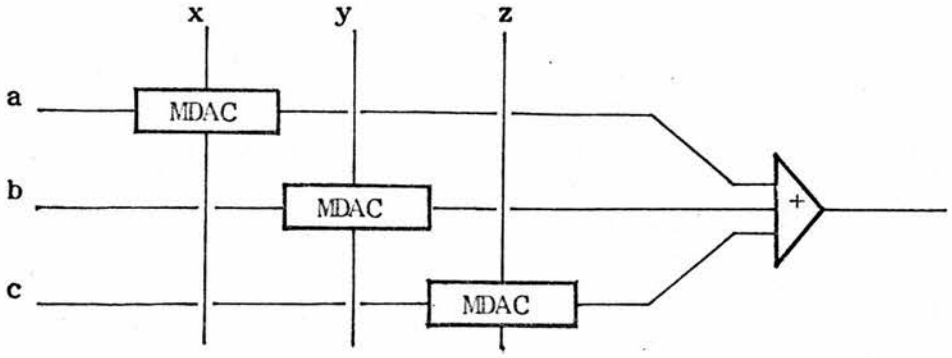
However if the picture plane is rotated to AA; with the new viewing direction towards D as shown in Figure 33 it is not necessary to



Rotation in the Viewing Direction

Figure 33

transform the plane coordinates; the same effect can be obtained by transforming the values of the raster line coordinates. This can be achieved using the circuits shown in Figure 34.

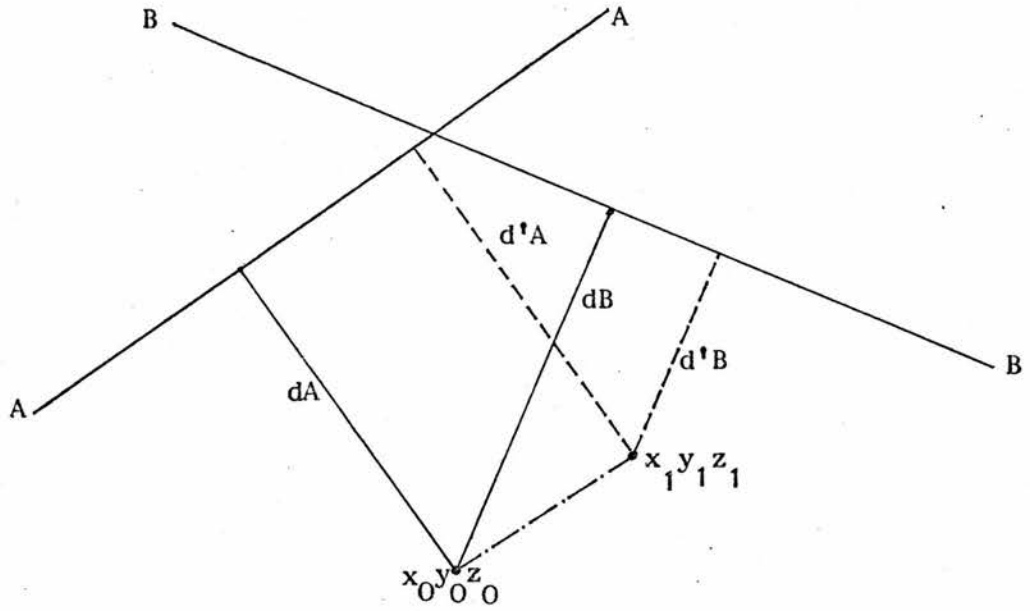


Extended Input Circuit to Allow Plane Rotation

Figure 34

The value of d: the perpendicular distance of the plane from the viewing point will not change if the direction of viewing is modified. The translation of the viewing point can be achieved by the appropriate modification of the value of d:

PARALLEL PROCESSOR USING ANALOGUE SIGNALS



Translating the Viewing Point

Figure 35

This is the justification for the form:

$$(a.x + b.y + c.z)/d + 1$$

since it permits a simple implementation of rotation and translation with minimum intermediate calculation. The effect of translating the eye position has already been calculated if it moves by distances x', y', z' then the new value of d becomes: $d' = a.x' + b.y' + c.z' + d$

The creation of curved surfaces is discussed more fully in the next chapter where alternative implementations of a digital display processor are considered. The basic idea is to produce curved surface templates in the operation shown in Figure 28. However, the principal problem is to find a way of specifying the nature of the curve in the input data, and then generating the curve at a fast enough rate.

CHAPTER 16

DEVELOPMENT OF THE DIGITAL PROCESSOR

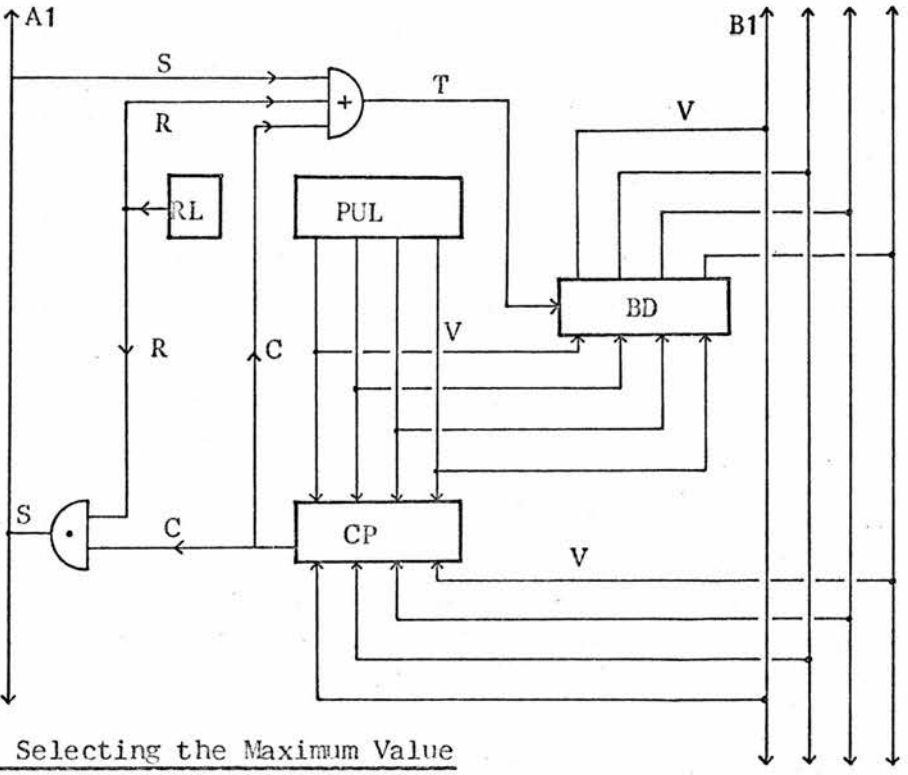
DEVELOPMENT OF THE DIGITAL PROCESSOR

During the simulation of the analogue processor described in the previous Chapter, new possibilities emerged which could be implemented using digital circuitry. In this final Chapter in this section, these ideas are briefly reviewed and the basic proposals for a digital processor which were the conclusion of this work are presented.

The first development which was relatively simple, resulted from the sorting process used in the analogue device. Only one line was used for each of the two buses in comparison with the many bit lines used in the equivalent digital operation. The output from the first stage of sorting was decided by the comparison of the outputs from the Maximum seeking circuits linked to the positive or front facing plane units and the negative or back facing plane units. The final output to the next stage was only of interest if the value was positive; however, the original arrangement also allowed negative final results. The negative values were only employed to decide when to inhibit the positive outputs where the object did not exist. The same end result could be achieved by comparing the current value on the positive bus with all the individual outputs from the plane units holding back faces. If any of these comparisons indicated that the object was not visible then the output from the positive planes could be inhibited using a single line bus. The results from negative plane comparisons with the positive bus could be wire ored onto this bus.

Because the subvolumes represented in this first stage sort would be convex, a relatively simple arrangement allowed the sorting operation to be carried out, and the schematic arrangement to do this is shown

in Figure 1.



Stage I. Selecting the Maximum Value

Figure 1

In Figure 1 B1 is the main bus for comparing output values from a convex volume. A1 is the control bus for inhibiting output when the viewing ray passes outside the convex volume. PUL is the Plane Unit Latch which provides the next value to be tested for the next raster line position. BD is the bus driver which places the maximum value on the bus, but which can be switched off by the signal T when the unit in inhibited from giving output: either because it is not producing the current maximum, because it is a rear plane, or because other rear planes are inhibiting it. CP is the comparator which compares the value on B1 with the output of PUL. RL is a one bit latch which defines whether the plane unit holds a front or rear plane.

DEVELOPMENT OF THE DIGITAL DISPLAY PROCESSOR

The relationship between the signals R, S, C and T; and the state of the plane unit is defined as follows:

S = 0 Another Plane Unit is creating an inhibiting Signal.

S = 1 No other Plane Unit is creating an inhibiting Signal.

R = 0 Plane Unit contains a Rear Plane.

R = 1 Plane Unit contains a Front Plane.

C = 0 Plane value is less than bus value.

C = 1 Plane value is greater than or equal to bus.

T \ RC	00	01	11	10
S = 0	0	0	0	0
S = 1	0	0	1	0

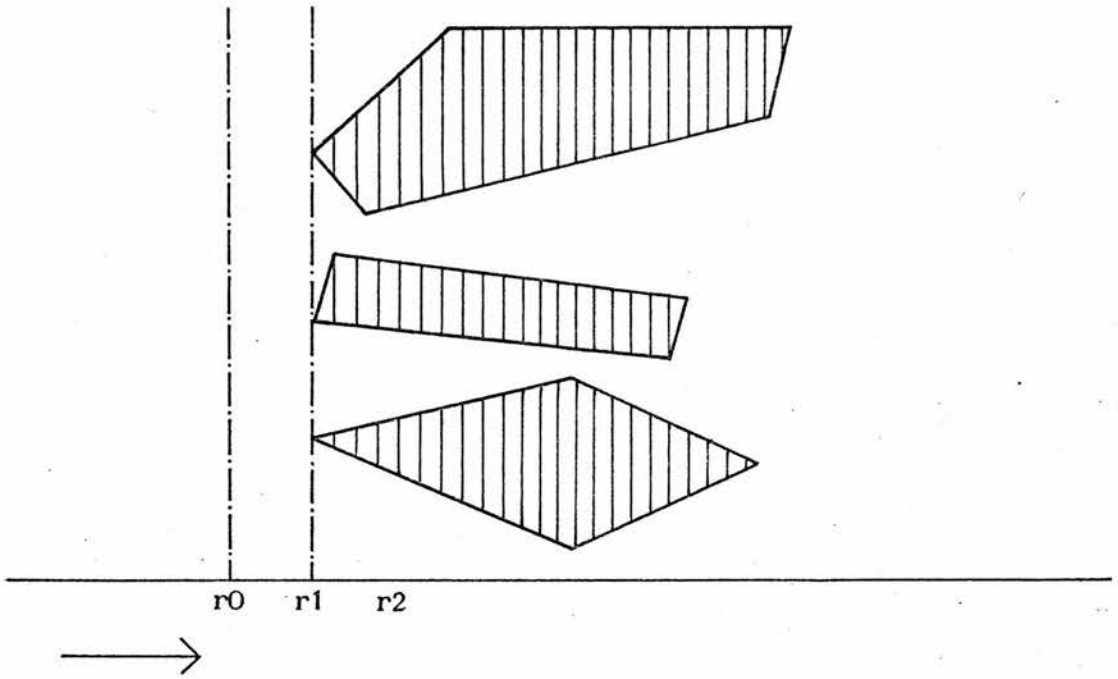
$T = R + C + S$

S \ R	0	1
C = 0	0	1
C = 1	1	1

$S = R \cdot C$

The next stage in development is to include the second comparison bus: B2. It is also possible to use a comparator in each plane unit to link the output of the PUL to this second bus (Figure 3). This will work for most situations but there is one situation which will create difficulties. Consider the relationship of the convex volumes shown in Figure 2.

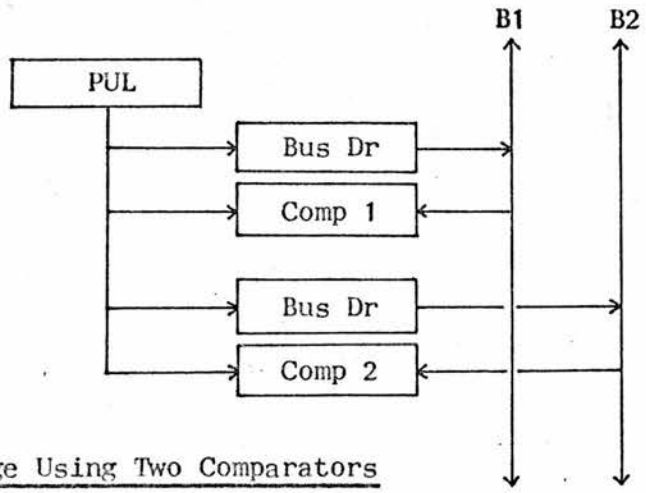
At raster point r0 all outputs from the plane units will be inhibited. On moving to raster point r1 three values, all less than the neutral value on B2 will compete to be switched through to the second bus. Since this will occur nominally at the same instant, and since there is no form of intercommunication between the plane units to arbitrate between them,



Three Competing Convex Volumes

Figure 2

there seems to be no way of preventing all three values being wire ored together on the bus B2. Except for this special case (for which some arbitration scheme would need to be found), the arrangement in Figure 3 would be sufficient to select the visible surface.



Bus Linkage Using Two Comparators

Figure 3

It was necessary to solve this special case before any further work was carried out on the layout in Figure 3 . If more than one value was wire ored together on bus B2, unless they were related for example in the way that 11011 and 01011 are related, then zero values in the larger number could create a number on the bus which would switch off all the values ored onto it.

A	001011
B	011010
Bus	001010

Both A and B are larger values than the value on the Bus; consequently, the comparison of the Bus value with the Plane Unit value will switch both A and B off for the next raster point. As soon as the Bus has returned to a neutral value, both these units will switch on again - so oscillating round an incorrect Bus value.

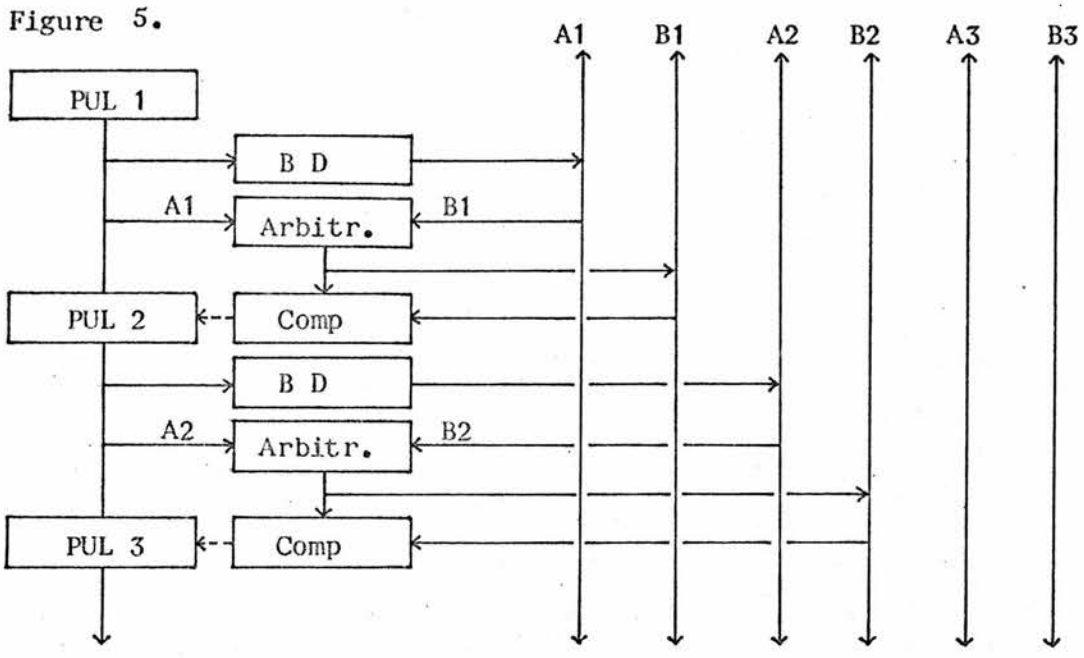
The solution proposed in Chapter 14, Figure 26 could be used to resolve this problem, but it meant too many duplicate components which would rarely be used. The problem was that a separate pair of buses had to be provided for each bit in the comparison word. In concept the scheme worked and could work fast enough to process raster points at the speed required, unlike the alternative shown in Figure 25 of Chapter 14. Consider the four numbers shown in Figure 4.

Bus 1	Bus 2	Bus 3	Bus 4	
00011111	00011111*	00011111	00001111	A
00101101	00111111			B
00110101	00111111			C
00010111	00011111*	00010111	00000000	D*
00000101	00011111	00010111	00000000	Bus

Comparison of Four Numbers to Find the Smallest

Figure 4.

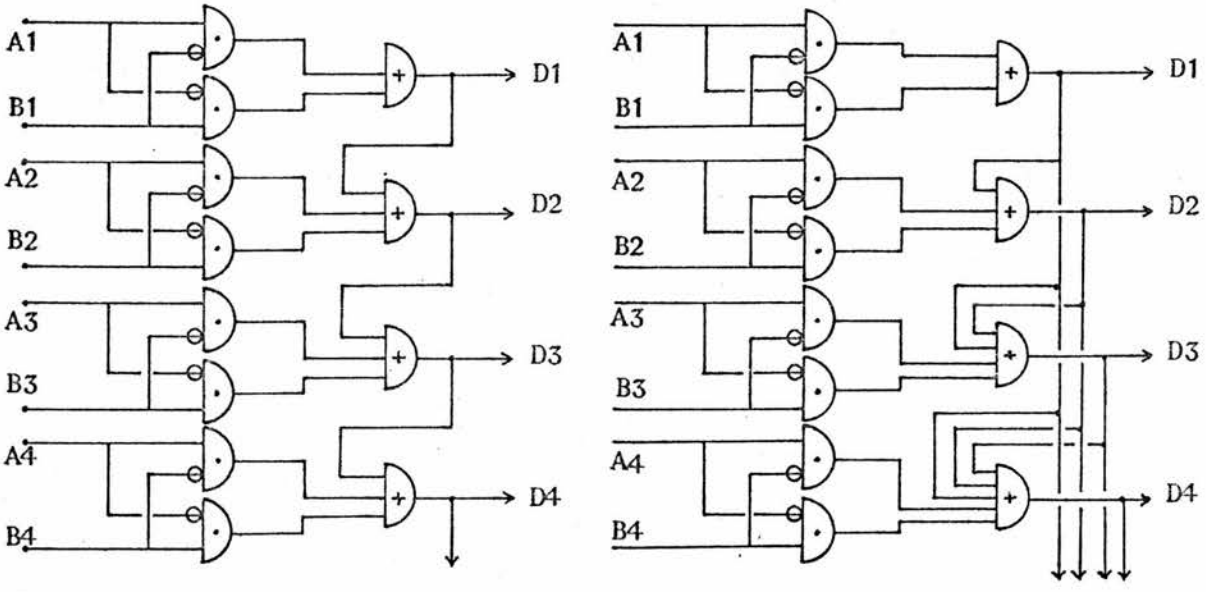
The kind of circuit layout to carry out such a process is shown in Figure 5.



Pipelining the Second Comparison Stage

Figure 5

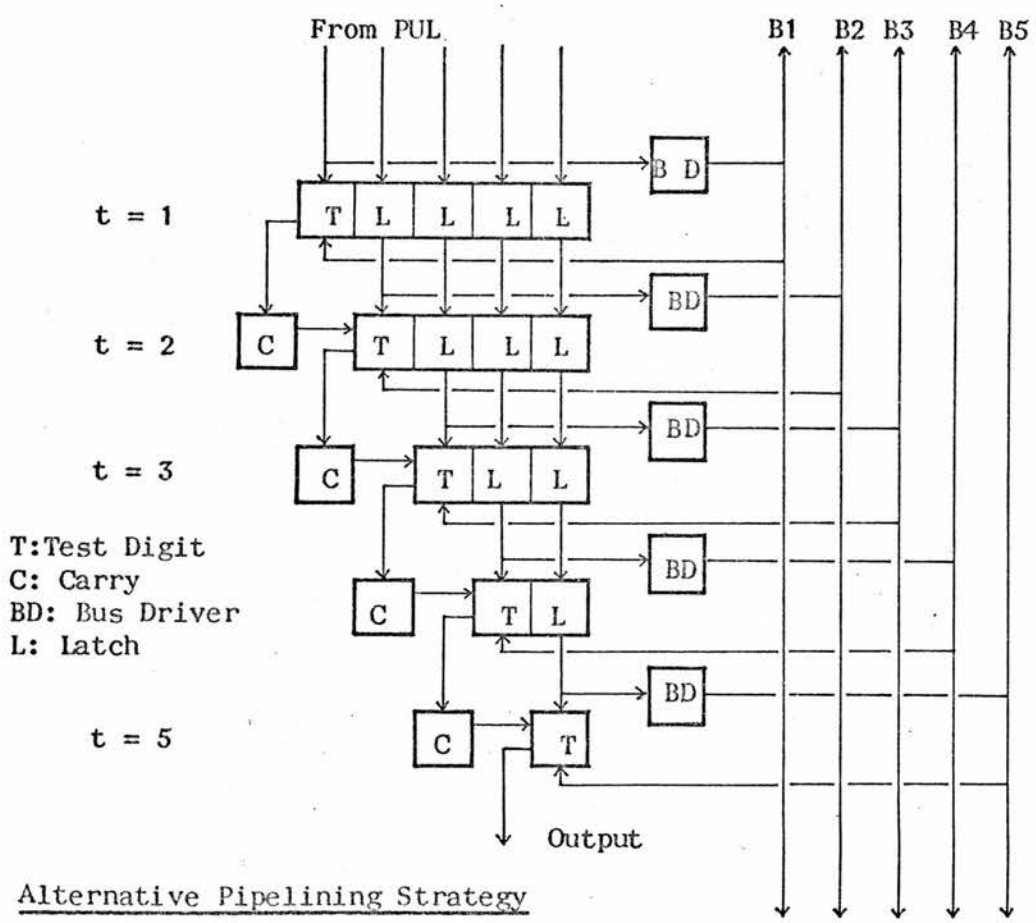
The circuitry within the box Arbitr. is relatively simple and can take one of the two forms shown in Figure 6.



Creating Intermediate Bus Values

Figure 6.

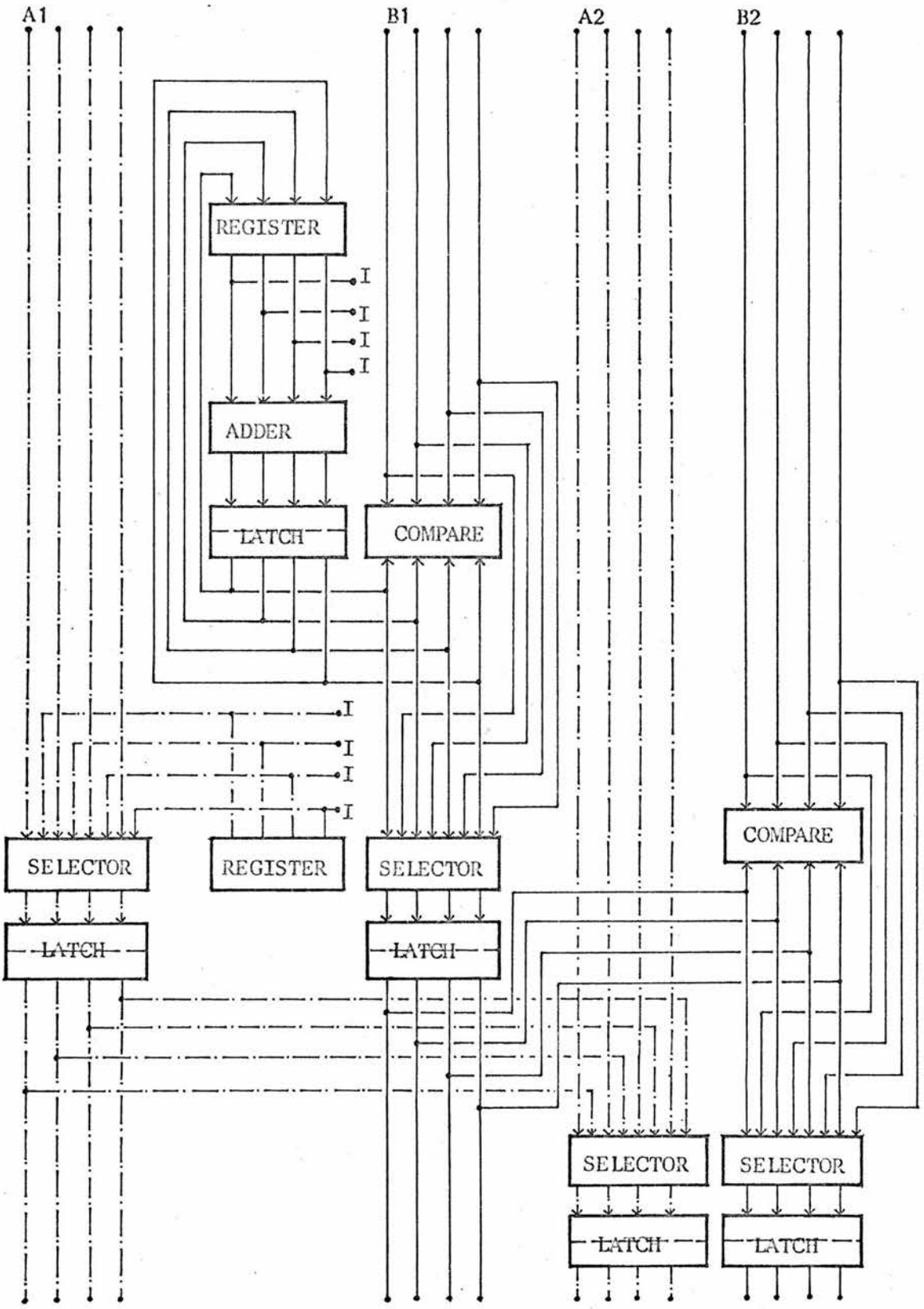
The problem with this approach was that to cope with all the possible combinations of eight bit values required eight levels of comparison, each with two eight bit buses. This did not appear to be a reasonable line to pursue. However, it suggested a way in which the alternative strategy given in Figure 25 of Chapter 13 might be employed. Originally this approach had been discarded because it was too slow to create the required output in the 100 nano seconds necessary to create a 500 point TV raster line in 52 microseconds. By pipelining the values being passed to the bus as shown in Figure 7 it would be possible to improve the time without the massive duplication of components used in the previous case.



Alternative Pipelining Strategy

Figure 7

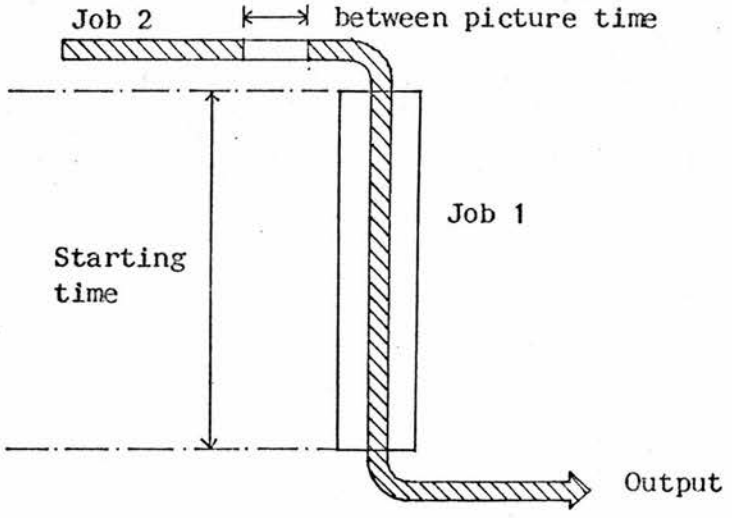
DEVELOPMENT OF THE DIGITAL PROCESSOR



Plane Unit for a Totally Pipelined Comparison Process

Figure 8

While investigating the various ways in which pipelining could be used to improve the speed of this second comparison it was realised that if restrictions were placed on the order in which data values were entered into the device, then the original sequential sort proposed in Chapter 13 could be employed. Consider the circuit layout shown in Figure 8. It can be seen that the values corresponding to the visible plane will be passed down from one plane unit to the next. Once the initial overhead-time, to pass values down this pipeline, has been taken then a continuous stream of output will be produced at one clock pulse intervals.



Time Relationships for the Fully Pipelined System

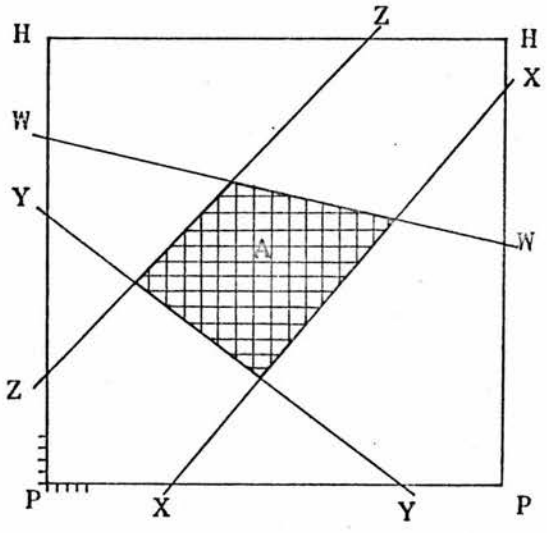
Figure 9

Figure 9 shows the time relationships which would be important in the use of the fully pipelined system. If there are 10,000 units in a sequence, for the sake of argument, then the switching on time, in other words the time between entering the first value and having it passed to the display unit, will be $10,000 \times 100/10^{-9}$ secs. This must be distinguished from the time between displaying two different pictures. This

interval depends on the space between the data being entered for the first picture and the data being entered for the second picture.

This work took the initial approach as far as it could be taken without actually building and comparing the relative performance of different devices in a practical environment. The next general advance resulted from a suggestion made by G. Rankine in Heriot-Watt University. In an attempt to reduce the original concept of the plane processing unit to a device which it might be feasible to build, the idea was to start with the point in volume test as a means of producing a display.

Consider the raster line section through the volume shown in Figure 10 The basic process can be described in the following way.



A: W.X.Y.Z

Section through Raster Line

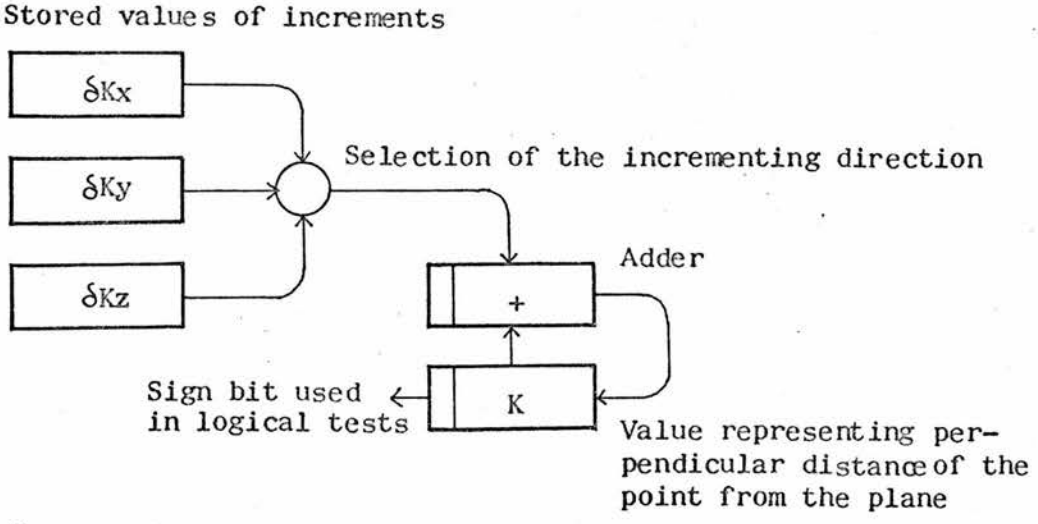
Figure 10.

If any point is taken on the picture plane PP then it will lie inside or outside the various planes making up the volume A. If the whole display scene: PPHH is treated as a three-dimensional grid of points,

and the points in lines perpendicular to the picture plane are processed sequentially from the picture plane backwards, then the first point to be found inside a volume will indicate a visible surface. The inside outside relationship will depend on the relationship between the planes defined in the boolean expression of the volume.

Because of the properties of planes and the fact that a regular grid is being used, this can be achieved in a similar way to the proposal in Chapter 14, Figure 6 except that in this case there will be three incrementing values required for K: Kx, Ky, Kz. This will demand a basic module of the form shown in Figure 11.

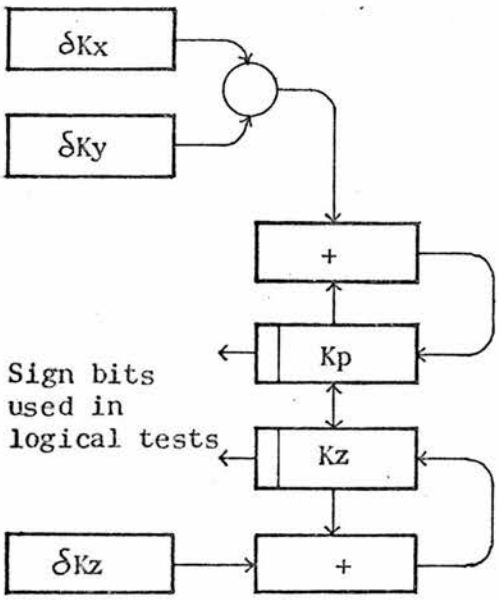
A preliminary estimate of the time needed to process a new point using this arrangement is 100 nanoseconds. Given the preliminary goal to be able to process the whole display scene in 1/10th second, results in a grid resolution of 100x100x100 points. This is obviously way below the resolution being aimed at. However, there are some immediate ways in which the number of tests can be reduced in producing a display.

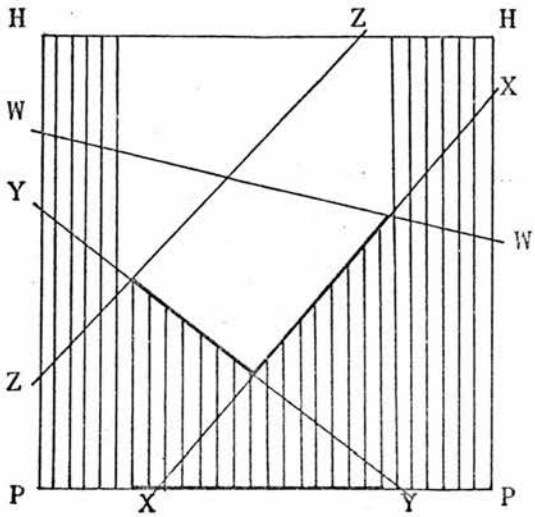


Basic Processor

Figure 11.

To start with it is not necessary to continue with Z direction tests for a particular display point once a visible surface has been located. However, to be able to stop at a surface and return to the picture plane demands an extra storage element in the basic module. The new pattern of scan lines for the previous scene is shown by the shaded area of the section drawing in Figure 12.





Kp: Value of K at the current display point on the picture plane.

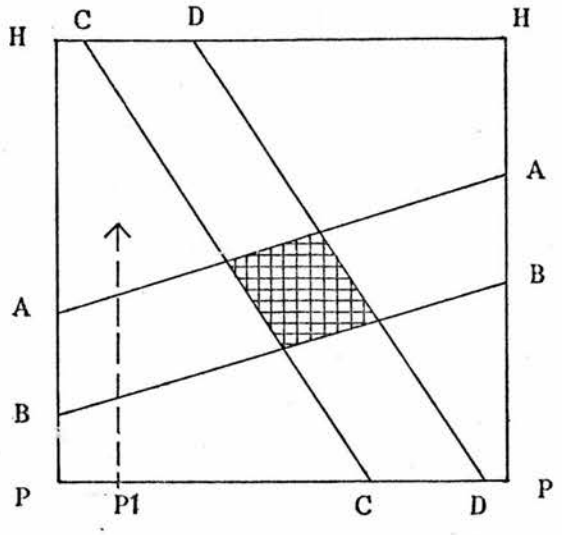
Kz: Value of K at the current point being used in Point in Volume tests.

Modified Processor

Figure 12

A further reducation in the number of tests could result from the following observation:

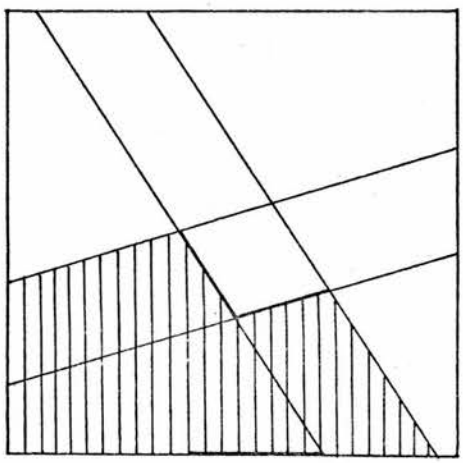
If the point P1 is taken on the picture plane grid, as shown in Figure 13, it will lie inside A and D, but lie outside C and B.



Section through Raster Line

Figure 13

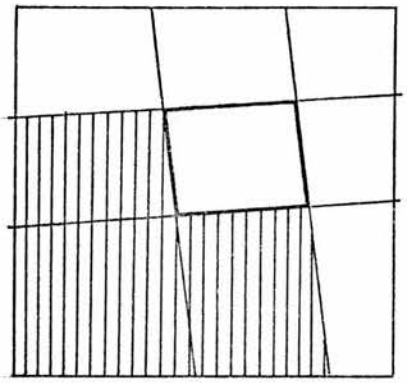
This means that Ka and Kd are negative and Kc and Kb are positive. In processing in the z direction from P1 each of these values will be moving towards $\emptyset$, which indicates that it is possible that a visible surface will be found. However, once the processing passes through plane A, Ka will be positive and δKa_z being positive Ka can only get larger. It can be seen that once this has happened there will be no possibility that a visible plane will be found. Consequently, further tests in the z direction can be avoided. Applying this principle gives a scan line pattern for the previous volume of the form shown in Figure 14.



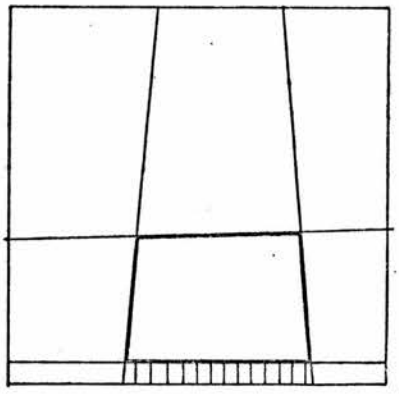
Scan line pattern

Figure 14

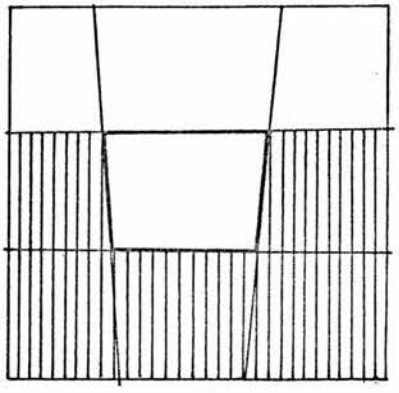
The position of the planes will have a considerable effect on the actual amount of testing necessary;



but this could reduce to as little as:



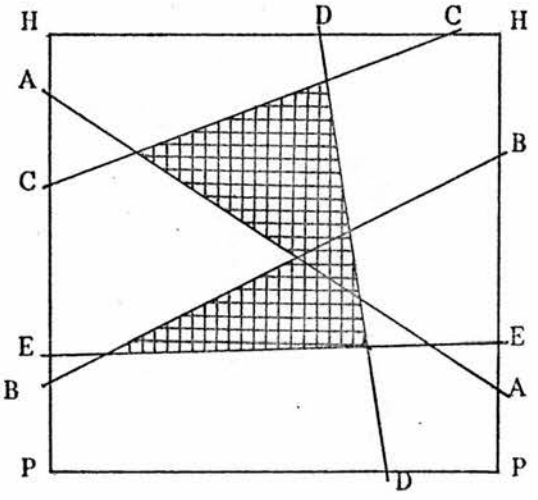
or be as much as:



The Relationship between scan lines and an Object's Geometry

Figure 15

This procedure depends on the values of the signs of K and δKz. Where these are both positive then further increments in the z direction for this plane can be stopped. If this state is indicated by a flag for each plane unit then for a volume defined by a dot phrase all the other plane units for this volume can be switched off as well. Thus the logical or of these flags can control all these plane units. If the expression is of the form (A+B)$_{\circ}$C$_{\circ}$D.E in other words a scene of the form shown in Figure 16.



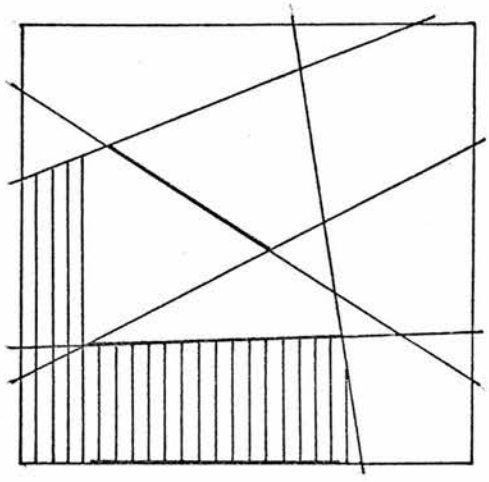
Concave Object

Figure 16.

then these flags can control the operation by anding the flags for A and B and then oring the result with the flags for C, D, and E. This results in a scan line distribution of the form shown in Figure 17

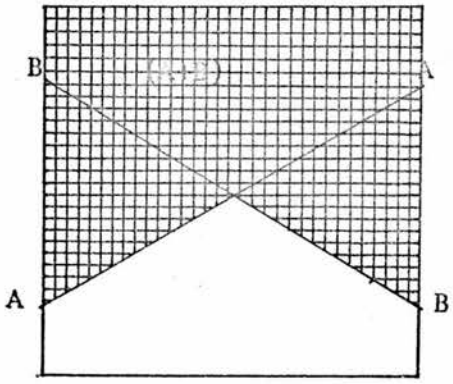
A further possibility can be demonstrated in the case of scene shown in Figure 18.

It can be seen in the case of this simple scene that once the visible plane A has been found that it should be possible to follow the surface



Scan Line Distribution for Concave Object

Figure 17

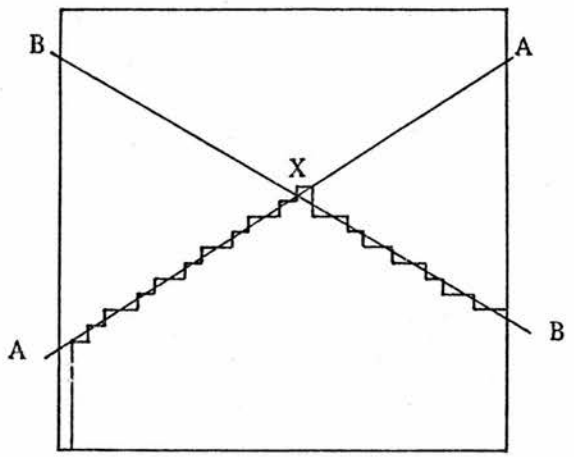


Looking into a Concave Section of an Object

Figure 18

of A until the surface of B is located. If, as soon as a test point is found to lie within the volume (A+B) the next increment is taken in the x direction, then as soon as the point tested is found to lie outside the volume the increments are again made in the z direction it should be possible to chase the surface in the way shown in Figure 19.

At point X the sequence passes outside A but enters B, which means that



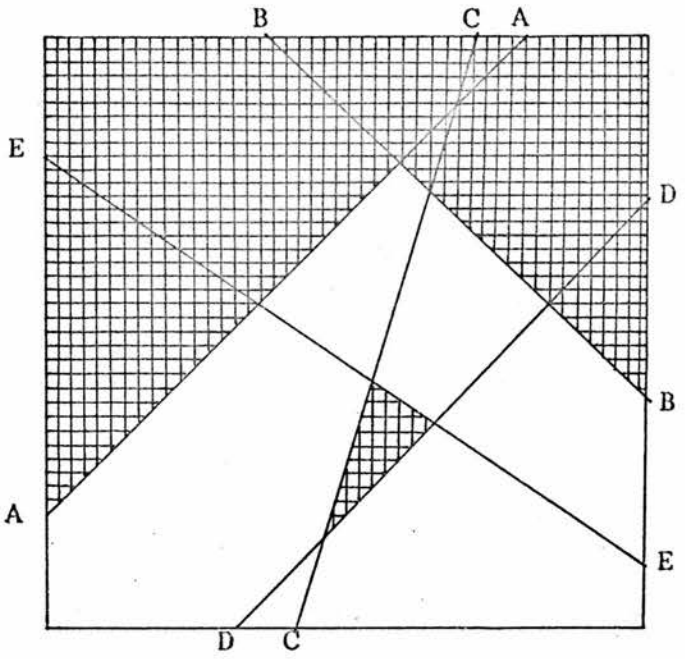
Stepping along a Surface

Figure 19

the stepping sequence must now be controlled by B. The obvious difference is that the z direction increments have been reversed. This choice must depend on the sign of δKb_z. Plainly the number of tests in this procedure are very much fewer than by scanning in the previous way. The question is whether this process can be used where more complex scene descriptions are considered.

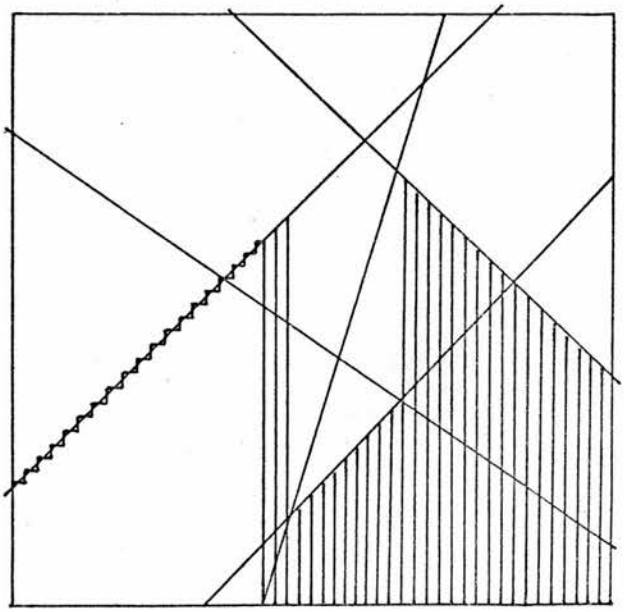
If the diagram of the scene A+B+C.D.E is constructed, then chasing along the surface of A or B could well pass behind a volume defined by C.D.E. However, it can also be seen that where the points on the picture plane lie outside C.D.E in the manner described above then there is no reason why surface following should not be adopted as shown in Figure 20.

Each of the scan line patterns considered so far has been controlled by the values of K on the picture plane. If the coefficients for each



A + B + C.D.E

The sequence of tests could take the form:



Mixture of Surface Following and Scanning

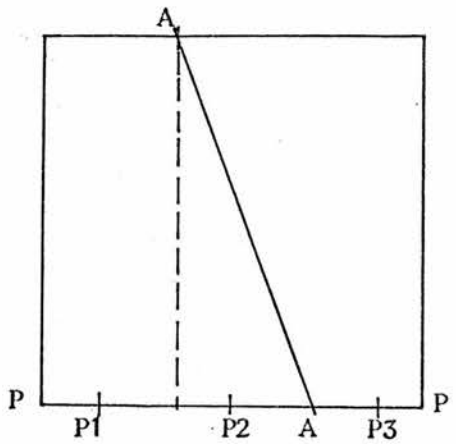
Figure 20

plane are used in the expression

$$(a.x + b.y + c.z)/d + 1 = K$$

than the value of this plane when subtracted by 1 will give the value

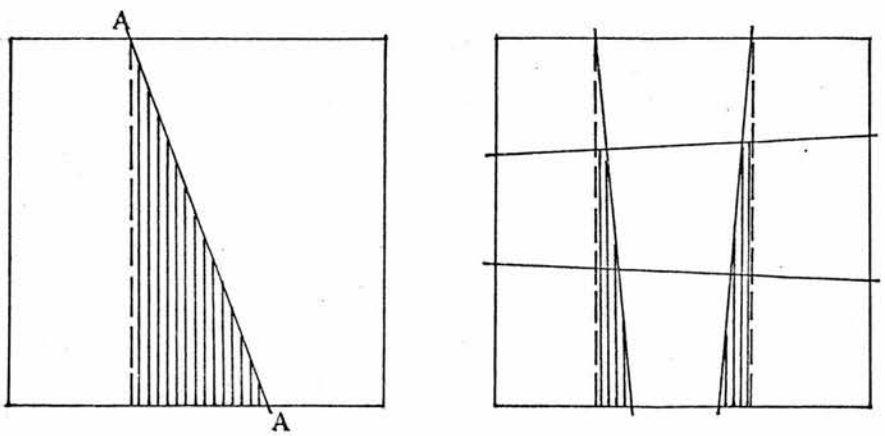
of the plane which passes through the viewing point parallel to the original plane. This plane will define the position of the horizon line on the picture plane. The significance of this point can be seen if the single plane A is considered in Figure 21.



Area of the Display Plane Affected by a Plane

Figure 21.

The value of Ka at P1 will be greater than 1, and at P3 less than 0. In the zone with K values outside the range 0 to 1, the effect of a particular plane on the display can be ignored. The scan line patterns for the plane and a previous object are shown in Figure 22.



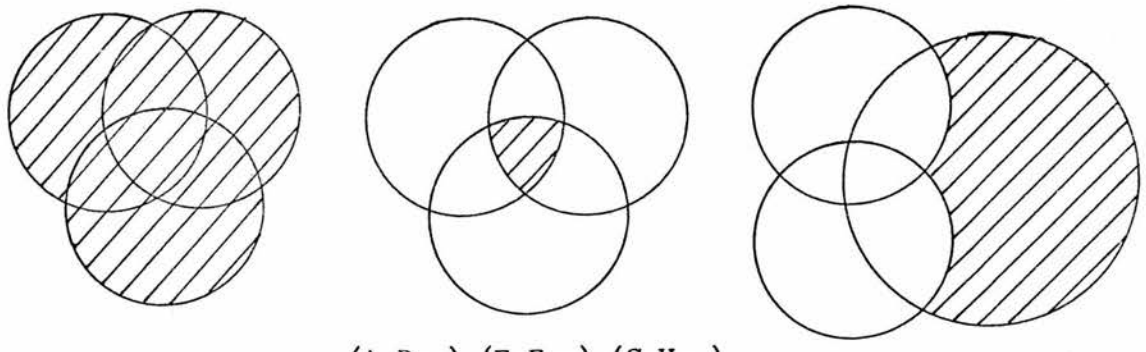
Scan Line Patterns for a Plane and Simple Object

Figure 22

This line of investigation was not taken further than this general survey. It did not appear from the initial analysis to provide the speed necessary for real time movement of displayed objects. However, it led to a possible way in which curved surfaces could be included as a display option. The surface following mechanism outlined above which consisted of incrementing the position so that the value of K alternated between small positive and small negative values, could be linked to a counter which only followed the Z direction increments so keeping track of the distance from the picture plane. This distance could then be used to drive the two comparison circuits already discussed. The same process can be applied to the appropriate description of a curved surface.

The use of curved surfaces in this way poses a major problem which will have to be resolved before this approach will apply to all situations. Assume to start with that only convex curved surfaces are used, then each curved surface can be thought of as a boolean product. Where these are added together there is no difficulty since the form of the expression matches the structure which has already been analysed. Similarly, the intersection of curved objects creates no difficulties. It is when the curved void, or volume subtraction is considered that difficulties emerge.

The subtraction of a curved surface from a convex object requires a three level boolean expression. It has already been realised that three level expressions would be useful in processing volumes with plane facets. Where planes are being processed in parallel there is a fixed upper limit to the size of the boolean expression which can be



(A.B..)+(E.F..)+(G.H..)

(A.B..).(E.F..).(G.H..)

(A.B..).($\overline{E.F..}$).($\overline{G.H..}$)

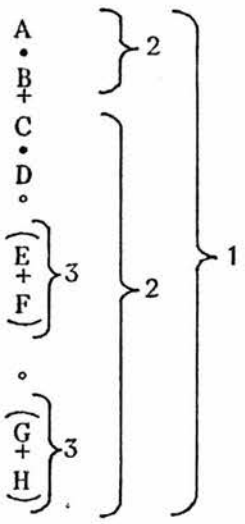
The Use of Curved Surfaces

Figure 23

used. Where an object with concave sections to its surface is being displayed, the two level boolean expression will tend to be very large. If three levels are permitted such surfaces can be expressed in a much more compact form. Consequently, for the parallel processing of planes it is necessary to consider the requirements of processing this extra level.

The same units described above can be used to process a three level expression, merely by permitting a limited form of sequential processing for limited areas of the display. Consider the expression shown in Figure 24.

Where C.D is not producing a visible surface the sub-expressions can be ignored. As soon as C.D produce a visible surface at the dot level if this is compared with all the back faces in the sub-expressions before it is passed to the plus bus, and then each of the visible front faces in the sub-expressions are similarly treated in sequence the correct



Processing a Three-Level Boolean Expression

Figure 24

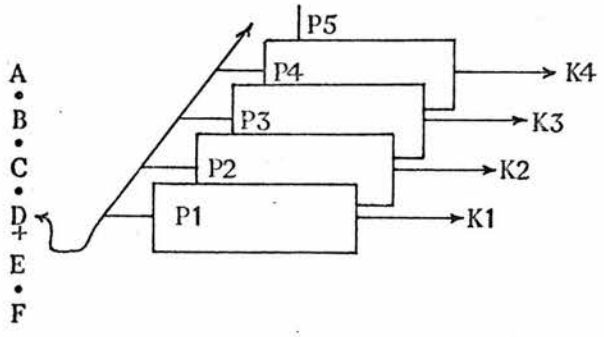
visible plane will ultimately be found. This will mean an extra test per sub-expression per point.

This sequential procedure even if it is only used for limited areas of a display, will prevent the processing of a new raster point each clock pulse. In the case of a TV raster line, 52 microseconds are taken for the actual display, while 10 microseconds are allowed for the fly-back time. This gives a very small leeway in which extra processing can be fitted. Working on 100 nanoseconds as the clock pulse period, this means that less than 100 extra points can be processed per line, and a buffer to store the values in one or more lines is necessary.

The expansion of plane based expressions can lead to a very great increase in the length of the overall expression. The use of curved surfaces does offer one improvement in this situation. A curved surface can be represented by two equivalent plane surfaces at any raster point - a front and back surface - so all third level curved

surfaces can be represented by sub-expressions of the form: (F+B). The expansion of a product expression containing several sub-expressions of this form can be achieved by the appropriate duplication of the surfaces in the parallel processor. The advantage is that this duplication will result in a smaller overall expression than if plane surfaces had been used to approximate the original surface.

Because of the limitations associated with having an upper limit on the size of the boolean expressions which can be processed in this way, an alternative use of parallel processing was considered. If the boolean expression is processed sequentially, but the raster points in the display are processed in parallel it appeared that it should be possible to create a fast display operation which had no upper limit on the size of scene description it could accept. Given an element which can process the boolean expression sequentially, then by arranging these in parallel for successive points in a raster an arrangement of the form shown in Figure 25 becomes possible.

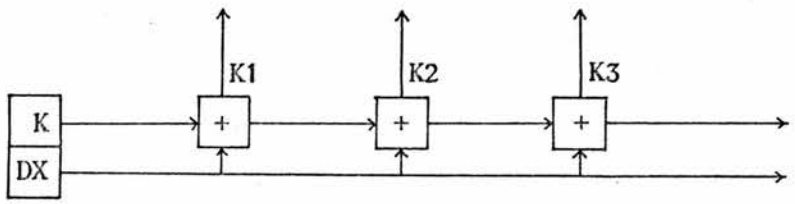


Parallel Processing of Raster Points

Figure 25

If the raster points are equally spaced then it should be possible to use the result established in one section of the display for other

sections of the display. The original idea for creating the appropriate value of K for a plane at all points represented in the parallel processor is shown in Figure 26.

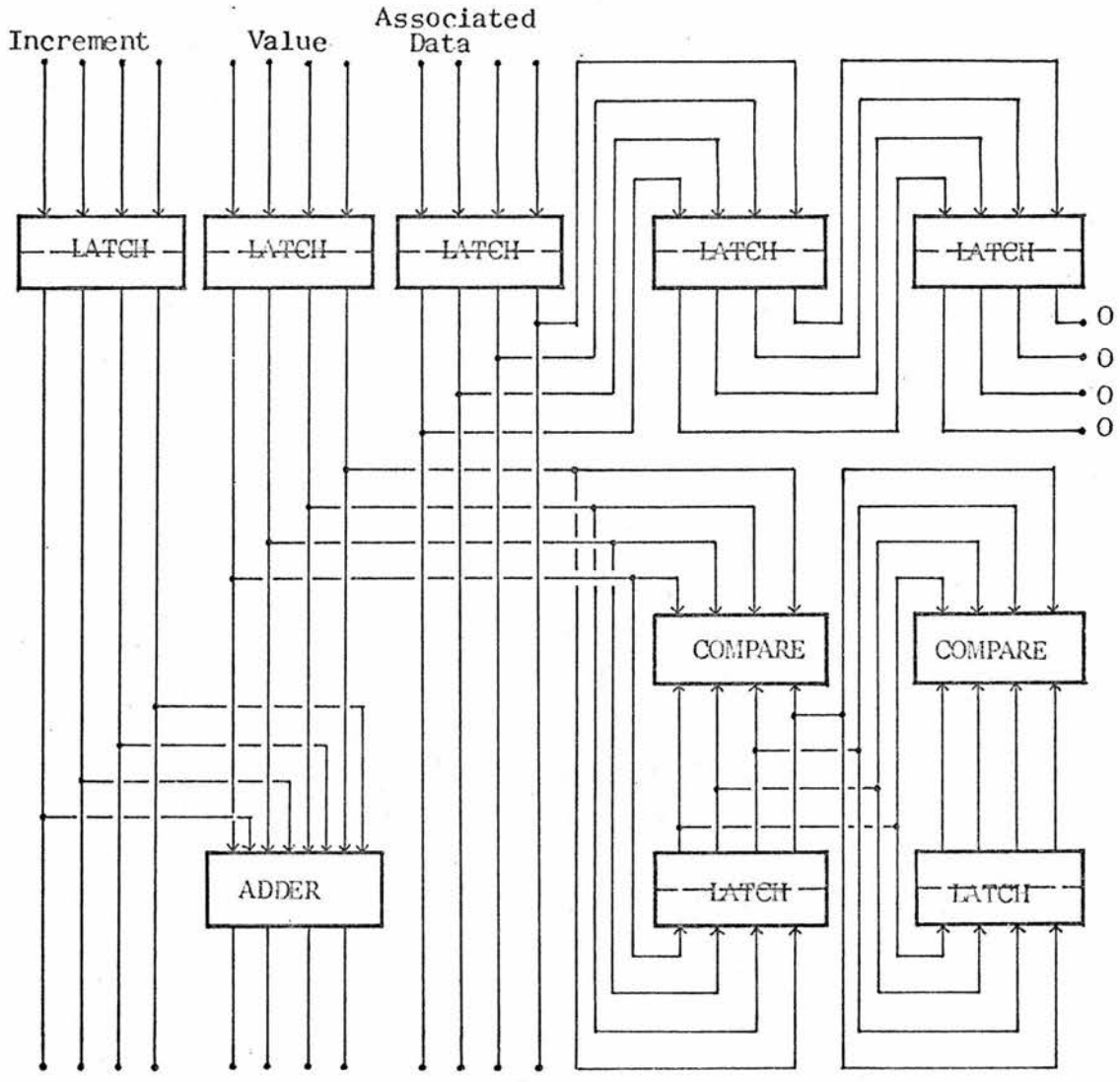


Generating the Value K for a Plane at all Raster Points

Figure 26.

The problem with this simple layout was the time it would take to sequentially add DX to successive values of K. The original idea was to generate all the values to be displayed at the same time. It is clearly not necessary to do this since it takes time to display each point - approximately 100 nanosecs for a TV raster line. It was consequently possible to pipeline this arrangement to give a raster point processing unit shown in Figure 27 . If a different form of display device is considered where the display point is an integral part of the processing unit creating the value to be displayed, then the need to create all display values simultaneously is more important. It was while following this approach that a different solution to the basic process shown in Figure 27 was found.

The basic problem was to enter a value K for one point in the display, and then to calculate all the values of K for other points represented in the parallel processor so that they are presented simultaneously to the picture plane. This did not appear possible using the scheme

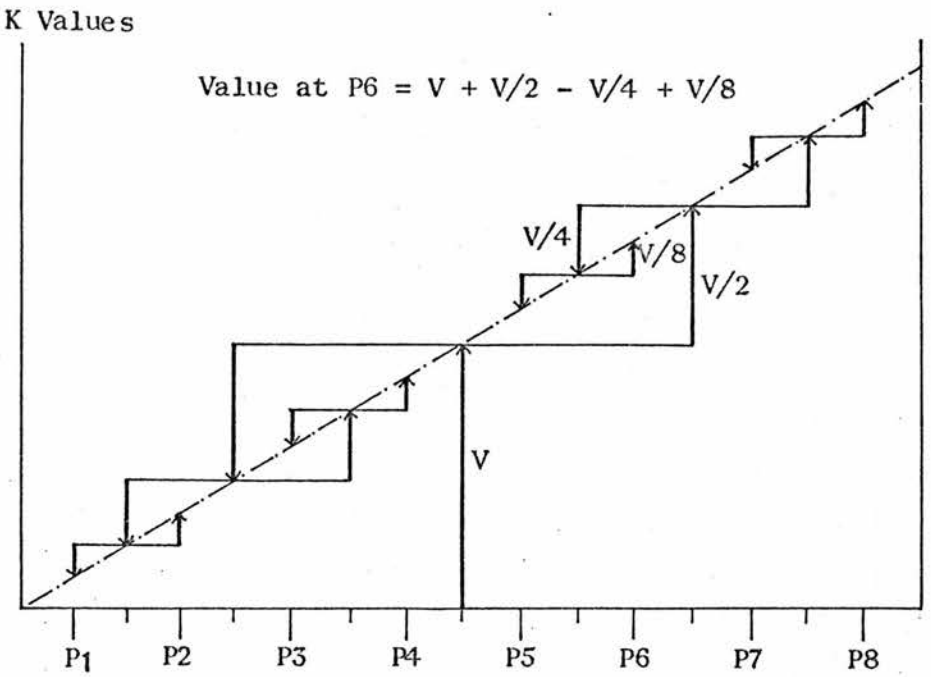


Raster Point Unit in a Pipelined Display Processor

Figure 27

in Figure 26 without waiting for n.t seconds - where n is the number of additions and t is the time for each addition. The goal was to process as many plane values in each processing unit as possible in a/50th of a second which is the refresh time for a TV monitor picture.

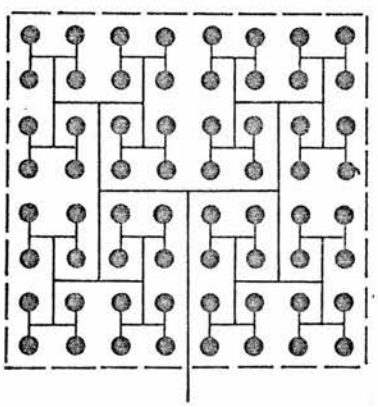
One way of creating the required values at each raster point in a line is shown in Figure 28. Using a binary tree it is possible to input a single value at the centre V, then at the first level by adding and subtracting V/2 two intermediate values are created which



Binary Tree Used to Create Values on a Ramp at Raster Points

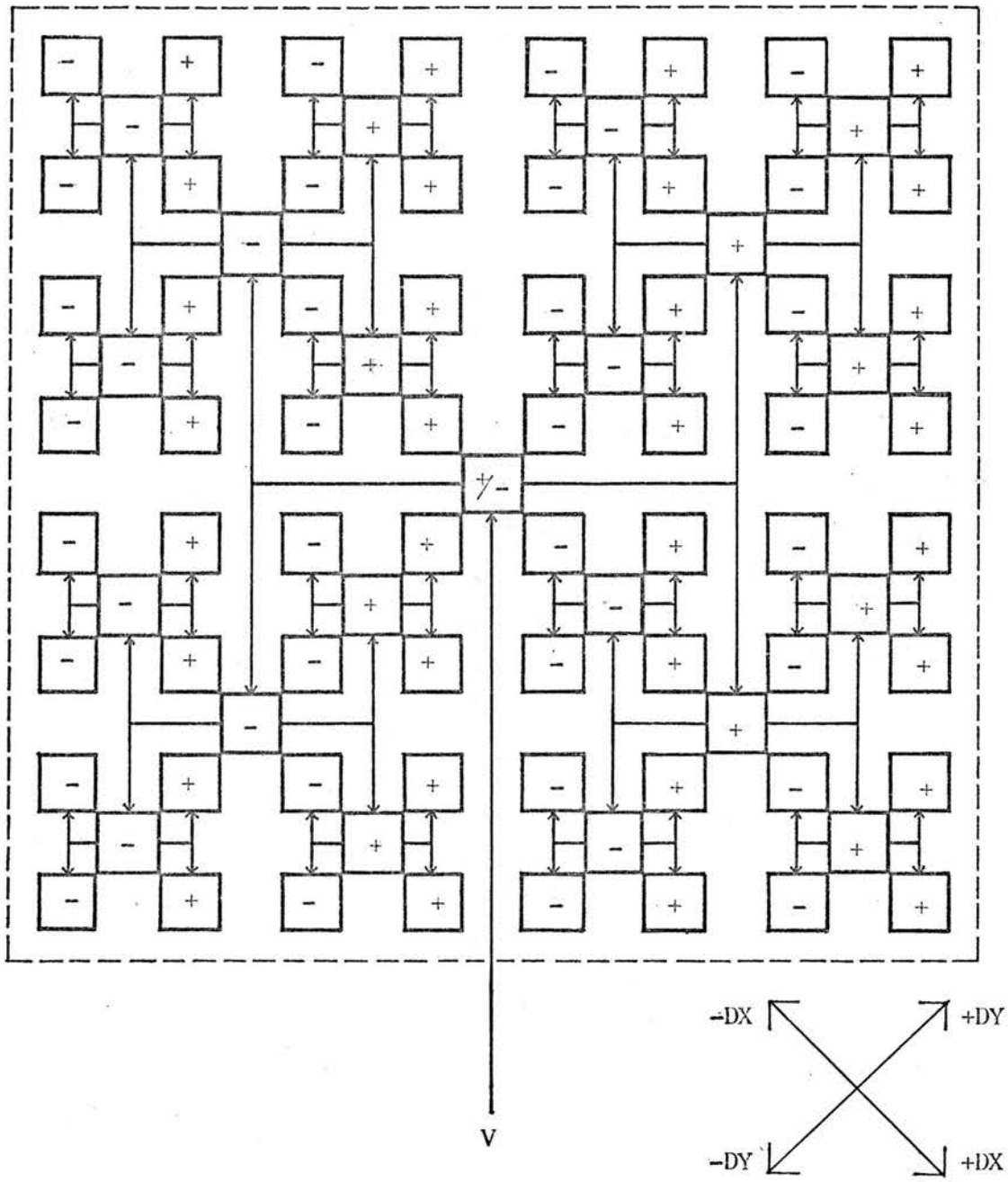
Figure 28

in turn can be used to create 4 more intermediate values, so increasing the number of points until the required resolution is reached. The advantage of this scheme is that the division by two is a simple shift operation which means linking the bit lines with an offset either one unit to the left or one unit to the right. If this idea is carried to its logical conclusion it results in the two dimensional tree shown in Figure 29.



Two Dimensional Tree

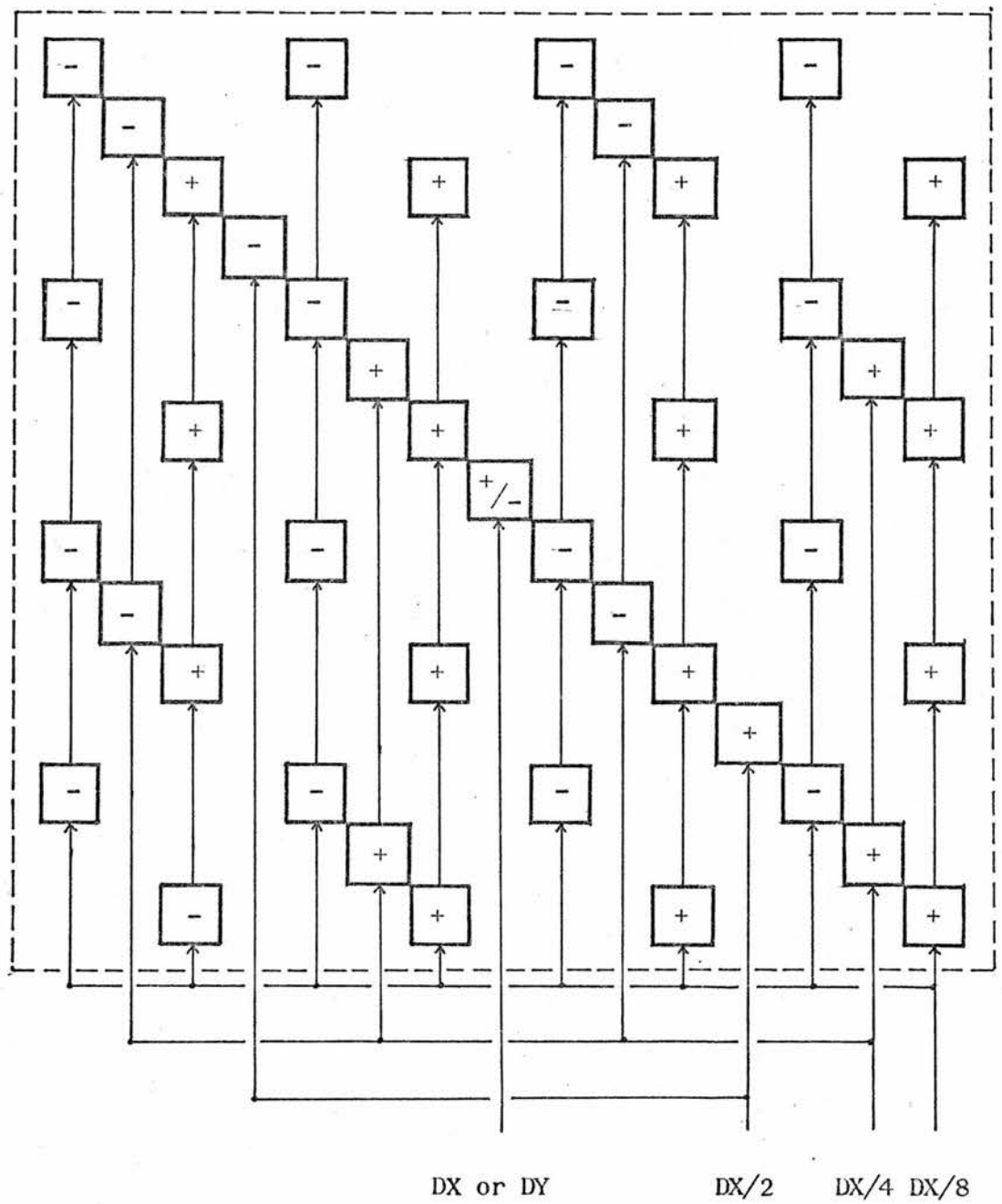
Figure 29



Addition Tree for the Display Panel

Figure 30

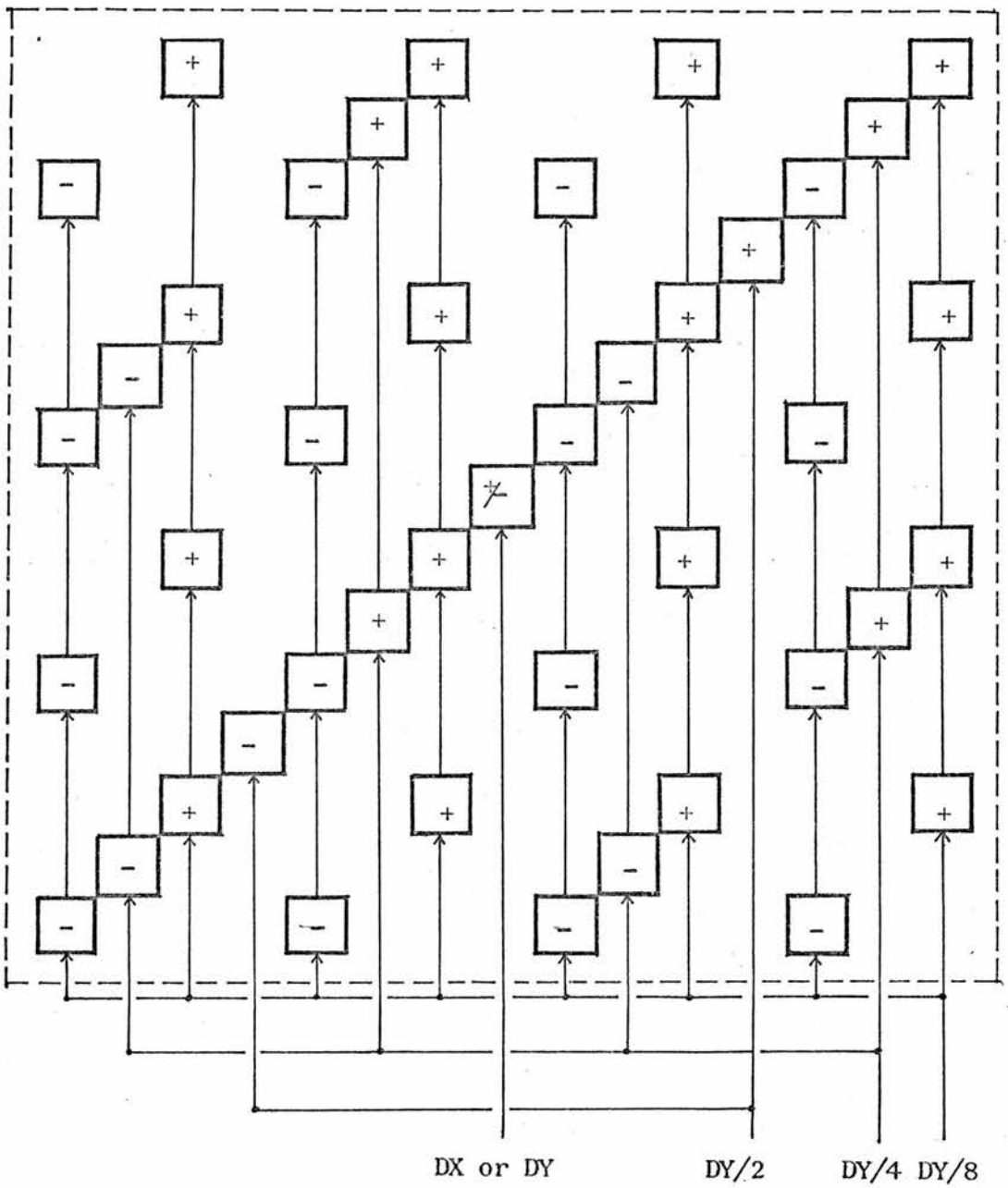
To create the correct X and Y increments the additions and subtractions have to be arranged in the way shown in Figure 30. It can be seen that



X Increments for the Display Panel

Figure 31

the additions and subtractions will work correctly for a positive increment but will have to be reversed for a negative increment. The axes for the increments will have to be at 45^{o} to the axes of the



Y Increments for the Display Panel

Figure 32

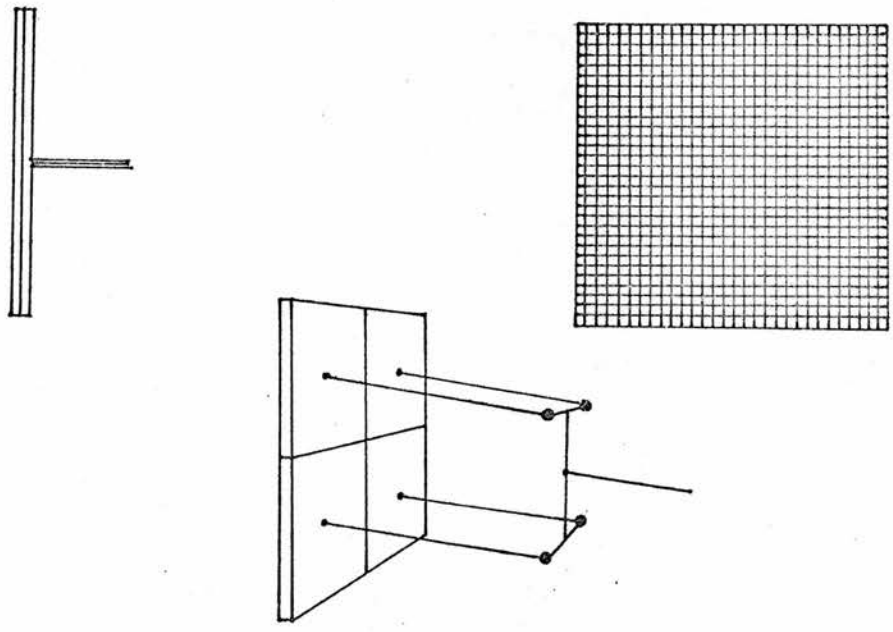
picture frame. The way that the incrementing values are distributed to the various adding units is shown in Figures 31 and 32.

There are practical difficulties with this approach but before consider-

ing them it is worth while taking these ideas to their logical conclusion. All that the tree described above does is to distribute the values for each plane in parallel to all the display points on the surface of the picture plane. At each point of the display plane there will have to be a processing unit very much like the right hand part of Figure 27. The incoming values for each plane can be pipelined through this addition tree if the incrementing values are also passed along a string of latches each time they are shifted to the right. If the time for one addition is taken as 100 nanoseconds, then the number of planes which can be processed every fiftieth of a second is given by

$$\frac{1,000,000,000}{100.50} = 200,000 \text{ planes.}$$

The natural application of this approach is to create a display panel where the display surface is integral with the display-logic circuits.



Display Panel and the Extension to the Addition Tree

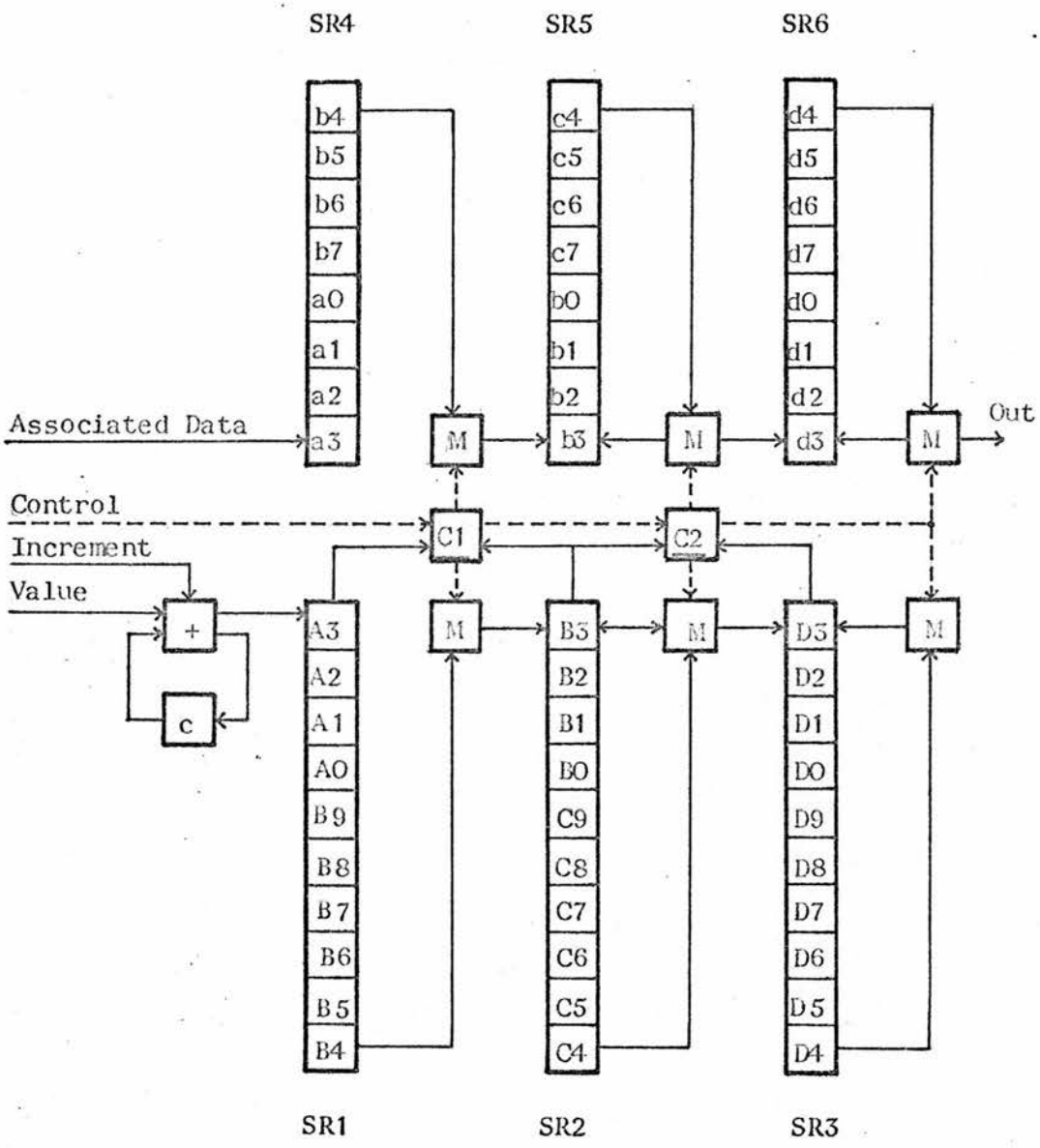
Figure 33

The use of light emitting diodes or liquid crystal might possibly be used in such a device. The physical arrangement of such a panel is shown in Figure 33.

As an abstract machine this device is very attractive. The difficulties with this initial approach, apart from a very high cost, are the physical number of lines which are needed to input data to a panel and the problem of integrating a display surface with the electronics. If a single panel of the appropriate size could be constructed, the number of input lines would not be a serious handicap since the internal linkages could be achieved using printed circuit techniques. The present limit on the size of silicon chips is such that the individual panels would be very small and the problem of linking over 100 lines from each individual panel into a higher level in the external adding/incrementing tree becomes prohibitive.

However, the absolute speed of the device makes an alternative implementation possible which does not lose the attractive modular features of the original idea. To create a resolution of 1000x1000 points has already been shown to need value and increment sizes of 30 bits for the simple algorithm proposed. If these numbers instead of being processed in parallel are processed in sequence, one bit at a time, then it is still possible to create a very complex display. Assume that it takes 20 nanoseconds to process a one bit addition, then the number of planes which can be input, given a need for a 1000x1000 resolution, in 1/50th of a second will be:

$$\frac{1,000,000,000}{30 \cdot 20 \cdot 50} = 30,000 \text{ planes.}$$



Plane Unit for the Sequential Processing of Numbers

Figure 34

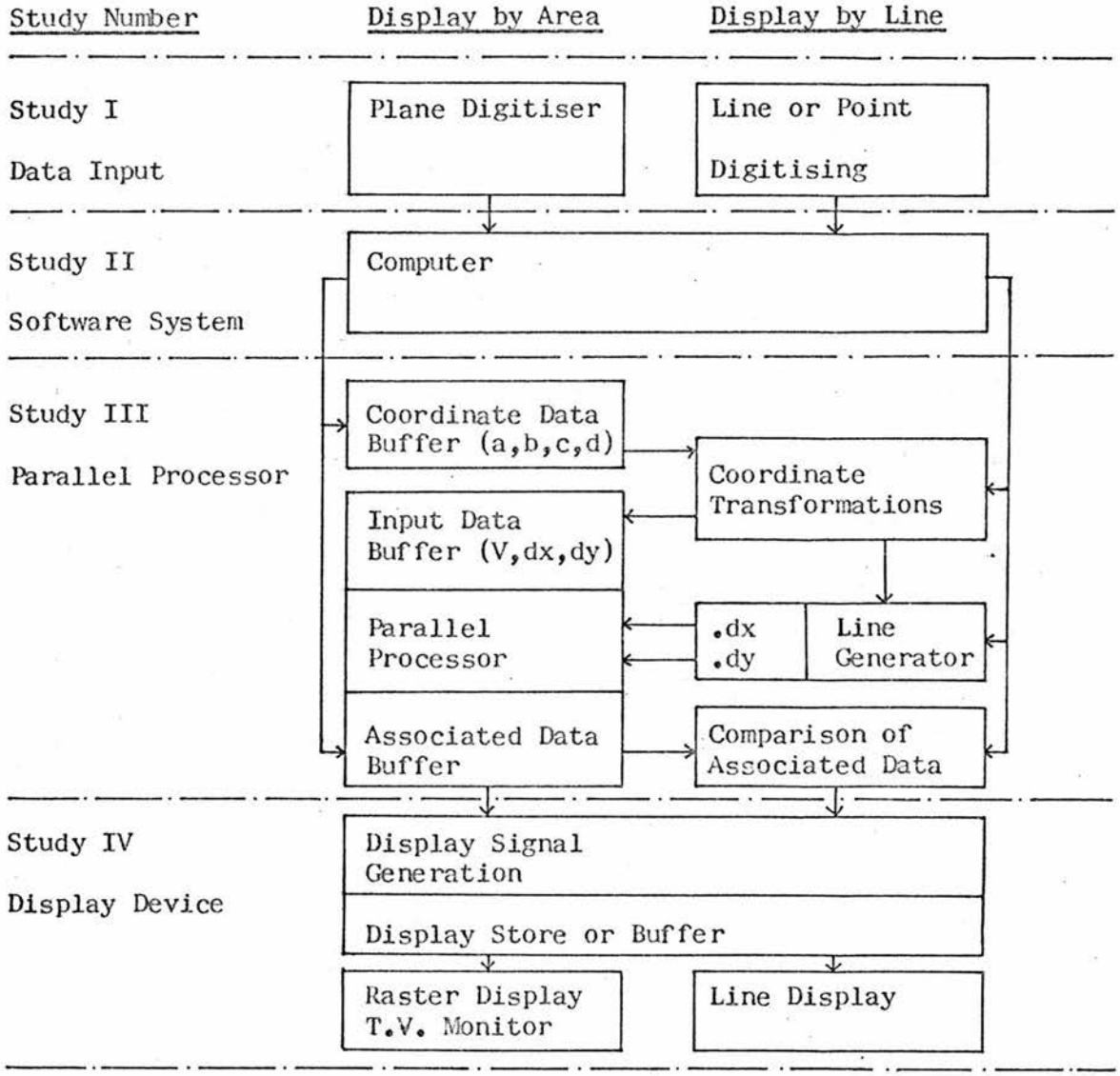
This approach makes it possible to construct a relatively simple point processing unit made up from six shift registers, six one bit two way selectors and two comparison control units. In Figure 34 registers SR1, SR2 and SR3 carry the depth information, while SR4, SR5 and SR6 carry the associated data which will be used to create the display.

DEVELOPMENT OF THE DIGITAL DISPLAY PROCESSOR

The output from SR6 at the end of processing the full boolean expression will be used to create the appropriate intensity at a display point. It appears that at present the use of light emitting diodes cannot be used to create a variable intensity point value. However, if the plane units include two more comparison cells, then the two neighbouring display values can be compared in the manner discussed in Chapter 13, Figure 28, and an array of LEDs, one per point unit could be used to create line displays. Though the same tree structure can be used to generate the values at each point unit the previous shift and add implementation of the increments at each level in the addition tree has to be changed when sequentially processing values, but this does not appear to create any major difficulties.

Further development of the proposals made in this Section are being carried out at Heriot-Watt University in the Department of Electrical and Electronic Engineering, under the project name of ASPECT 1., (A Sequential, Parallel Electronic Colour Terminal). As a conclusion to this section it seems appropriate to include the system diagram which is being used to structure this work, which is given in Figure 35. It can be seen from this diagram that there are four separate areas of study outlined in this proposal. The area of primary interest will be study III. However, to make the results of this study operational will require further thought to be given to the problems of entering geometrical data into the system. The early stages of the software system discussed in the previous section will have to be developed to provide the appropriate input data for the hardware

processor, and alternative display possibilities will have to be considered.



System Diagram for ASPECT 1

Figure 35

CONCLUSIONS

CONCLUSIONS

In this final section the task must be to bring together as many of the different issues that have been raised in the main body of the thesis as is possible. Perhaps the best way of doing this is to consider the applications which can be made of this work. Some of these were indicated in the introduction, and were taken as a starting point to the investigation; others have arisen from the research itself.

The applications divide themselves into two areas. The first are linked to the original objectives and are found in the construction and use of data bases for planning and design work where spatial information needs to be stored. The second area of application arises from the hardware proposals made in the previous Section and depend on the speed at which certain display operations can be conducted. In this Chapter a brief outline of these possibilities is given, followed by a more speculative assessment of further developments.

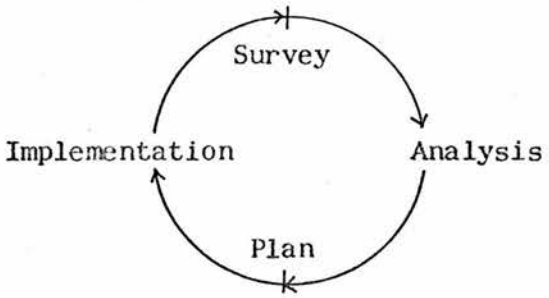
Data Base Development for Planning and Local Government

Before considering the particular application in this area which the work in this thesis might make possible, it is necessary to provide a short review of the development of the Planning process in Local Government. The Planning process can be caricatured by the sequence of operations:

Survey ⟶ Analysis ⟶ Plan Development ⟶ Implementation

In the early stages, the plan, or the Master Plan was regarded very much as a static goal, even though provisions were made to review its contents periodically, and it was primarily concerned with the development of the Physical Environment. As the limitations of this approach

became clear, a more flexible policy was adopted and the paradigm of the planning process became a cyclic procedure.



The Planning Process

Figure 1

The Plan itself was couched in more general terms and extended to consider social and economic factors. Proposals were kept amorphous enough to maintain a range of possibilities open for the future so retaining necessary room to manoeuvre. Future physical developments were only loosely defined in a 'Structure Plan'. As these ideas were developed, hopes were formed that they could be extended to substitute conscious - therefore intelligent - government control over many more of the self-regulating processes in the social system which, at times, led to unfair or socially unacceptable situations. However, this appeared to require a monitoring program capable of handling information disaggregated to the level of the individual citizen and to need a continuous flow of feedback data to make it operational.

This goal raises difficult political questions, but long before such objectives were in sight, practical constraints came into operation; principally in the form of a lack of data processing power. One of the primary ideas which lies behind the design of any control system seems to be that the energy required to control a system should only be a

FIGURE 2

Classification of Twenty Urban Planning Models

MODEL NAME	AUTHOR(S)	CITY	APPROX. DATE	SUBJECT: LAND USE	a. Residential	b. Industrial (Mfg.)	c. Commercial	d. Govt. or Institutions	e. Roads, Streets, Alleys	f. Public Open Space	POPULATION	TRANSPORTATION	a. Interzonal Trips	b. Other Transp.	ECONOMIC ACTIVITY	a. Employment	1. Retail Trade	2. Manufacturing	3. Service	b. Trade	1. Retail	2. Other	c. Personal Income	FUNCTION: PROJECTION	ALLOCATION	DERIVATION	THEORY: BEHAVIORAL	a. Economic (Market)	b. Preference	GROWTH FORCES	a. Gravity	b. Trend	c. Growth Index	d. Input-Output	METHOD: ECONOMETRIC	a. Regression	b. Input-Output	c. Markov Process	MATHEMATICAL PROGRAMMING	a. Linear Programming	b. Other Analytic Forms	SIMULATION	a. Autonomous	b. With Intervention
1. How Accessibility Shapes Land Use	Hansen	(Hypothetical)	1959	x						x													x						x										x				
2. Activities Allocation Model	Seidman	Philadelphia	1964	x	x	x		x		x					x		x						x		x	x		x			x									x			
3. Chicago Area Transportation Model	C.A.T.S. Group	Chicago	1960	x	x	x	x	x	x			x					x						x			x						x									x		
4. Connecticut Land Use Model	Voorhees	State of Conn.	1966	x						x						x	x	x					x		x					x												x	
5. Econometric Model of Metro. Employment and Population Growth	Niedercorn	(Hypothetical)	1963							x			x			x	x	x					x		x							x			x						x		
6. EMPIRIC Land Use Model	Brand, Barber, Jacobs	Boston	1966							x						x	x	x					x		x				x			x			x						x		
7. Land Use Plan Design Model	Schlager	S.E. Wisconsin	1965	x		x	x	x	x	x														x	x	x	x		x									x		x	x	
8. Model of Metropolis	Lowry	Pittsburgh	1964	x	x	x				x						x	x							x	x				x									x				
9. A Model for Predicting Traffic Patterns	Bevis	Chicago	1959								x	x											x		x				x													x	
10. Opportunity-Accessibility Model for Alloc. Reg. Growth	Lathrop	Buffalo	1965	x						x	x	x												x	x				x										x				
11. Penn-Jersey Regional Growth Model	Herbert	Philadelphia	1960	x																			x	x	x	x												x		x		
12. Pittsburgh Urban Renewal Simulation Model	Steger	Pittsburgh	1964	x	x	x				x						x	x				x	x		x	x			x	x			x	x			x					
13. POLIMETRIC Land Use Forecasting Model	Hill	Boston	1965							x						x	x	x				x	x	x	x				x			x								x	x	
14. Probabilistic Model for Residential Growth	Donnelly, Chapin, Weiss	Greensboro	1964	x																				x	x							x		x	x						x	
15. Projection of a Metropolis: New York City	Berman, Chinitz, Hoover	New York City	1960							x						x	x	x	x	x	x	x		x								x										
16. RAND Model	RAND Corp.	(Hypothetical)	1962	x						x					x									x	x	x	x	x				x								x		
17. Retail Market Potential Model	Lakshmanan, Hansen	Baltimore	1964								x									x				x	x				x											x			
18. San Francisco C.R.P. Model	A.D. Little Corp.	San Francisco	1965	x						x													x	x	x	x	x								x						x		
19. Simulation Model for Residential Dev.	Graybeal	(Hypothetical)	1966	x											x								x	x	x	x	x				x								x	x		
20. Urban Detroit Area Model	Doxiadis	Detroit Area	1967							x	x												x	x	x				x					x					x			

small proportion of the energy needed to run the system itself. Applying this principle meant that there were inadequate resources to either collect or process the data needed to run an effective monitoring system.

It was in this context that the use of automatic data processing appeared to offer a possible advance. The ability to use raw disaggregate data and model the processes which lack of time and manpower previously prohibited, seemed to provide at least the beginning of the answer to the planning dilemma.

The first use of automatic data processing in this area appeared in the form of automated filing systems, classified as Urban Data Banks. Following this, though to some extent in parallel, planners started using the available machine readable data to model urban systems in an attempt to assess the effect of changes made to these systems. In 'A Conceptual Framework for Urban Planning Models' (M. Kilbridge, R. O'Block, P. Teplitz, January 1968) the authors summarise the more important attempts to construct such models in the United States. A table, taken from this paper is given in Figure 2. The earliest use of these techniques in a practical situation, and on a large scale, was in the Chicago Area Transportation Model (1960), and from this start a variety of other studies followed.

By their nature, where models are used to predict future conditions, they are being used as theories. They are being used as pattern generating procedures which it is assumed can be extended beyond the immediate observational data on which they were based. Urban models

CONCLUSIONS

can be classified by the generating procedure adopted in this process. In the paper 'A Conceptual Framework for Urban Planning Models', the principal types are given as (1) Growth Force Models, (2) Econometric Models, (3) Mathematical Programming Models and (4) Other Analytic Forms. Simulation is used as a term to denote the way in which a model is used rather than a classification of a model type.

In use, these models came up against many practical problems, mostly in the area of data collection. Again, the energy required to provide the predictive data was too great to give the overall improvement which was presumably the objective. However, the experience gained provided an outline objective for future efforts.

One of the issues which the early work on models raised was the relationship between the model and the modelled. If people could, by choice or by command, conform to the behaviour required by a model then presumably its predictions would be achieved. At one level this is a policy which can be adopted, and is the approach which underlies the basic rules and agreed code of conduct in most organisations. At a different level where freedom of personal behaviour is required, then this prescriptive solution cannot apply. However, there are still patterns of behaviour which can be discerned in an aggregate view of human systems, and as has already been observed, these patterns can be used as theories in an alternative application of modelling. The distinction between the two approaches seems to rest ultimately on the size and reliability of the total system. In the first approach, if one of the components of the system fails to meet its target, then the activities of all the other components may be placed in jeopardy.

CONCLUSIONS

The solution to this problem which is to introduce parallelism and redundancy in a systems of independent but cooperating parts, or even competing components, is to create something which is much more difficult to model.

The pattern of behaviour of a large number of individuals can be such that at an aggregate level, global relationships can be discerned which do not change. The Gauss distribution of the velocities of gas molecules, and its relationship to global properties such as gas temperature and pressure was one of the earliest discoveries of this kind of relationship. Similar relationships have been found in mass human behaviour such as the gravity model used in shopping centre location studies. In finding such relationships in the Social Sciences the problem seems to be the opposite to that faced by the Physical Sciences. In the latter, the global behaviour tends to be the starting point, and the disaggregation of possible components the general problem. The Social Scientist in contrast faces his ultimate 'Atomic Particles' and his problem is to find useful patterns of global behaviour.

By their nature, it would seem that global properties can only be applied to give aggregate or average results. They can, however, be used in certain forms of spatial analysis, where it is possible to assume the subdivided zones in an area will behave in an average way. This disaggregate behaviour can then form the basis for spatial interaction models used to provide further global information which is dependent on spatial distributions such as population, income, and raw materials, such global information being useful in resource management and planning

at an aggregate level. The problem with this approach appears to be defining the 'average response' of people or any other unit being considered to the multitude of different conditions which might affect the whole situation. This becomes a complex problem in the area of statistics which has been excluded from this study at its present level. One goal for this kind of statistical research would appear to be to provide the kind of analysis carried out by Niedercorn and Bechdolt (1969) on the Gravity Model. They showed that, starting from the behaviour of economic man, i.e. an individual who tries to maximise his individual utility, it is possible to construct general equations describing mass shopping behaviour which correspond to the empirical findings summarised in Reilly's Law.

Below the scale at which aggregate properties can be used it would appear that the prescriptive approach is needed to be certain of predefined results. In political and planning work this approach can appear in many guises from educational policy: moulding certain beliefs and codes of conduct into the behaviour of the individual, to much less acceptable forms in a police state. The designer's approach is essentially the creation of a prescriptive model and, as outlined in the introduction, there is an upper limit to the size of operation which can be efficiently undertaken on such principles. The administrative difficulties of retaining effective control become too great.

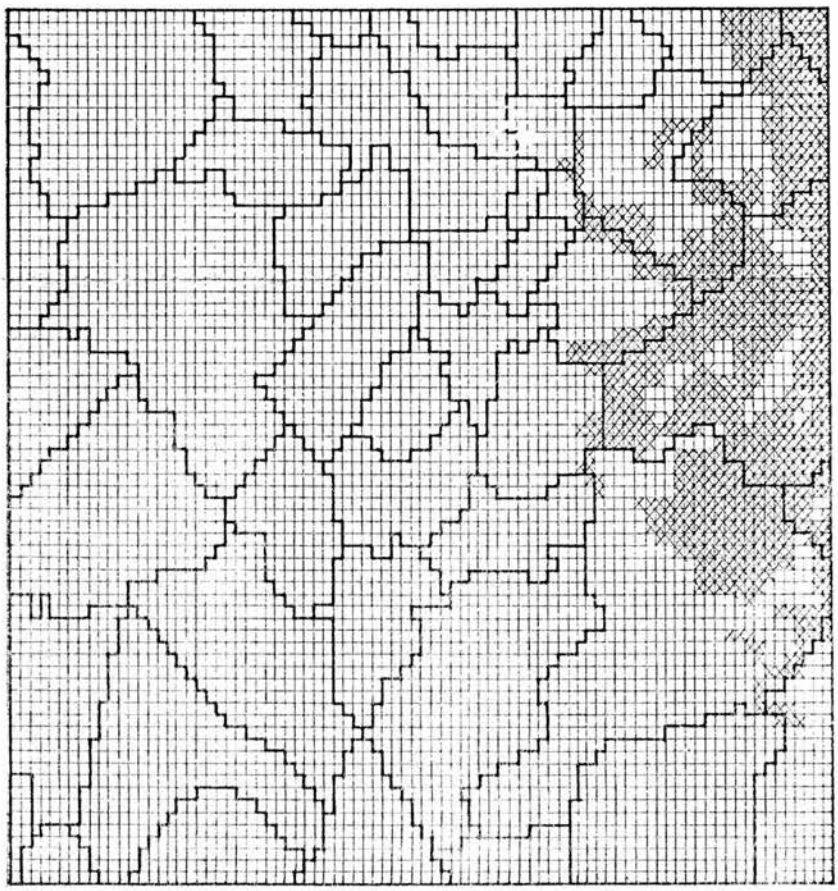
Returning to the early experiments in large-scale urban model building, though they were not as successful as perhaps it was originally hoped they would be, they provide a goal for future work. These experiments can be loosely regarded as the outcome of a top-down analysis of the

problem of applying computer techniques. The work which followed adopted the alternative approach starting with data which could be collected, processes which were relatively simple, goals which were more easily attainable, working from the existing situation, 'upwards' the aim became that of creating immediately useful techniques though, in the back of the mind, there were still the overall objectives expressed in the earlier work.

It was this more pragmatic approach which can be seen in the 'urban information system' model used as a teaching device by Steinitz and Rogers in their course 'Urbanisation and Change' initiated in the Design School in Harvard University in 1967. The first stage in this course was to collect from all available sources information about a study area and to create a data base for the area disaggregated as far as possible to a grid. The example given in Map 1 from a study carried out in this course in 1969 shows the basic population information for the 'Metropolitan Core' of Boston. The grid used in this study is given in Figure 3. Data was collected either disaggregated to individual cells, or to the larger 'town areas' shown by the heavier boundaries in this diagram. By gridding the larger collection zones, it was possible to create average grid cell figures from the figures collected by different agencies for each township.

The shopping centre allocation model shown in the flow diagram in Figure 4 was based on population data from Census sources. The income distribution was created by developing a weighting factor for each grid cell by classifying house types from aerial photographs. The existing shopping centres were defined and the quantity of their total

CONCLUSIONS



Data Collection Grid

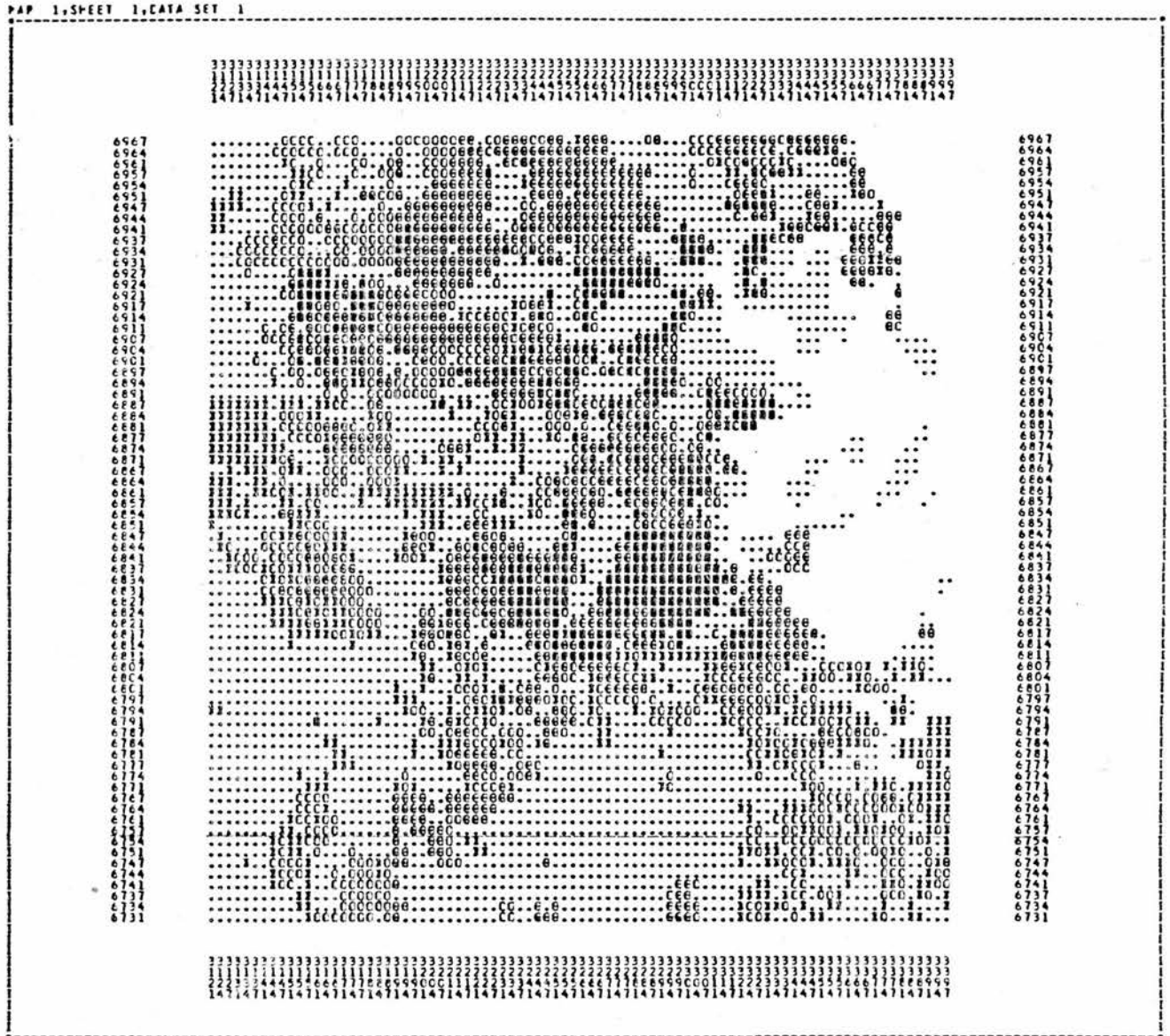
Figure 3.

sales provided by a census survey of Shopping Centres. From this starting point two results were desired: firstly, a method of assessing whether the existing pattern of shopping was providing an adequate service, or whether there were openings for further additions in the form of new shopping centres. Secondly, obviously based on the success in achieving the first objective, a way of allocating shopping services for new population which resulted from either internal redistribution, migration or internal growth.

The approach adopted was simple in the context of many shopping studies. Based on the existing distribution of people and therefore income, and

CONCLUSIONS

MAP 1,SHEET 1,DATA SET 1



POPULATION PER GRID CELL IN HOUSEHOLDS

COMMERCE REEL - TREASURE * THOMAS
CCURSE - ORGANISATION AND CHANCE STEINITZ * ROGERS, SPRING TERM 1969

DATA MAPPED IN 6 LEVELS BETWEEN EXTREME VALUES OF 0.0 AND 600.00

ABSOLUTE VALUE RANGE APPLYING TO EACH LEVEL
(*MAXIMUM* INCLUDED IN HIGHEST LEVEL ONLY)

MINIMUM	0.0	1.20	15.00	36.00	120.00	300.00
MAXIMUM	1.20	15.00	36.00	120.00	300.00	600.00

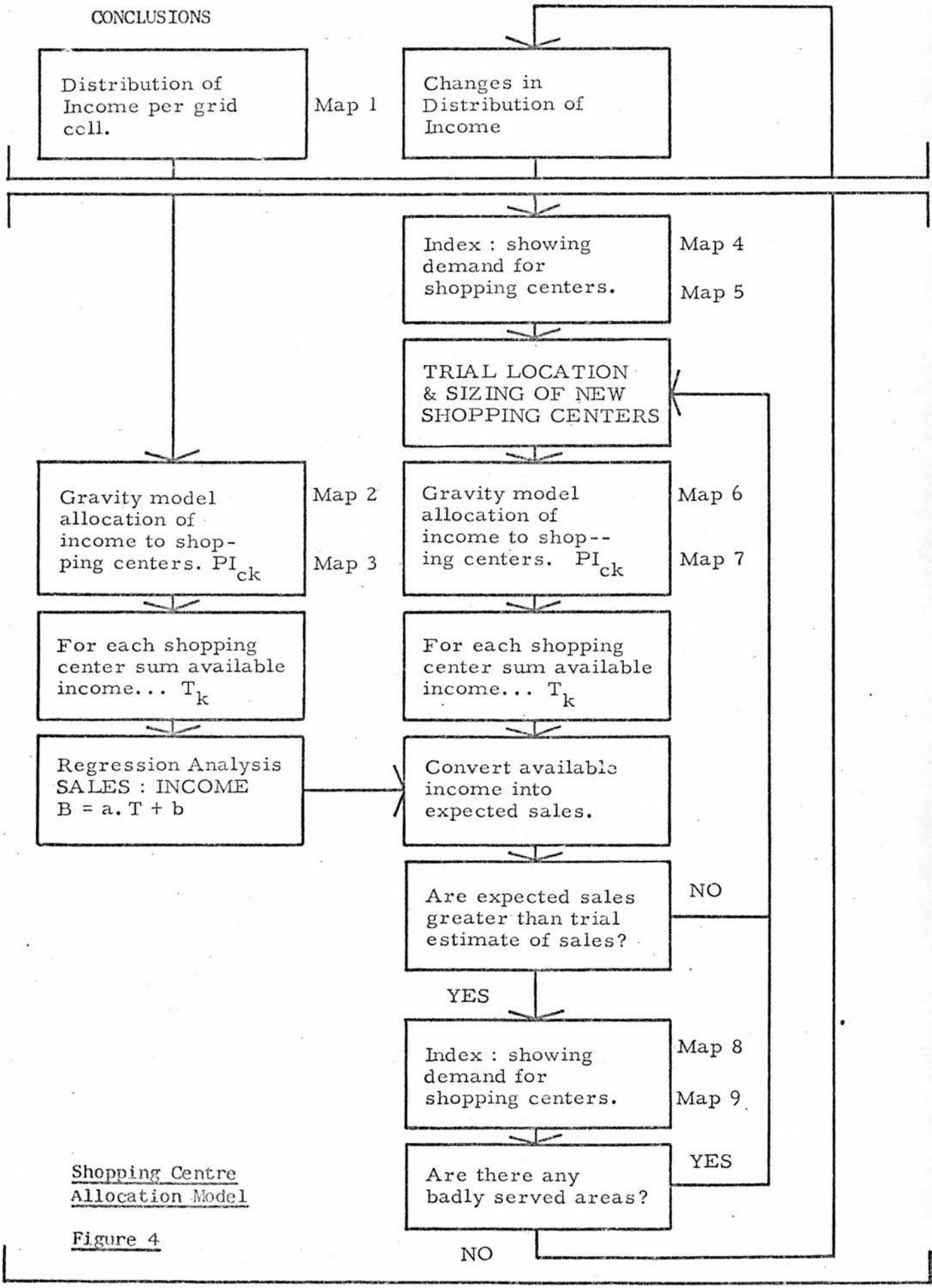
PERCENTAGE OF TOTAL ABSOLUTE VALUE RANGE APPLYING TO EACH LEVEL

MINIMUM	1.20	1.30	3.50	14.00	30.00	50.00

FREQUENCY DISTRIBUTION OF DATA POINT VALUES IN EACH LEVEL

LEVEL	1	2	3	4	5	6
SYMBOLS	.	+	1	0	@	#
FREQUENCY	2535	0	577	871	941	362

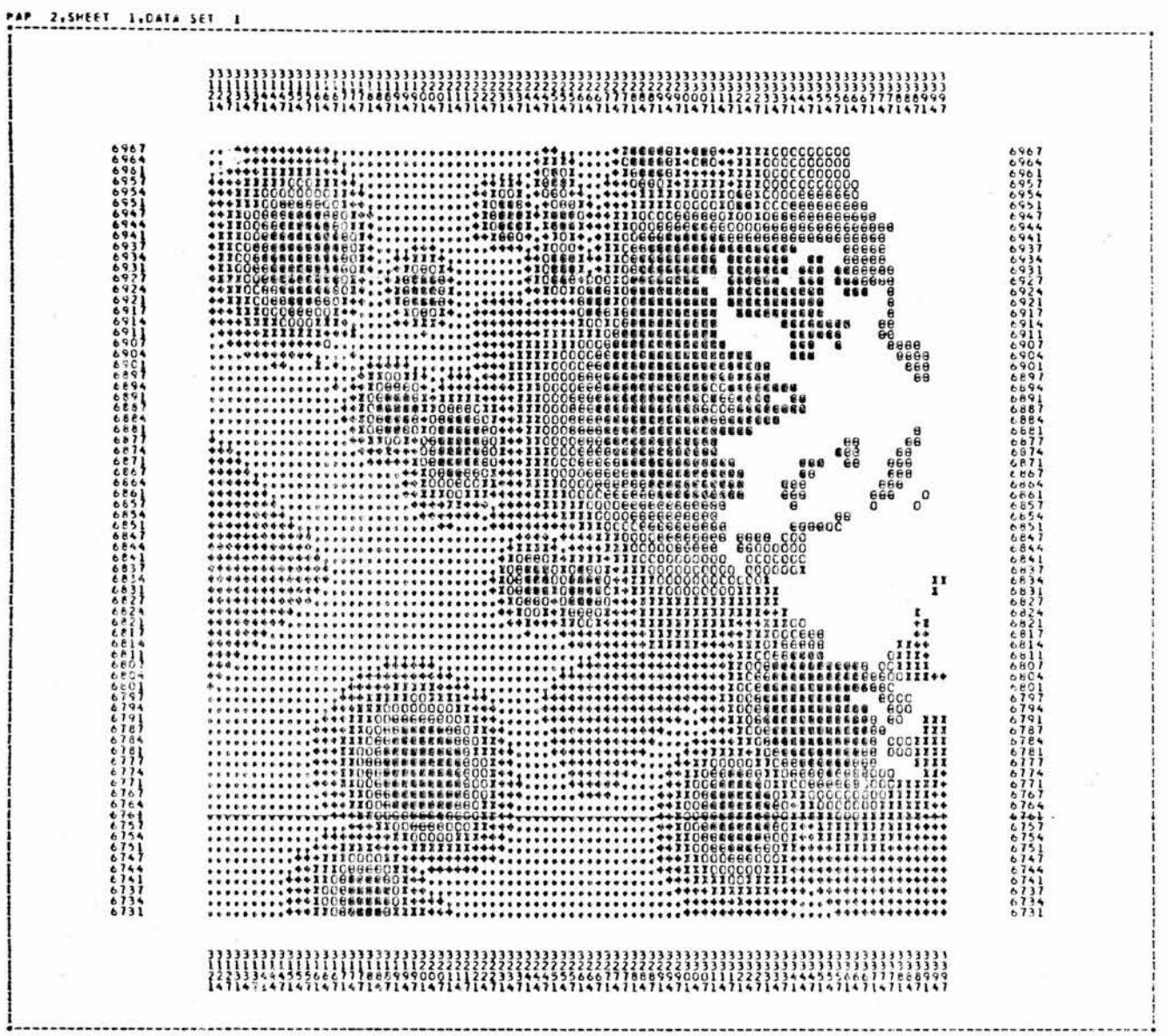
Map 1



Shopping Centre Allocation Model

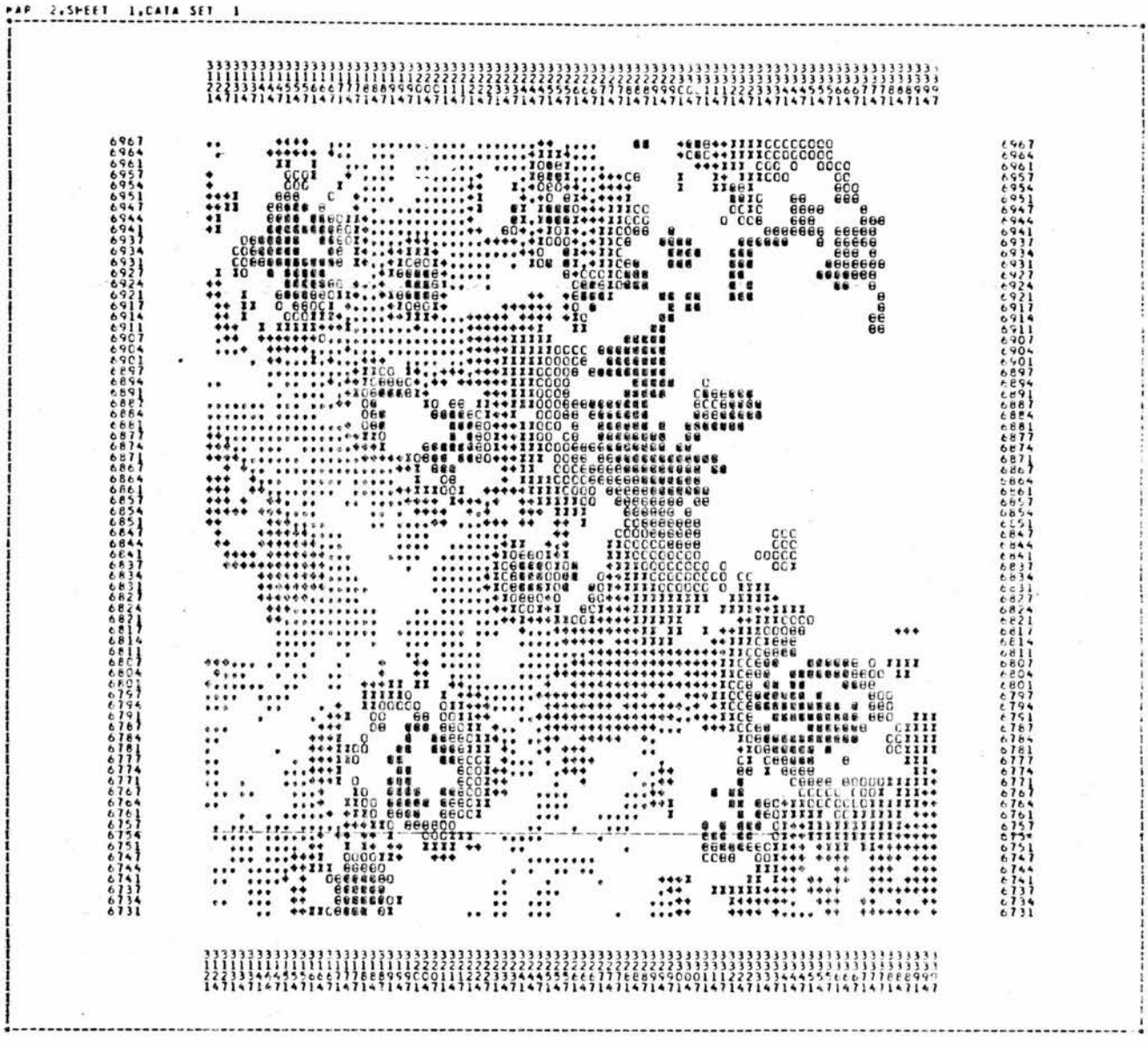
Figure 4

CONCLUSIONS



Map 2

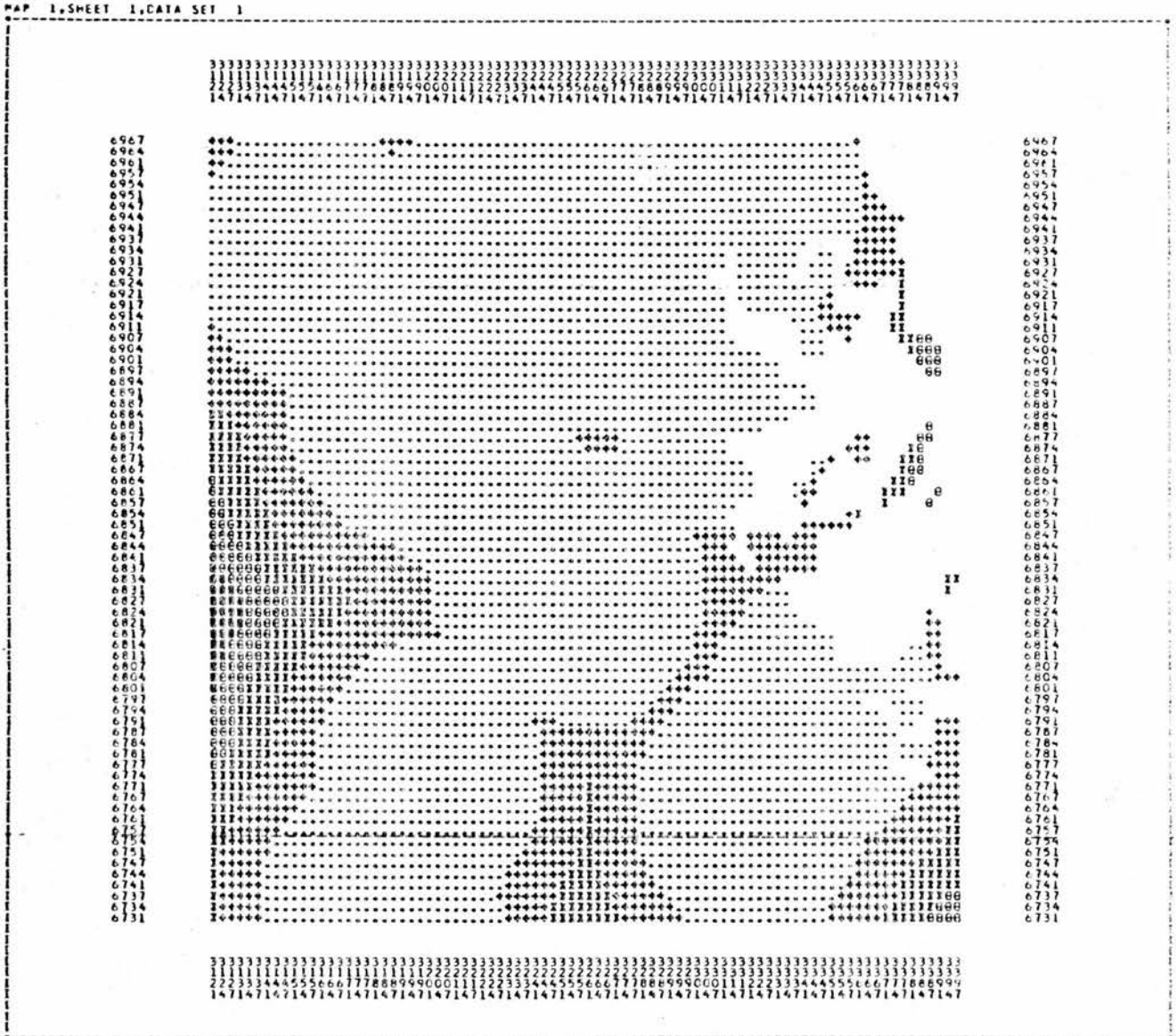
CONCLUSIONS



Map 3

CONCLUSIONS

MAP 1,SHEET 1,DATA SET 1



SERVICE LEVEL = DISTANCE SQUARED TO THE NEAREST SHOPPING CENTRE

COMMERCE MODEL - TREASURE & THOMAS
COURSE - URBANISATION AND CHANGE STEINITZ & ROGERS, SPRING TERM 1969
MAP TO SHOW AREAS NEEDING NEW SHOPPING CENTRES IN BOSTON S.M.S.A.

DATA MAPPED IN 5 LEVELS BETWEEN EXTREME VALUES OF 0.0 AND 8869.00

ABSOLUTE VALUE RANGE APPLYING TO EACH LEVEL
('MAXIMUM' INCLUDED IN HIGHEST LEVEL ONLY)

MINIMUM	0.0	1773.80	3547.60	5321.40	7095.20
MAXIMUM	1773.80	3547.60	5321.40	7095.20	8869.00

PERCENTAGE OF TOTAL ABSOLUTE VALUE RANGE APPLYING TO EACH LEVEL

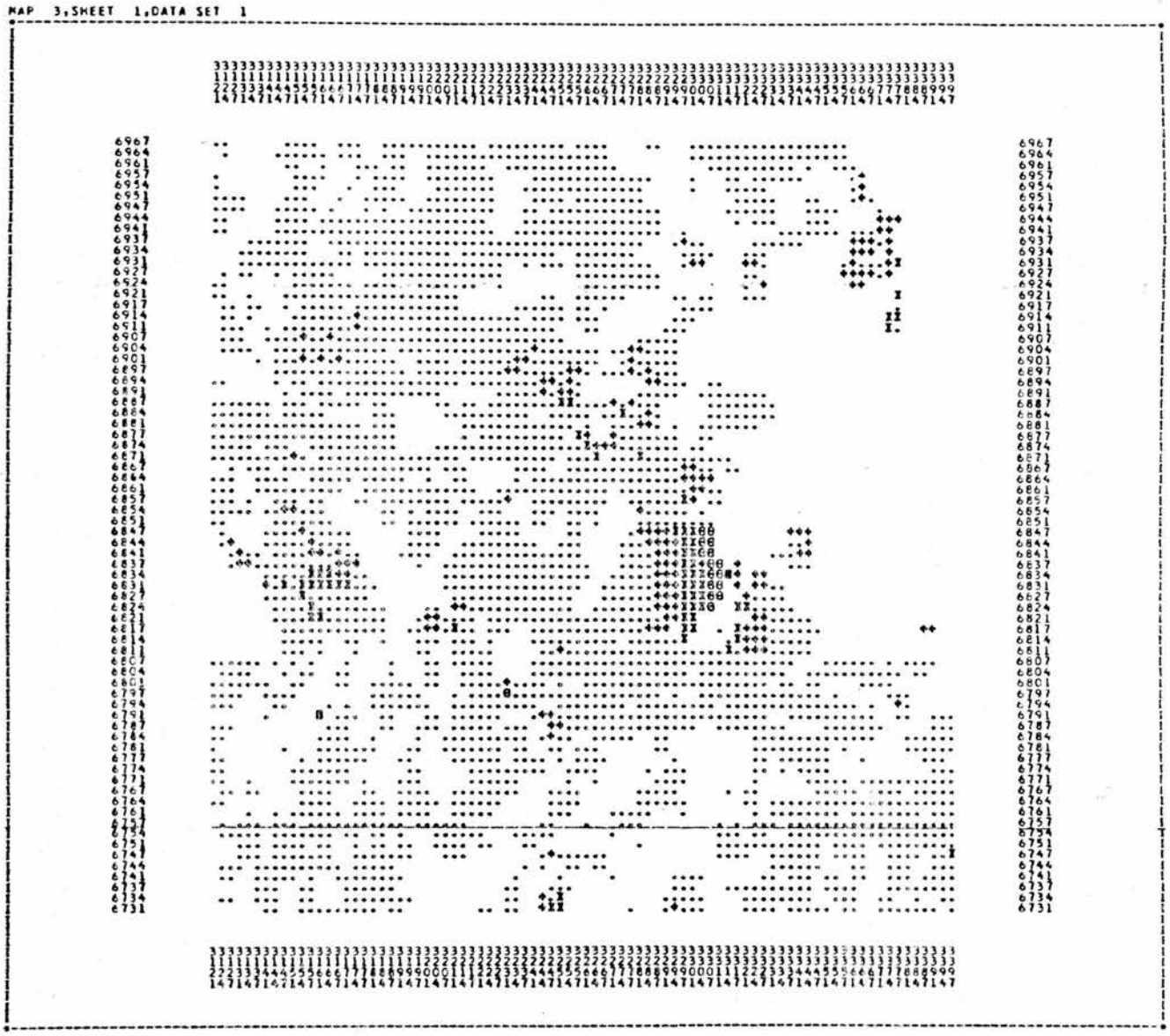
MINIMUM	20.00	20.00	20.00	20.00	20.00

FREQUENCY DISTRIBUTION OF DATA POINT VALUES IN EACH LEVEL

LEVEL	1	2	3	4	5
SYMBOLS	. + I 8 #	. + I 8 #	. + I 8 #	. + I 8 #	. + I 8 #
FREQUENCY	3936	907	295	120	28

Map 4

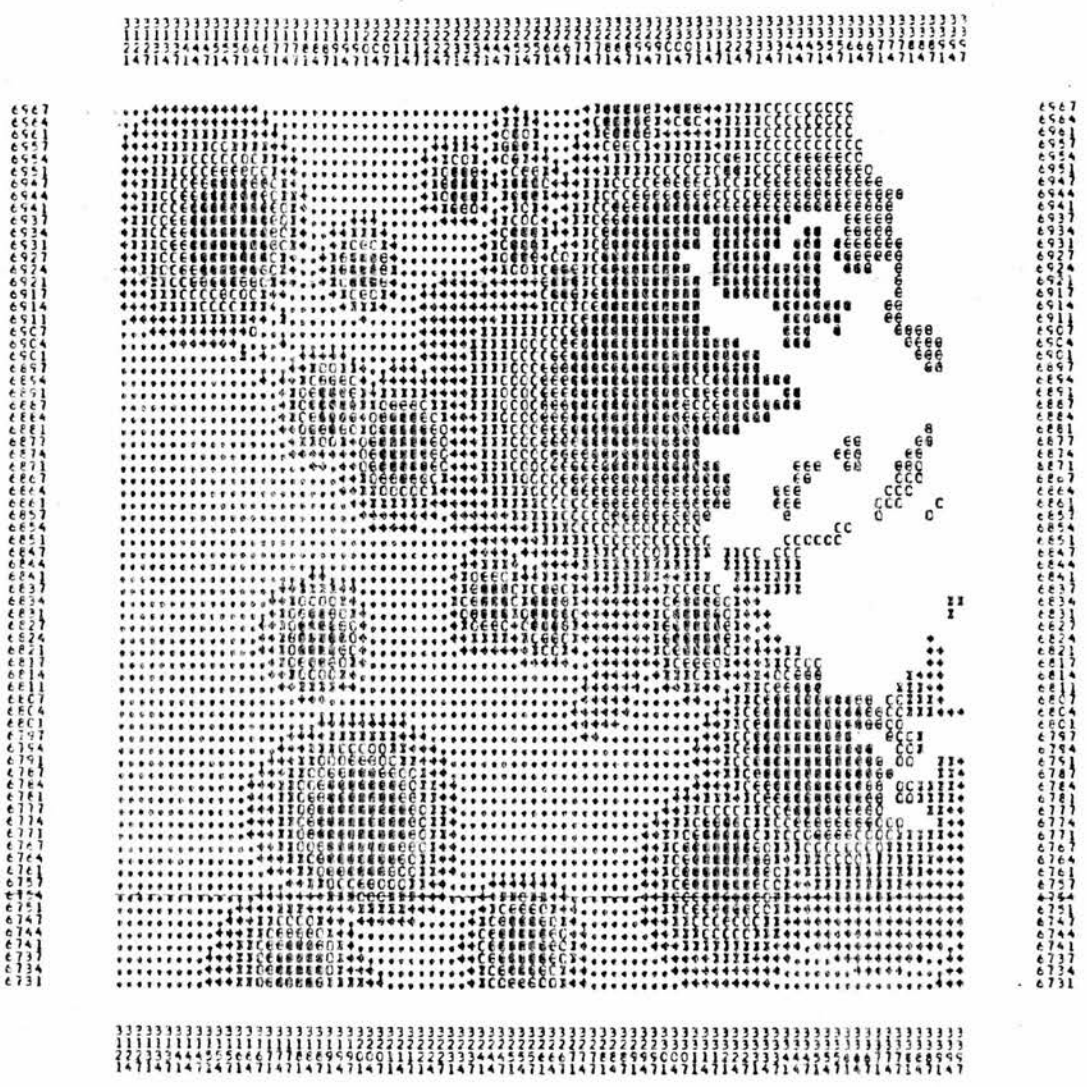
CONCLUSIONS



Map 5

CONCLUSIONS

MAP 4,SHEET 1,DATA SET 1



MAX. PERCENT OF CELL INCOME ALLOCATED TO ONE SHOPPING CENTRE

CONFIDENCE LEVEL - TREASURE * THOMAS
COURSE - ORGANISATION AND CHANGE STEINITZ + ROGERS, SPRING TERM 1969
MAP TO SHOW SERVICE AREAS OF SHOPPING CENTRES IN BOSTON S.M.S.A.

DATA MAPPED IN 10 LEVELS BETWEEN EXTREME VALUES OF 0.0 AND 1.00

ABSOLUTE VALUE RANGE APPLYING TO EACH LEVEL
("MAXIMUM" INCLUDED IN HIGHEST LEVEL ONLY)

MINIMUM	0.0	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90
MAXIMUM	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00

PERCENTAGE OF TOTAL ABSOLUTE VALUE RANGE APPLYING TO EACH LEVEL

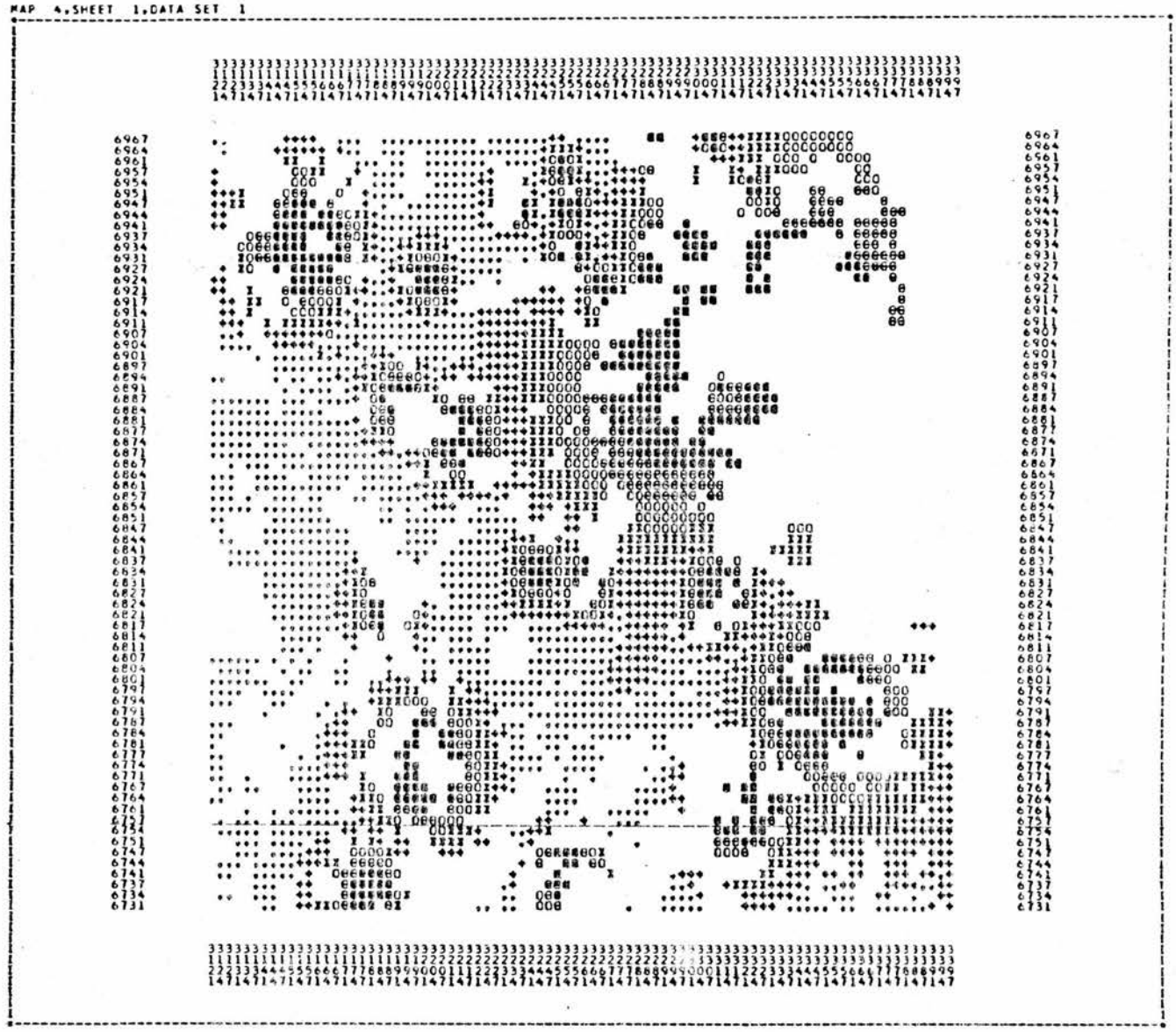
MINIMUM	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00

FREQUENCY DISTRIBUTION OF DATA POINT VALUES IN EACH LEVEL

LEVEL	1	2	3	4	5	6	7	8	9	10
SYMBOLS	.	:	!	/	+	1	C	@	#	8
FREQUENCY	0	31	1795	952	665	533	472	366	275	197

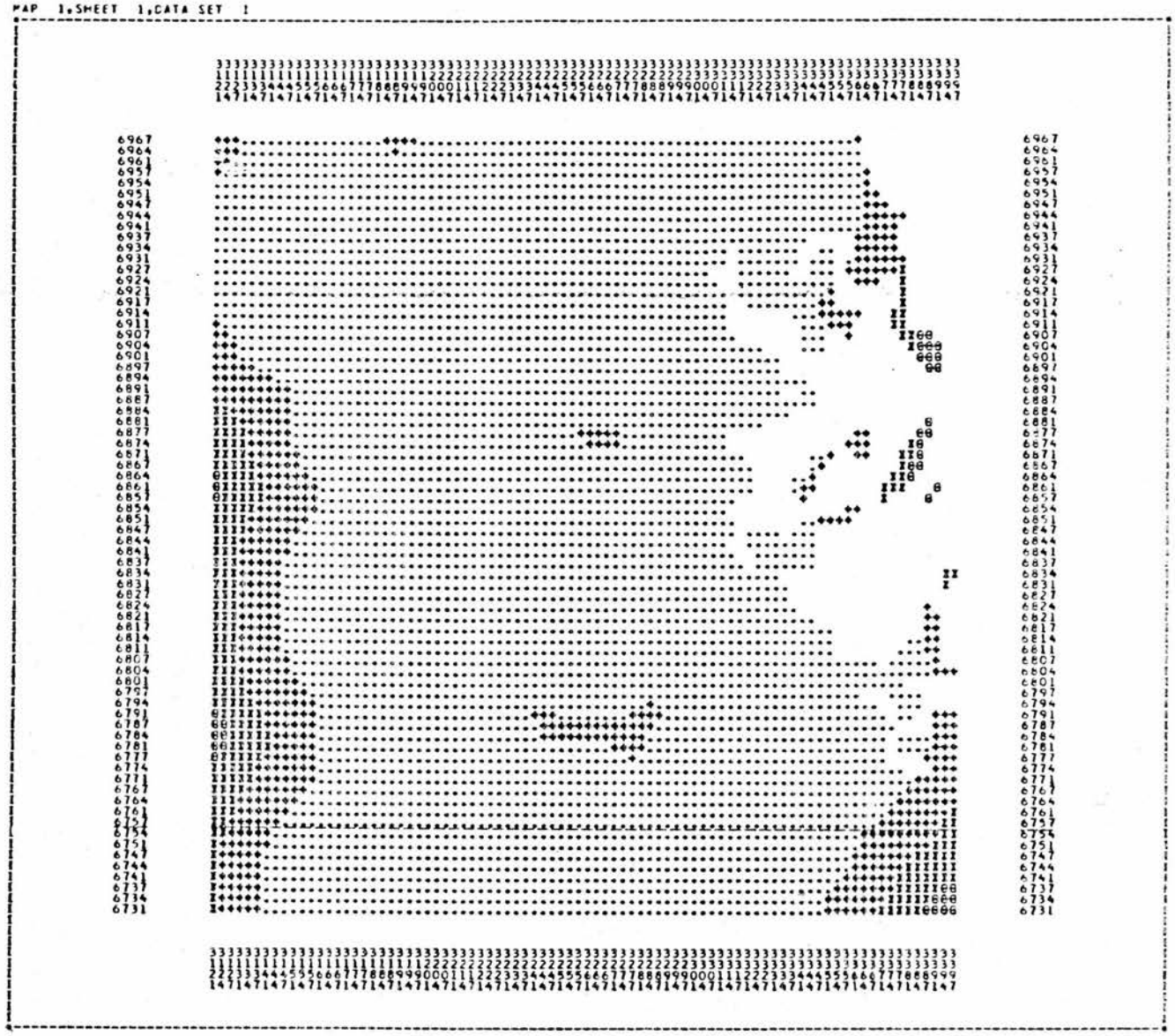
Map 6

CONCLUSIONS



Map 7

CONCLUSIONS



SERVICE LEVEL = DISTANCE SQUARED TO THE NEAREST SHOPPING CENTRE

COMMERCE MODEL - TREASURE * THOMAS
COURSE - URBANISATION AND CHANGE STEINITZ * ROGERS, SPRING TERM 1969
MAP TO SHOW AREAS NEEDING NEW SHOPPING CENTRES IN BOSTON S.M.S.A.

DATA MAPPED IN 5 LEVELS BETWEEN EXTREME VALUES OF 0.0 AND 8869.00

ABSOLUTE VALUE RANGE APPLYING TO EACH LEVEL
("MAXIMUM" INCLUDED IN HIGHEST LEVEL ONLY)

MINIMUM	0.0	1773.80	3547.60	5321.40	7095.20
MAXIMUM	1773.80	3547.60	5321.40	7095.20	8869.00

PERCENTAGE OF TOTAL ABSOLUTE VALUE RANGE APPLYING TO EACH LEVEL

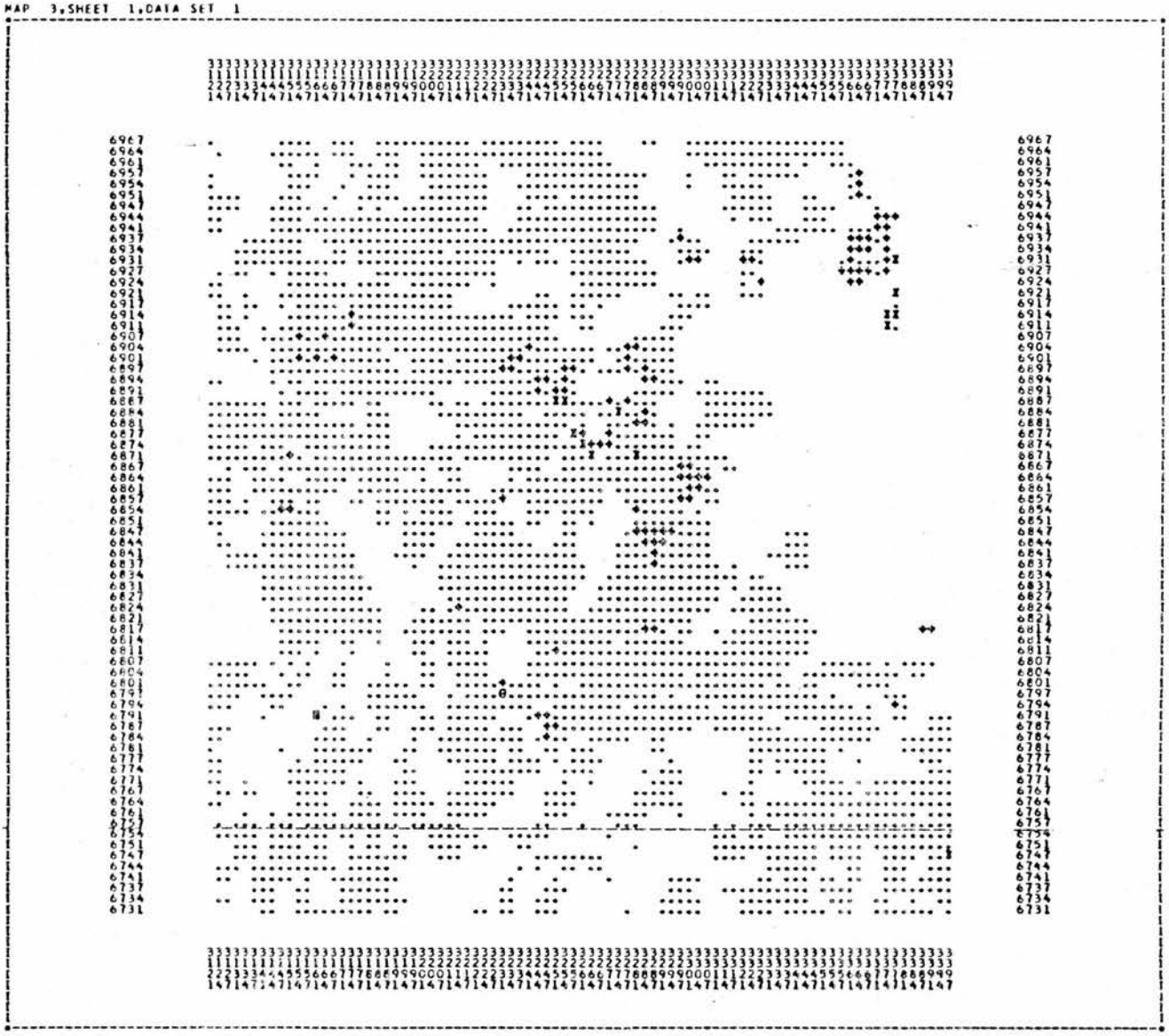
MINIMUM	20.00	20.00	20.00	20.00	20.00

FREQUENCY DISTRIBUTION OF DATA POINT VALUES IN EACH LEVEL

LEVEL	1	2	3	4	5
SYMBOLS	. + I @ #				
FREQUENCY	4462	563	221	40	0

Map 8

CONCLUSIONS



SERVICE LEVEL * DISTANCE SQUARED * POPULATION

COMMERCE MODEL - TREASURE * THOMAS
COURSE - URBANISATION AND CHANGE STEINITZ * ROGERS, SPRING TERM 1969
MAP TO SHOW AREAS NEEDING NEW SHOPPING CENTRES IN BOSTON S.M.S.A.

DATA MAPPED IN 5 LEVELS BETWEEN EXTREME VALUES OF 0.0 AND1133399.00

ABSOLUTE VALUE RANGE APPLYING TO EACH LEVEL
('MAXIMUM' INCLUDED IN HIGHEST LEVEL ONLY)

MINIMUM	0.0	226679.75	453359.50	680039.25	906719.00
MAXIMUM	226679.75	453359.50	680039.25	906719.00	1133399.00

PERCENTAGE OF TOTAL ABSOLUTE VALUE RANGE APPLYING TO EACH LEVEL

MINIMUM	20.00	20.00	20.00	20.00	20.00

FREQUENCY DISTRIBUTION OF DATA POINT VALUES IN EACH LEVEL

LEVEL	1	2	3	4	5
SYMBOLS	. + I c #				
FREQUENCY	3228	106	13	1	1

Map 9

shopping centres, two types of map were produced. The first, from the point of view of the individual, shopping from a home base gave each cell an index of the level of service provided at its location by the existing pattern of shopping centres (Map 4). When this primitive measure of service was weighted by the population density of the cell and by its income, it gave a map (Map 5) of the areas which needed more detailed analysis, with a view to providing new shopping facilities. The second set of maps was constructed using the gravity model principle to allocate people to different shopping centres. This permitted available shopping income at a particular shopping location to be assessed and, when a new centre was proposed, to see the effect of locating it in a variety of different situations. In Maps 2, 3, 4 and 5 the initial analysis of the study area is shown. In Maps 6, 7, 8 and 9 the same analysis is shown after locating three new shopping centres where the previous analysis suggested a poor level of service existed. The result is an improved level of service but only two of the centres were able to attract more available income than their initial estimated size. Thus, the third centre was assumed not to be a viable development.

The basic system used in this study was a simple grid data-base associated with a primitive line printer mapping package. Each model was designed to draw information from this data base and supply new and derived data back into it. These facilities, however, only provide an automated accounting system to provide the maps and other documents needed in planning decision making. The other component of the teaching exercise was a gaming simulation of the decision-making necessary to resolve

conflicting interests between different sectors of the whole community. The models provided, at least potentially, a more effective way of assessing the external interaction of decisions made within these sectors. For resource management this was an improvement on the situation where decisions were made only on the basis of discussion between specialists. By providing a role-playing situation, it is possible to ensure that different specialists are relating decisions in their own area of expertise with that of others. The combination of gaming and modelling provided both a way of using expert knowledge as effectively as possible and, at the same time, showing the cumulative and interactive effect of more than one point of view.

The work in this research project was initially prompted by a desire to improve and extend the basic facilities provided in this kind of system. The general kinds of information considered in planning can be classified as relating to people, activities, land, or their interaction with each other. The initial goal set for study was to represent the properties of the physical environment as a computer model. The first stage of this work consisted of a study of computer cartography. However, it seemed that because of the problem of data collection, it was necessary to start work at a much finer level of detail than is represented in most maps and, at this scale, it became important to represent the three-dimensional form of the land and buildings on it.

The basic assumption was that information about the physical environment could be represented by two components: the first as a description of the location and extent of an object - in other words, its spatial characteristics; the second as a definition of what it is - in other

words, the content of the space defined by the first component. The aim was to construct the first component as a three dimensional model.

The objective was to construct the spatial description as far as possible using existing sources of data. This approach made it necessary to divide the problem into two parts: firstly, the creation of the three-dimensional topographic surface; and secondly, the creation of objects on this surface such as roads and buildings. This distinction was made more because of the way that topography and building forms are represented in maps or technical drawings than because of any intrinsic difference in the three-dimensional description of these two elements.

Topographic Surfaces

Starting with a topographic map of an area, perhaps the simplest three-dimensional model of the surface can be obtained by overlaying the map with a rectangular grid and determining the height values at each grid node. This was the procedure adopted in preparing data for the drawing in Figure 5 . This is a very convenient form in which to hold the data, since the height values can be stored in a table which matches the original sampling grid. The position of a value in the table acts as a way of defining its other two coordinates, since they are effectively the row and column numbers of its position in the table. This table can be converted into a line printer map merely by substituting the numbers by the appropriate symbols and printing out the table.

The same table of numbers, when used as data for a contouring program,

can be used to generate a drawing such as that shown in Figure 5. The contouring program assumes a simple surface linking each of the point values round a grid cell and calculates the intersection line between this surface patch and the previously defined contour planes. The accumulation of these line segments, when they are drawn out, gives the contour map. It can be seen that this model can be used to generate any set of contour lines which might be required.

Computer generated contour map of Honey Hill, New Hampshire, U.S.A.

Figure 5

It is possible, starting with this form of surface model, to derive two-dimensional data which can then be used with other graphic programs.

CONCLUSIONS

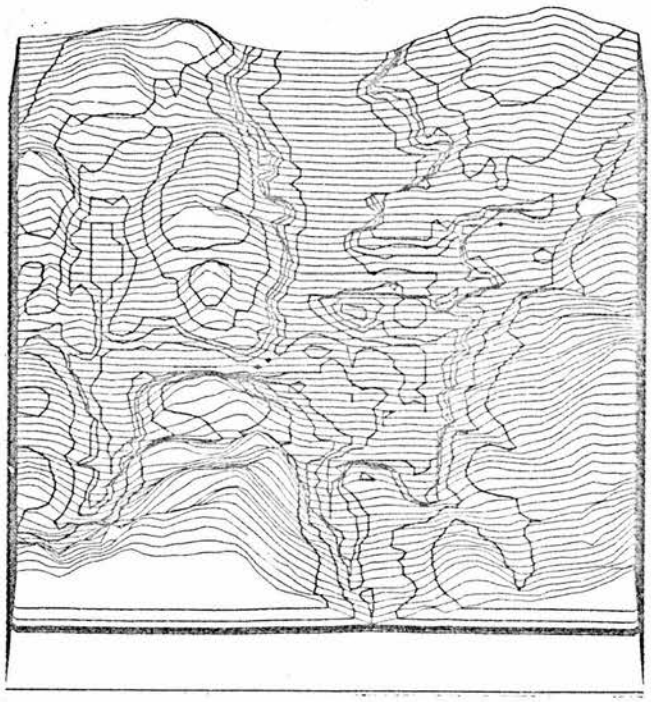
Computer generated contour map of Edinburgh with relative height indicated by line shading symbolism

Figure 6

In Figure 6 the contour line segments were generated in the same way as

the previous example. They were then sorted into order to give the closed boundaries of the areas between the contour lines. These areas were then shaded to indicate the relative height of each contour band, so making it possible to tell the high areas from the low areas in the resulting map.

Figure 6 shows the topography of Edinburgh, using six equally-spaced contours from Oft. to 500ft., the two areas above this level being the Pentland Hills and Arthur's Seat.



Computer generated block model from the same data used to generate Fig

Figure 7

Having the data in a three-dimensional form makes it possible to produce projected drawings such as the block model in Figure 7. This drawing was produced from the same data as Figure 5. In this case, the block model was produced to study the visual impact of a new reservoir.

CONCLUSIONS

However, this form of drawing has a variety of uses. For the landscape architect, it allows the effect of different forms of land grading to be assessed. One use that such drawings have been given is in the location of radar sites to cover the flight paths of low flying aircraft through mountainous terrain. Generally speaking, this form of drawing is of most use where the location of a road or building needs to be hidden from view, or conversely sited to get a good view. One of its advantages over plan-views is that it is possible to plot extra information onto the surface of the block model. An interesting example of this was suggested for oil exploration. If the upper surface of the oil-bearing strata is drawn in this way and contour lines of the strata's thickness is drawn onto this surface, then good locations to sink wells could be assessed from the final drawing.

Surfaces with Superimposed Objects

Having briefly considered the representation of topographic surfaces where there were no buildings or other man-made features which need to be included in the model, or where the scale was large enough for them to be ignored except perhaps for surface symbols, the next stage was to consider the models necessary within a city. Though the representation of building structures demands techniques which are being developed for computer-aided design work, generally the requirements are much less detailed.

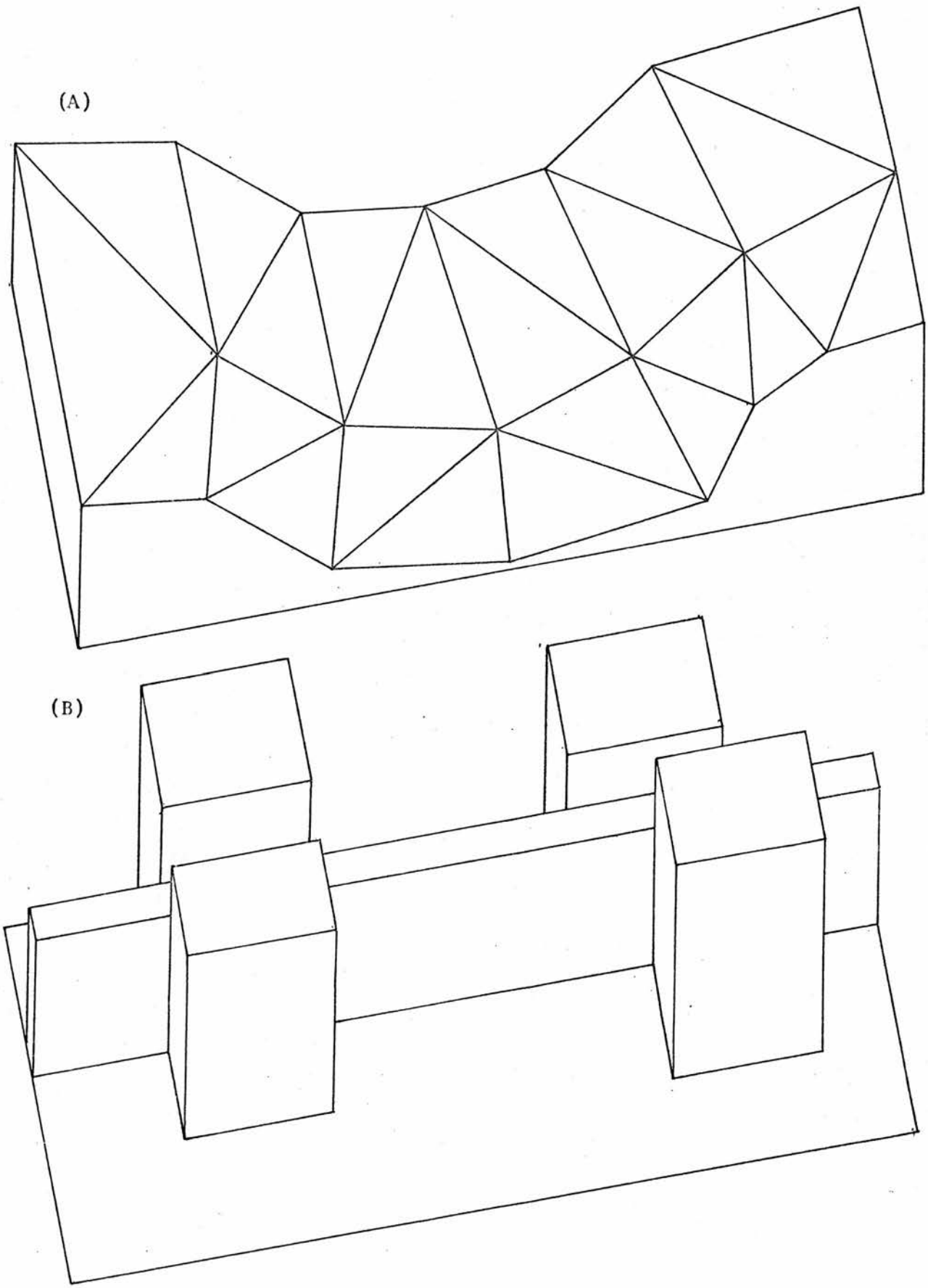
In a city map buildings are usually shown by boundary lines and the remaining area of the map is taken up by roads, parking areas, parks and

other features which, generally speaking, will follow the original topographic surface. For this reason, the approach which has been taken was to create two models to start with, the first representing the original topographic surface, and the second the vertical blocks of the building forms erected as prisms on their plan boundary lines. These two models could then be merged together automatically to give the final model of the buildings set in their correct landscape. Although, for some cities, the topography could be considered to be a plane surface as far as most planning purposes are concerned, Edinburgh is a case where this obviously could not be done.

At the scale of city maps, topography, within a built-up area, can be approximated by plane areas more easily than for the open countryside. For this reason, an alternative model can be used to the regular grid of height values discussed above. A block model using plane facets is shown in Figure 8A . The basic data for this model is a series of point height values. Since these values are not in a grid, they must be expressed as complete three-dimensional coordinates. These point height values can be triangulated to give a surface made up from plane facets as shown in the diagram.

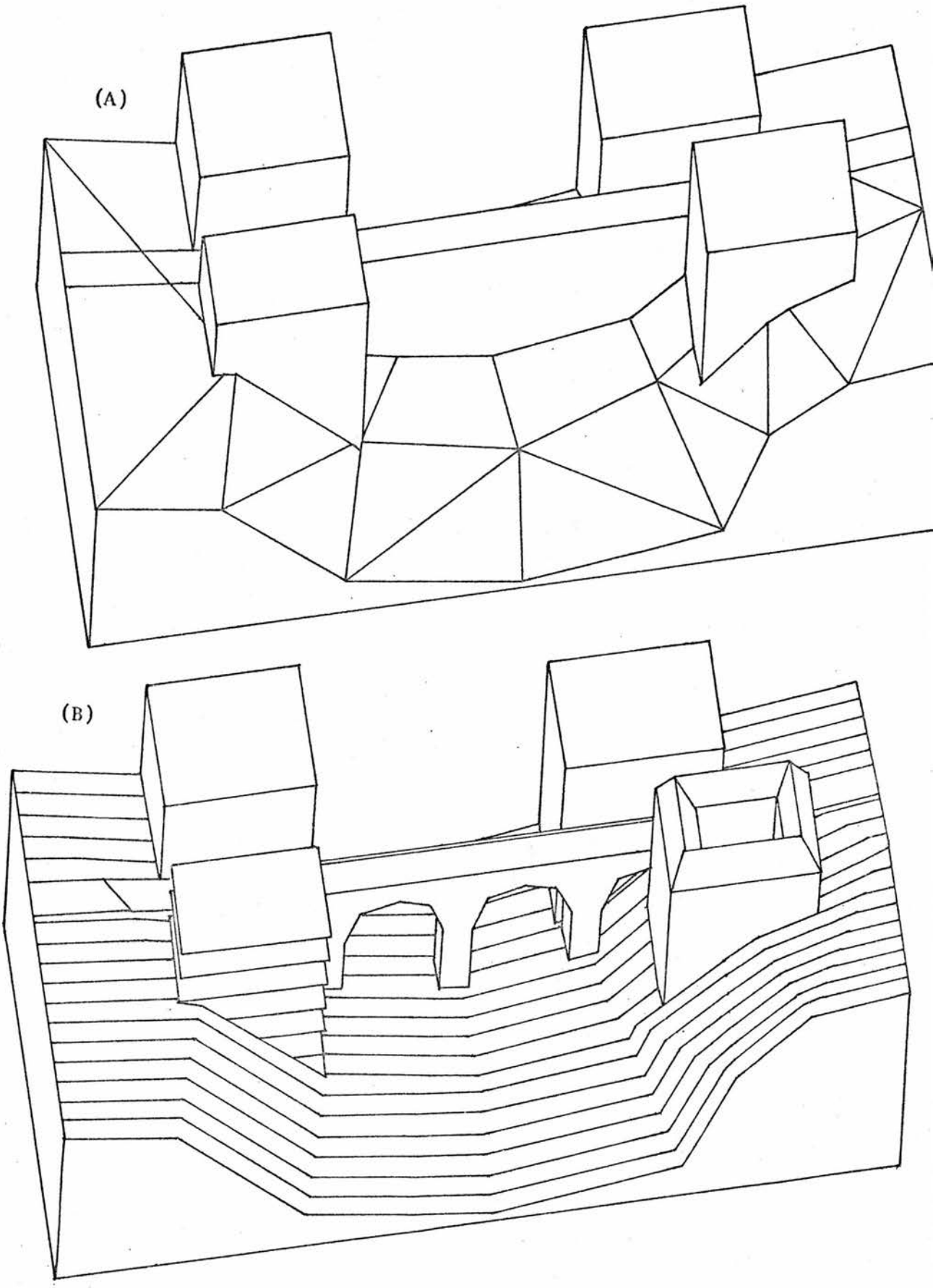
The problem with triangulation, given a collection of points on a map, was to choose which triangulation. To get consistent results from this process, the triangulation program was based on a nearest neighbour algorithm. This linked the points together in the same way, even if the whole distribution of points was rotated to a new orientation relative to the map frame. Once these triangles had been created, they could then be used for creating drawings, for calculating surface areas,

CONCLUSIONS



Data Base Construction I

Figure 8



Data Base Construction II
Figure 9

CONCLUSIONS

calculating volumes, determining drainage basins and so on.

Having created the topographic surface the next stage was to create the prisms representing the building forms. The basic data for this model was the polygon boundaries of the building plans in their correct location on the base map. These outlines, which can be created in a similar way to the polygon boundaries required for choropleth maps, are needed with the height of the buildings' roofs above the base plane. The prism for the building can then be generated automatically by duplicating the plan polygon in a horizontal plane at the height of its roof, and then linking the corresponding vertices in the plan and roof polygons. The result of this process is shown in Figure 8B . The final description can be formed as a collection of polygon boundaries so that this model is in the same form as that of the triangulated topographic surface.

Once both models have been created, they can then be merged automatically to give the result shown in Figure 9A . Though the model at this stage is three dimensional, it is often referred to as a 2½ D model. If the blocked-in version of the bridge is considered, the reason becomes apparent. Using this construction process, it is not possible to create overhangs or, in this case, the arches of a bridge, which is the consequence of starting with relatively simple data.

It can be seen that to create Figure 9A from 8A and 8B means that, where the surface facets cut each other, they have to be redefined. Only those parts of a facet which do not fall within the solid section

or interior of another volume are required. The reason for adopting this two-stage operation is apparent if the problem of creating vertical facets using the method for generating the topographic surface is attempted.

Though this model is fairly crude, it need only be a starting point. By using appropriate editing and updating procedures it is possible to improve sections of this basic model as data becomes available or as it becomes necessary. In Figure 9B two possible modifications to the building forms are shown. On the right the original block has been edited to give a better visual representation of the building form. In this case, the creation of arches in the bridge is primarily for visual reasons. An alternative modification is shown on the left. The building block has been converted into a series of open floor levels. If all the buildings are represented in this way it becomes possible to generate a very detailed three-dimensional land-use or floor-use data base. This may only be worth attempting in city centres. However, it can be seen that, from this model, many useful statistics such as gross floor area used for shops could be calculated for any area or sub-area in the city. Such information, when used for defining clusters of shops in shopping models would provide detailed information which, at present, is too time-consuming to calculate. The groundscape in this diagram has been modified to give a better visual effect. It is possible to shade these facets in a variety of ways using a similar technique to the one used in generating contour lines in previous examples.

The processes needed to edit and update the basic block model were more

complex than those needed to set it up in the first place. The initial stages - the preparation of the data and the creation of the drawings - could be carried out using a batch processing system. However, it is hard to see how the editing can be done efficiently without using some form of interactive system.

To prepare a detailed model of a city structure, even to the level giving the main floor levels in each building, would be a fairly major undertaking. However, once this task was done, it would not have to be repeated. The life-span of buildings being in the range 20 to 100 years, once the original model has been set up, it would only require the modifications resulting from new buildings and demolitions to be made. Since planning permission is required for new buildings, the data for such modifications should become available automatically. In fact, it is not unreasonable to speculate that when architects start to use computer-aided design techniques, such submissions would be made in machine readable form.

The three-dimensional modelling of objects which was described in Section III, though originally undertaken in the context of Urban Systems Analysis, finds its natural application in computer-aided design work. It was important to start the discussion of applications in the area of City Planning because it was an investigation in this area which showed the importance of taking a general approach to representing information. If data structures used in Architectural and Engineering computer-aided design work can easily and naturally provide input data to urban systems, data-bases, then the problem will not be one of data collection where separate people have to rework

subjects to create data, but be one of data selection and data compaction which can be automated.

The review of modelling, though not strictly speaking belonging in a Chapter providing the conclusions to this research work, seemed necessary to illustrate a difference in approach between that taken in the initial applications of comput rs to planning problems, and the one which is adopted in the present discussion. The belief which guided the choice of subject was that, though the original goals of large-scale urban models might well be attained, it will only happen in a general way when the necessary data about human activities become available automatically from other machine-based activities. This requires an exploration of the way in which activities carried out by people, which need to be coordinated or monitored, can be made self-recording without the need for a large number of intermediate data collectors. The present solution to this problem, which is to make it a statutory responsibility for individuals, for example to file tax returns, apply for licences, submit plans for planning permission, though possibly appropriate for these applications, cannot be extended very far. The specific activities considered from this point of view have been the design and communication activities which use graphics.

Computer-Aided Design Applications

In the remaining parts of this Chapter, the applications considered are much more directly related to the detailed work in the thesis. As a result, they can be presented more briefly and more directly than can applications to city planning.

CONCLUSIONS

The application of the volume manipulation language and the different display facilities to the designer should make it possible to create a machine model of the object being designed, as decisions are made rather than creating drawings. It is hard to say whether this will save design time, though it is clear that such a process will not be accepted if it increases the effort and time of the designer. Where the advantage seems to lie is the greater versatility of the machine model over the collection of drawings. Where necessary, the machine model can be used to provide data for many analytical processes for which using drawings would be impractical.

In creating physical descriptions of objects as drawings, there are many procedures which are trivial but which have to be worked out in detail. Given the appropriate facilities, a computer-aided system should be able to allow the designer to work with higher level descriptions and allow the machine system to check and implement the lower level work which, at present, still needs detailed attention. In Architectural design there are many examples of this kind of work. Checking that a final design has correct head-room for all stairways, that all doors and windows can be opened without clashing, that all the requirements of building codes and planning regulations are being met, is time-consuming and rarely creates major problems if faults are found in time. If this area of bookkeeping work can be achieved automatically, then design effort can be given to more important general problems.

CONCLUSIONS

Two particular applications can be developed from the work presented in Section III for Architectural and Engineering work. The first is the ability to build up a functional description of the relationships in a building so that dimensions and exact locations need not be specified until all the information necessary to do so is available. An example of this occurs in the case of coordinating different modular elements. Where a variety of different elements is required in alignment, for example tops of doors, wall-tiling, cupboard tops and so on, and where tiles come in a variety of sizes, doors are limited both by manufacturing standards and local bye-laws, then the datum position can be defined as a relationship which will not be converted into a real location until the necessary choices have been made. More than this, as soon as a choice is made which makes a relationship impossible to achieve, then the system can give the designer warning of the clash.

The ability to create relational descriptions makes it possible to create archytype solutions to standard problems. The joints in a door frame can be defined generally, and as soon as the door frame is dimensioned, the system can automatically create the necessary descriptions of its components.

The second specific application appears in what Comba (1968) calls the placement problems. When using a computer model of volumes it is usually difficult to know whether two objects have been made to overlap. By using the approach developed in Section III, it is possible when locating a new element such as a pipe in a duct, to display only the

volume of overlap between the new object and the setting in which it is being located. If nothing is shown on the display, then the placement is successful.

It is this latter development which makes it possible to extend the use of the hardware display processor considered in Section IV to controlling automatic machine tools. If the volume being cut from a block of raw material is intersected with the volume of the cutting tool, and the result displayed, then as long as the display is empty the cutting operation is not destroying a part of the final article. This procedure can also be used to prevent the cutting head of the machine from cutting other parts of the control machinery.

In each of these applications the machine has a useful model of the extent of physical objects which it can manipulate at speed. Though creating graphic output is an important facility, it is only necessary where drawings are needed as a method of communication such as, for example, to a building contractor where automatic construction is out of the question. In the final area of applications, this form of communication is of paramount importance and occurs where people are interacting with information stored as machine data. Perhaps the most important aspect of this use, based on the discussion in Chapter 4, appears in learning and educational applications. Significantly, these two need not be the same. When a city planner moves from one location to another, much of the detailed knowledge of one area will not be of any use when transferred to the new area. If there is the appropriate access to a data base for the new area, then hopefully some of the initial stages of learning about it will take less time

and, if appropriately constructed, the data base will provide a form of continuity which is independent of particular people.

The use of animated graphics for real time simulation provides a range of applications in teaching people to coordinate complex machinery. The aircraft pilot-training simulators save aircraft companies from using aircraft in pilot-training, thus making them available to carry passengers. There are two conditions which make this form of simulation valuable. The first is where the training has to include dangerous situations which people should not be exposed to unless absolutely necessary. The second is where there is no alternative, for example in the training provided to astronauts.

Applications Employing Animated Graphics

The final areas of applications depend on the developments discussed in Section IV which, with the advances in integrated circuit technology, are more or less in striking range for creating real, rather than abstract graphic systems.

The immediate application appears in creating computer animated cartoons for T.V. displays. The use of these varies from entertainment to educational films. It appears possible to create from the output of a T.V. camera a signal structured as a boolean expression description of the image. This possibility has to be the subject for future investigation. It would appear to offer a completely different starting point for discussing the communication and use of spatial information since the representation of the image created in a camera

would become directly compatible with those created by main computer systems.

It is clear that the work which has been taken to its conclusion, for example in computer-mapping, has already shown its usefulness. Some of the later work depends on further development to establish its value. The work on the display processor being continued in Heriot-Watt University will provide answers for one aspect of this work and, if this is successful, then its application to real situations will eventually allow some of the other more speculative applications to be tried out in practice.

BIBLIOGRAPHY

BIBLIOGRAPHY

ABLER
ADAMS
GOULD
Spatial Organisation
The Geographers' View of the World
Prentice Hall International Inc., London, 1972.

ABRAHAM, R.L.
FOX, R.M.
SCHILLER, W.L.
VAN DAM, A.
A Microprogrammed Intelligent Graphics Terminal, Transactions of I.E.E.E., Vol. 20, No. 7, July 1971.

AHLBERG, J.H.
NILSON, E.N.
WALSH, J.L.
The Theory of Splines and their Applications, Academic Press, 1967.

AHUJA, D.V.
COONS, S.A.
Geometry for Construction and Display, I.B.M. Systems Journal, Vol. 7, Nos. 3 and 4, 1968.

AITKEN, W.
HOTSON, J.M.
Computing for Geographers and Environmental Scientists, Conference Paper, Lancaster University, March 1974.

ALONSO, W.
Location and Land Use: Towards a General Theory of Land Rent, Harvard University Press, Cambridge, Mass., 1964.

APPEL, A.
Modeling in Three Dimensions, I.B.M. Systems Journal, Vol. 7 Nos. 3 and 4, 1968.

The Notion of Quantitative Invisibility and the Machine Rendering of Solids, Proceedings A.C.M., 22nd National Conference, 1967.

ARMIT, A.P.
Computer System for Interactive Design of Three-Dimensional Shapes, University of Cambridge Computer Laboratory, Nov. 1970.

ARON, J.D.
Information Systems in Perspective, Computing Surveys, Vol. 5, 1973.

ASHBY, W.R.
An Introduction to Cybernetics, Chapman and Hall, Ltd., 1956.

ATKIN, R.H.
Mathematical Structure in Human Affairs, Heinemann Education Books Ltd., London, 1974.

BARBER, C.L.
The Story of Language, Pan Books Ltd., London, 1964.

BARBER, W.J.
A History of Economic Thought, Penguin Books, 1967.

BARRON, D.W.
Assemblers and Loaders, Macdonald, London, 1969.

BARRON, D.W. — Recursive Techniques in Programming, Macdonald, London, 1968.

BAUER, F.L. (Editor) — Advanced Course in Software Engineering, Springer-Verlag, Berlin, 1973.

BELL, C.G.
GRASON, J.
NEWELL, A. — Designing Computers and Digital Systems, D.E.C., Digital Press, 1972.

BERNE, E. — Games People Play, The Psychology of Human Relations, Grove Press Inc., New York, 1964.

BERRY, B.J.L.
MARBLE, D.F. — Spatial Analysis, A Reader in Statistical Geography, Prentice-Hall, Inc., 1968.

BLALOCK, H.M. — Social Statistics, McGraw Hill Book Co., 1960.

BOULAYE, G.G. — Microprogramming, MacMillan Press Ltd., 1975.

BRAID, I.C. — Designing with Volumes, University of Cambridge Computer Laboratory, Feb. 1973.

The Synthesis of Solids bounded by Many Faces Communications of A.C.M., Vol. 18, No. 4, April 1975.

BRAINERD, W.
LANDWEBER, L.H. — Theory of Computation, John Wiley & Sons, 1974.

BRASSEL, K. — Models and Trials in Automatic Relief Shading, Dissertation, 1973.

BRITTON, J. — Language and Learning, Penguin Books, 1970.

BULLOCK, N.
DICKENS, P.
STEADMAN, P. — A Theoretical Basis for University Planning, Land Use and Built Form Studies, Cambridge

BURSTALL, R.M.
COLLINS, J.S.
POPPLESTONE, R.J. — Programming in POP 2, Edinburgh University Press, 1971.

BUSACKER, R.G.
SAATY, T.L. — Finite Graphs and Networks, An Introduction and Applications, McGraw-Hill Book Co., 1965.

BUTT, E. — The PAX Picture Processing System, Technical Report, 67,65p, Computer Science Center, University of Maryland, 1968.

CHAITIN, G.J. — Randomness and Mathematical Proof, Scientific American Magazine, May 1975.

CHILDS, D.L. — Feasibility of a Set Theoretic Data Structure, Conference Report, IFIP, 1968.

CHOMSKY, N. — Syntactic Structures, Morton & Co. Ltd., The Hague, 1957.

CHOMSKY, N. — Aspects of the Theory of Syntax, The M.I.T. Press, 1965.

CHORLEY, R.J.) Editors HAGGETT, P.) — Models in Geography, Methuen & Co. Ltd., 1967.

CLIFF, A.D. ORD, J.K. — Spatial Auto Correlation, Pion Ltd., London, 1973.

COHEN, P.M. — Linear Equations, Routledge & Kegan Paul Ltd., London, 1958.

Solid Geometry, Routledge & Kegan Paul Ltd., London, 1961.

COLIN, A.J.T. — Introduction to Operating Systems, Macdonald, London, 1971.

COMBA, P.G. — A Language for Three-Dimensional Geometry, I.B.M. Systems Journal, Vol. 7, Nos. 3 and 4, 1968.

COOK, B.G. — A Computer Representation of Plane Region Boundaries, Australian Computer Journal, Vol. 1, 1967.

COX, D.R. MILLER, H.D. — The Theory of Stochastic Processes, Chapman and Hall, 1965.

COXETER, H.S.M. — Regular Polytopes, Collier Macmillan Ltd., London, 1963.

CUTTLE, G. ROBINSON, P.B. — Executive Programs and Operating Systems, Macdonald, London, 1970.

DANTZIG, G.B. — Linear Programming and Extensions, Princeton University Press, 1963.

DAVIES, W.K.D. — The Conceptual Revolution in Geography, University of London Press Ltd., 1972.

DAVIS, J.C. McCULLAGH, M.J. — Display and Analysis of Spatial Data, N.A.T.O., Advanced Study Institute, John Wiley & Sons, 197

DENNING, P.J. — Virtual Memory, Computing Surveys, Vol. 2, No. 3 1970.

DEUTCH, K.W. — The Nerves of Government, Models of Political Communication and Control, Collier-Macmillan Ltd London, 1963.

Department of Environment — Computer Program Descriptions for Architects, Directorate of Research Requirements, Sept. 1972

DODD, G.G. — Elements of Data Management Systems, Computing Surveys, Vol. No. 2, June 1969.

GARDNER, M. — Mathematical Games, Scientific American Magazine, April 1975.

GREENBERG, D.P. — Computer Graphics in Architecture, Scientific American Magazine, May 1974.

Edinburgh Corporation — Review of the Functions and Methods of the Corporation, Edinburgh, 1970.

EVERITT, B. — Cluster Analysis, SSRC Review, Heinemann Educational Books, London, 1974.

FAIRMAN, M.) NIEVERGELT, J.) Editors — Pertinent Concepts in Computer Graphics, Proceedings of the Second University of Illinoi Conference on Computer Graphics, 1969.

FEATHERS, N. — Mass, Length, Time, Edinburgh University Press, 1969.

FELDMAN, J.A. — Aspects of Associative Processing, MIT Lincoln Laboratory, Technical Note 1965/13, 21st April 1965.

FELT, A.G. — The Community Land Use Game, Description and Manual produced by Systems Gaming Associates, 1966.

FETTER, W.A. — Computer Graphics in Communication, McGraw Hill Engineering Graphics Monograph, 1964.

FORREST, A.R. — Analytic Curves for Describing Engineering Shapes, Computer-Aided Design Group, Cambridge, 1968.

FOSTER, J.M. — List Processing, Macdonald, London 1967.

Automatic Syntactic Analysis, Macdonald, London 1970.

FOX, L.
MAYES, D.F. — Computing Methods for Scientists and Engineers, Monographs on Numerical Analysis, Clarendon Press, Oxford, 1968.

FREEMAN, H. — Computer Processing of Line Drawing Images, Computer Surveys, Vol. 6, 1974.

FREGE, G. — The Basic Laws of Arithmetic, Translated and Edited with an Introduction by Montgomery Firth University of California Press , 1964.

FRIEDMAN, Y. — A 'Non-Paternalistic' Attitude in using Models of Social Organisations, Springer-Verlag, Berli 1973.

GARDINI, E.J. — Total Communications, Switching Centre Applications, Final Article on Two-Way Information Systems, Wireless World, 1973.

GERO, J.S. — Computers in Architectural Science, University of Sydney, 1970.

GRAY, C. — Fundamental Concepts in Computer-Aided Architecture (1) Storage and Data Structure, Working Paper 49, Land Use and Built-Form Studies, Cambridge, 1970.

GRIES, D. — Computer Construction for Digital Computers John Wiley & Sons Inc. 1971.

GUZMAN, A. — Computer Recognition of Three-Dimensional Objects in a Visual Scene, Thesis, Dec. 1968. M.I.T.

HAGERSTRAND, T. — The Computer and the Geographer, The Institute of British Geographers, Publication No. 42, 1967.

HALAS, J. — Computer Animation, Computer-Aided Design, Spring 1971.

HALMOS, P.R. — Finite Dimensional Vector Spaces, Van Nostrand Reinhold, London, 1958.

Naive Set Theory, Van Nostrand Reinhold, London 1960.

HANSON, J.L. — A Textbook of Economics, Macdonald & Evans Ltd. 1961.

HARBAUGH, J.W.
MERRIAM, D.F. — Computer Applications in Strategraphic Analysis John Wiley & Sons, London, 1968.

HARVEY, D. — Explanation in Geography, Edward Arnold Ltd., London, 1969.

HARRINGTON, J.J. — Operations Research, M.I.T. Press, 1966.

HAWKES, N. — International Seminar on Trends in Mathematical Modelling, Springer-Verlag, Berlin, 1973.

HENLEY, J.P. — Computer-Based Library and Information Systems, Macdonald, London, 1970.

HERMES, H. — Introduction to Mathematical Logic, Trans. from German, Springer-Verlag, Berlin, 1973.

HIGMAN, B. — A Comparative Study of Programming Languages Macdonald, London, 1967.

HILL, F.J.
PETERSON, G.R. — Introduction to Switching Theory and Logical Design, John Wiley & Sons, Inc., London 1968.

HITTINGER, W.C. — Metal Oxide Semi-Conductor Technology, Scientific American Magazine, August 1973.

Department of Environment — An Information System for the Construction Industry, H.M.S.O., London, 1971.

HOFFMAN, L. — Computers and Privacy, Computing Surveys, Vol. 1, No. 2, June 1969.

HOGG, R.V.
CRAIG, A.T. — Introduction to Mathematical Statistics, MacMillan Co. Ltd., 1965.

HOLLINGDALE,
TOOTHILL, G.C. — Electronic Computers, Penguin Books, 1965.

HOLT, A.W. — General Purpose Programming Systems, Communications of A.C.M., Vol. 1, 1958.

Program Organisation and Record-Keeping for Dynamic Storage Allocation, Communications of A.C.M., Vol. 4, Oct. 1961.

HOLT, A.W.
TURANSKI, W.J. — Man to Machine Communication and Automatic Code Translation, Proceedings of the W.J.C.C., 1960.

HOOVER, — The Location of Economic Activity, McGraw-Hill Book Co., 1948.

HOPCROFT, J.E.
ULLMAN, J.D. — Formal Languages and their Relation to Automata, Addison-Wesley Publishing Co., 1969.

HOPGOOD, F.R.A. — Compiling Techniques, Macdonald, 1969.

HOWARTH, R.J. — The Pattern Recognition Problem in Applied Geochemistry, Extracts from Geochemical Exploration, 1972.

HUDSON, J.C. — Pattern Recognition in Empirical Map Analysis Journal of Regional Science, Vol. 9, No. 2, August 1969.

HUNT, P.
WILLIAMS, C.M.
DONALDSON, G. — Basic Business Finance, Richard D. Irwin Inc., 1958.

HYNDMAN, D.E. — Analog and Hybrid Computing, Pergamon Press, 1970.

ILIFFE, J.K. — Basic Machine Principles, MacDonald, 1968.

I.B.M. — Interactive Graphics in Data Processing, I.B.M. Systems Journal, Vol. 7, Nos. 3 and 4, 1968.

ISARD, W. — Location and Space Economy, M.I.T. Press, 1956.

Methods of Regional Analysis, M.I.T Press, 1960.

General Theory, M.I.T. Press, 1969.

IVERSON, K.E. — A Programming Language, John Wiley & Sons, Inc. 1962.

JACKSON, W.E. — Local Government in England and Wales, Pelican Book, 1961.

JACOBSEN, J.D. — Geometrical Relationships for Retrieval of Geographic Information, I.B.M. Systems Journal, Vol. 7, Nos. 3 and 4, 1968.

JAMMER, M. — Concepts of Space, Harvard University Press, 1969.

JENKINS, W.M. — Matrix and Digital Computer Methods in Structural Analysis, McGraw-Hill, London, 1969.

JENNER, R.A. — An Information Version of Pure Competition, 1966 Collected in Economics of Information and Knowledge, D.M. Lamberton, Ed. Penguin Books, 1971.

JOHANSON, G. — Visual Motion Perception, Scientific American Magazine, June 1975.

JONES, C.B. — A New Approach to the Hidden Line Problem, Computer Journal, Vol.14, Part 3, August 1971.

JUDD, D.R. — Use of Files, MacDonald, 1973.

KAYE, D. — Boolean Systems, Longmans, 1968.

KELLAWAY, G.P. — Map Projections, Methuen & Co. Ltd., 1946.

KERNIGHAN, B.
PLAUGER, P.J. — The Elements of Programming Style, McGraw-Hill Book Co., 1974.

KILBRIDGE, M.
O'BLOCK, R.
TEPLITZ, P. — A Conceptual Framework for Urban Planning Models, Urban Analysis Project, Harvard Univ., Boston, Jan. 1968.

KNUTH, D.E. — The Art of Computer Programming -
Vol. 1. Fundamental Algorithms, 1968,
Vol. 2. Semi-Numerical Algorithms, 1969,
Vol. 3. Sorting and Searching, 1973,
Addison Wesley.

KOHAVI, Z. — Switching and Finite Automata Theory, McGraw-Hill Book Co., 1970.

LAMBERTON, D.M. — Economics of Information and Knowledge, Penguin Books, 1971.

LANG, T. — Computer Programs for Mapping, Experimental Cartography Unit, Royal College of Art, London, 1970.

LECHNER, B.J. — Liquid Crystal Displays, Proceedings of the Second University of Illinois Conference on Computer Graphics, 1969.

LOSCH, A. — The Economics of Location, Yale University Press, 1954.

LOWRY, I.S. — A short Course in Model Design, Paper prepared for Publication in the Journal of the American Institute of Planners, Vol. XXXI, No. 2, 1965.

LYONS, J. — Introduction to Theoretical Linguistics, Cambridge University Press, London, 1968.

New Horizons in Linguistics, Penguin Books, 1970

MARCH, L.
STEADMAN, P. — The Geometry of the Environment, RIBA Publications Ltd., 1971.

MAXWELL, E.A. — General Homogeneous Coordinates in Space of Three Dimensions, Cambridge University Press, 1951.

The Methods of Plane Projective Geometry based on the use of General Homogeneous Coordinates, Cambridge University Press, 1946.

McCLUSKEY, E.J. — Minimization of Boolean Functions, Bell Systems Technical Journal, Vol. 35, No. 5, Nov. 1956.

MELTZER, B. — Artificial Intelligence, University of Edinburgh Bulletin, Vol. 11, No. 11, 1975.

METELLI, F. — The Perception of Transparency, Scientific American Magazine, April 1974.

MINSKY, M.
PAPERT, S. — Perceptrons, An Introduction to Computational Geometry, The M.I.T. Press, 1969.

MINSKY, M. — Computation: Finite and Infinite Machines, Prentice Hall, 1967.

MORRILL, R.L. — The Spatial Organisation of Society, Duxbury Press, 1974.

NAKE, F.)
ROSENFELD, A.) Editors — Graphic Languages, I.F.I.P. Working Conference on Graphic Languages, Vancouver, Canada, 1972, North Holland Publishing Co.

National Economic Development Office — Urban Models in Shopping Studies, August 1970.

NERING, E.D. — Linear Algebra and Matrix Theory, John Wiley & Sons Ltd., 1963.

NEWMAN, W.M. — An Experimental Display Programming Language for the PDP 10 Computer, University of Utah, July 1970, UTEC-CSc-70-104.

NEWMAN, W.M.
SPROULL, R.F. — The Principles of Interactive Graphics, McGraw-Hill Book Co. 1973.

NIEDERCORN, J.H.
BECHDOLT, B.V.
An Economic Derivation of the 'Gravity Law' of Spatial Interaction, Journal of Regional Science, Vol. 9, No. 2, August 1969.

OETTINGER, A.G.
MARKS, S.
Educational Technology, New Myths and Old Realities, Reprint from The Harvard Educational Review, 38, 4 (Fall, 1968).

OPPENHEIM, A.V.
SCHAPER, R.W.
STOCKHAM, T.G.
Non-Linear Filtering of Multiplied and Convolved Signals, Reprint from Proceedings of the IEEE, Vol. 56, No. 8, August 1968.

OWEN, K.
Computer Services - A Special Report, Times Newspaper (London), Sept. 17, 1975.

PAGE, S.
Decision Making in Management, Paper to Town Planning Institute (Scottish Branch), Jan. 1971.

PHONG, B.T.
Illumination for Computer Generated Pictures, Communications of A.C.M., Vol. 18, No. 6, June 1975.

PFALTZ, J.L.
MILGRAM, D.L.
An Experimental Map Description System, University of Maryland, Computer Science Centre, Technical Report No. 70 130, Sept. 1970.

PFALTZ, J.L.
Representation of Geographic Surfaces within a Computer, NATO Advanced Study Institute, John Wiley & Sons, London, 1975.

PIAGET, J.
Biology and Knowledge, Edinburgh Univ. Press, 1971.

Planning Research Unit (Edinburgh University)
Threshold Analysis: Optimisation, Conference Proceedings, Planning Research Unit, Edinburgh 1970.

PLAITNER, S.
Rural Market Networks, Scientific Americal Magazine, May 1975.

POOCH, U.W.
NIEDER, A.
A Survey of Indexing Techniques for Sparse Matrices, Computing Surveys, Vol. 5, 1973.

POPPER, K.R.
Objective Knowledge, An Evolutionary Approach, Clarendon Press, London, 1972.

QUINE, W.U.
The Problem of Simplifying Truth Functions, American Mathematics Monthly, Vol, 58, No. 8, 1952.

RAIFFA, H.
Decision Analysis, Introductory Lectures on Choice under Uncertainty, Addison Wesley, 1968.

ROBERTS, Margaret
An Introduction to Town Planning Techniques, Hutchinson Educational Ltd., London 1974.

ROBINSON, A.H.
SALE, R.D.
Elements of Cartography, John Wiley & Sons Inc., London, 1953.

ROGERS, P. P. — Random Methods for Non-Convex Programming Ph.D. Thesis "Division of Engineering and Physics", Harvard University, Cambridge, Mass., June 1966.

A Systems Analysis Model for Regional Water Resource Planning, Conference Paper, Harvard University Center for Population Studies, Sept., 1969.

ROMMEL, R.J. — Understanding Factor Analysis, Conflict Resolution Vol. XI, No. 4.

ROMNEY, G.W. — Computer Assisted Assembly and Rendering of Solids, Advanced Research Project Agency, Order No. 829, 1970.

ROSEN, S. — Programming Systems and Languages, McGraw-Hill Book Co., 1967.

Electronic Computers, A Historical Survey Computing Surveys, Vol. 1, No. 1, March 1969.

ROSOVE, P.E. (Editor) — Developing Computer-Based Information Systems, John Wiley & Sons Inc., London, 1968.

ROUGELOT, R.S. — The General Electric Computed Colour T.V. Displa Proceedings of the Second University of Illinois Conference on Computer Graphics, 1969.

RUSHTON, W.A. — Visual Pigments and Colour Blindness, Scientific American Magazine, March 1975.

RUSSELL, B. — An Outline of Philosophy, George Allen & Unwin Ltd., London, 1956.

RUTHERFORD, D.E. — Introduction to Lattice Theory, Oliver & Boyd, Ltd., 1965.

SALTON, G. — Automatic Information Organisation and Retrieval McGraw-Hill Book Co., 1968.

Scientific American (Readings from) — Computers and Computation.

Mathematical Thinking in Behavioral Sciences

Information, Sept. 1969.

SHANNON, C.E. WEAVER, W. — The Mathematical Theory of Communication, University of Illinois Press, 1949.

SKYRME, D.J. — Computer Graphics, Part 1, Choosing a System, Computer-Aided Design, Spring 1971.

SHARPE, W.F. — The Economics of Computers, Columbia Paperback Columbia University Press, London, 1969.

SHEPARD, D. — A Two-Dimensional Interpolation Function for Computer Mapping of Irregularly Spaced Data, Harvard Papers on Theoretical Geography, March, 1968.

SPRUNT, B.F. — Relief Representation in Automated Cartography: An Algorithmic Approach, NATO Advanced Study Institute, John Wiley & Sons, 1975.

STEINITZ, C.
SINTON, D. — GRID, A User's Reference Manual, Laboratory for Computer Graphics and Spatial Analysis, 1968.

STOCKHAM, T.G.
BRITTON, R.L. — Application of Digital Signal Filtering to Image Processing, Nerem Record, 1969.

SUTHERLAND, I.E. — Computer Displays, Scientific American Magazine June 1970.

SUTHERLAND, I.E.
SPROULL, R.F.
SCHUMACKER, R.A. — A Characterization of Ten Hidden-Surface Algorithms, Computing Surveys, Vol. 6, No. 1, March 1974.

SWITZER, P. — Estimation of the Accuracy of Qualitative Maps, NATO Institute of Advanced Study, John Wiley & Sons, 1973.

TANAKA, K. — The Orthographical Relief Method of Representin Hill Features on a Topographical Map, Geographical Journal, Vol. 79, 1932.

TEUBER, Marianne L. — Sources of Ambiguity in the Prints of Maurits C. Escher, Scientific American Magazine, July 1974.

TIEN, P.K. — Integrated Optics, Scientific American Magazine, April 1974.

TOMLINSON, R.F. (Editor) — Environmental Information Systems, UNESCO/IGU Symposium Proceedings, Ottowa, Sept. 1970.

Do. — Geographical Data Handling, Vols. 1 and 2, UNESCO/IGU Symposium Proceedings on Geographica Information Systems, Ottowa, Aug. 1972.

The Application of Electronic Computing Methods and Techniques to the Storage Compilation and Assessment of Mapped Data, Ph.D. Thesis, London University, July 1974.

U.S. Department of Housing and Urban Development — Urban and Regional Information Systems, Oct. 1968, U.S. Government Printing Office, Washington, Oct. 1968.

VACROUX, A.G. — Micro-Computers, Scientific American Magazine, May 1975.

WAGNER, H.M. — Principles of Operations Research, Prentice Hall International, Inc., London, 1972.

WARNTZ, W.
WOLDENBERG, M. — Concepts and Applications: Spatial Order Harvard Papers in Theoretical Geography, Laboratory for Computer Graphics and Special Analysis, May 1967.

WATKINS, G.S. — A Real Time Visible Surface Algorithm. Computer Science Department, University of Utah June 1970. (UTECH CSc 70 101)

WATSON, R.W. — Time-Sharing Design Concepts, McGraw-Hill Book Company, 1970.

WAUGH, T.C.
THOMAS, A.L. — Geographic Information Management and Mapping System, System Manual, June 1970.

WAUGH, T.C. — GIMMS Reference Manual, Program Library Unit, Edinburgh University, Sept. 1975.

WEGNER, P. — Programming Languages, Information Structures and Machine Organisation, McGraw-Hill, London, 1971.

WEINER, N. — Cybernetics, The Technology Press, John Wiley & Sons, New York, 1948.

WINOGRAD, T. — Understanding Natural Language, Edinburgh University Press, 1972.

WEHRLI, R.
SMITH, M.J.
SMITH, E.F. — ARCAID, The Architects' Computer Graphics Aid, Computer Science, University of Utah, June 1970, (UTECH CSc 70 102).

WILKES, M.V. — The Growth of Interest in Micro-Programming, A Literature Survey. Computing Surveys, Vol. 1, No. 3, Sept. 1969.

WILSON, A.G. — Entrope in Urban and Regional Modelling, Pion Ltd., London, 1970.

YOVITS)
JACOBI) Editors
GOLDSTEIN) — Self-Organising Systems, Conference Proceedings, Sparton Books, Washington, 1962.