CARTOGRAPHIC DISPLAY OPERATIONS ON SURFACE DATA COLLECTED IN AN IRREGULAR STRUCTURE

by

Douglas Cochrane

B.Sc., Simon Fraser University, 1973

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF ARTS

in the Department

of

Geography

© DOUGLAS COCHRANE 1974

SIMON FRASER UNIVERSITY

July 1974

All rights reserved. This thesis may not be reproduced in whole or in part, by photocopy or other means, without permission of the author.

APPROVAL

Name: Douglas Cochrane

Degree: Master of Arts

Title of Thesis: Cartographic Display Operations on Surface Data Collected in an Irregular Structure

Examining Committee:

Chairman: M.C. Kellman

Thomas K. Peucker
Senior Supervisor

Robert B. Horsfall

Theodor D. Sterling

Carl Youngmann
Assistant Professor
University of Washington
Seattle

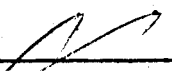
Date Approved: Oct 23 1974

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis or dissertation (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this thesis for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Title of Thesis/Dissertation:

Cartographic display operations on surface data collected in an irregular structure.



Author:

(signature)

Douglas Cochrane

(name)

November 1, 1974

(date)

ABSTRACT

Cartography is the art of displaying scaled reproductions of geographic areas or spatial phenomena. The procedure, or methodology, by which cartography is performed may be broken down into three basic steps; data collection, data storage and data display. Each step involves the manipulation of information; information describing that which the cartographer has attemped to reproduce. Information must flow through the process from the encoding of the original geographic area or spatial phenomena to the cartographic display. The process in its totality must be capable of maximum information retention, and, since the process is only as strong as its weakest step, each step must in turn be capable of maximum information retention.

The research involved in this thesis has been carried out for the purposes of maximizing both the information capacity and the efficiency of the cartographic process for computer cartography. A relatively new geographic data structure has been designed for the storage of information representative of surfaces. Specific points chosen for their ability to characterize both changes and similarities of a surface are collected and stored in triangular networks. The author's major contribution is a system of computer programs for the display of the new data structure. Three basic routines were developed; a contour plotting routine, a perspective view routine and a routine for construction of shaded relief maps. The objective of each routine was the display of all information contained by the data structure, and for a properly designed data structure this should be all the information collected from the surface. Sample runs of the system have been made and one such run is included accompanied by the programs and subroutines of the system. Sample displays of the test surface seem to imply that the data structure and display routines do indeed retain the majority of the information collected.

ACKNOWLEDGEMENTS

I wish to thank my senior supervisor, Dr. T.K.Peucker, for suggesting the topic of this thesis and for his assistance in the research. A great deal of thanks goes also to Dr. R.B.Horsfall for his encouragement throughout the research and for his welcome comments in proof reading the draft. A special thanks must go to my fellow graduate student Mr. David H. Douglas not only for the various subroutines and programs used by this system but also for his encouragement and many helpful comments throughout the entire project.

Finally I wish to express my most sincere thanks to my wife, Vickeri E. Cochrane, for her encouragement, understanding and patience.

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER I : THE CARTOGRAPHIC PROCESS

Cartography is the technique of collecting and displaying information pertaining to the spatial characteristics of geographic regions. Collection is the gathering of information, while display involves the reproduction of this information in such a way as to give the user or viewer maximum knowledge of the original region or spatial phenomena. Since geographic study areas are in general very large and the amount of collected information quite voluminous, certain storage techniques must be used to keep the information in a logical, usable form. Storage, the intermediate step to collection and display, should be based on consideration of the information being stored and the use which is to be made of the information.

However, cartography is not simply a matter of collecting, storing and displaying information - cartography is the totality of these processes and must consider each one as part of the group. Common to each of these stages is that which is of primary concern; the information! Since the objective is to reach display with as little loss of information as possible and since the efficacy of each step is dependent upon that of the previous step, it is necessary that retention be maximized throughout. The maximum amount (in terms of desired output) of relevant information must be collected, since the base of the cartographic pyramid is the collection stage (Figure 1.1) and all else is dependent upon this. The cartographic pyramid gives a good illustration of information flow. The shape of the pyramid illustrates information retention; information content generally decreases with height, or the processing distance, from the original region.

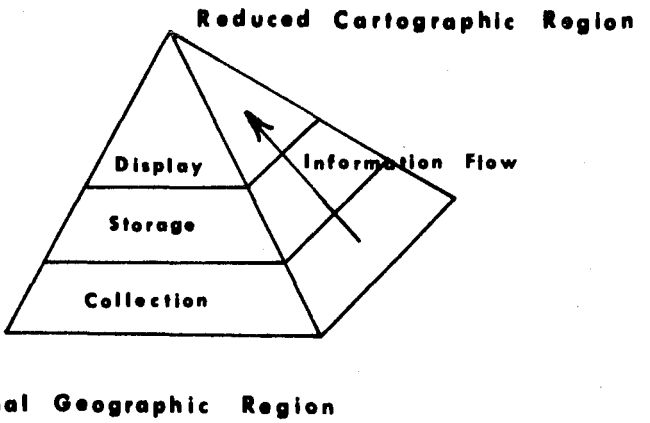


Figure 1.1 Cartographic Pyramid

The information content of cartographic maps has recently come under study by both cartographers and mathematicians. Sukhov (26), examines how the contents of a map may be generalized and, through the use of information theory, calculates the deviation of a given map from that of a fully "loaded" source, or what proportion of the maximum possible information is not on the finished map. Gokhman and Mekler (12) analyse the methodology in studying the information content and capacity of maps, deriving various formulas which may be of great use in the study of maps as the end product of the mapping process. These studies, which consider the map as an information source, are of great value in the optimization of the cartographic technique - since a good measure of the efficiency of the technique is in the analysis of the information content of the finished product (the map).

In all studies of information flow and the communication of ideas it is necessary to understand how information is measured. How much information does any given communication have or not have? If a map

contains information is this information in the best form, is the map a good medium for the transportation of information, and, how may the map be improved to maximize its information content with little or no loss of aesthetic value?

Information theory is that body of descriptive knowledge built up to describe and quantify the content of communications. A certain communication is deemed to carry information only if it alleviates or reduces the uncertainty which existed for the receiver before that particular communication took place. The true quantity of vagueness removed is difficult, if not impossible to measure. However through statistical treatment it has been possible to assign relative values to the information content and to evaluate certain modes of communication on this basis. In other words it is impossible to assign an absolute value to the amount of information contained in a given message, however the content can be compared to other messages and other modes of communication for a relative measure.

Communications may be said to relate information to the observer concerning an event (E) which occurs with a probability of P(E). If the observer is told that the event (E) has occurred then by definition he has received

$$I(E) = \log \left[\frac{1}{P(E)}\right]$$

units of information.

The base chosen for the logarithm dictates the units of information. The logarithm to base 10

$$I(E) = \log_{10} \left[\frac{1}{P(E)}\right]$$

gives information in units of Hartleys. With the natural logarithm

or log to base e

$$I(E) = \log_e \left[\frac{1}{P(E)}\right] = \ln \left[\frac{1}{P(E)}\right]$$

information is in nats. The most common, in computer usage, is the base 2 logarithm or binary unit in which the resulting unit of information

$$I(E) = \log_2 \left[\frac{1}{P(E)}\right]$$

is called the bit.

For example if the event (E) is the flipping of a coin and the subsequent observation of a head then $P(E) = \frac{1}{2}$. If the coin is flipped and a head is observed then the information received is

$$I(E) = \log_2 \left[\frac{1}{P(E)}\right] = \log_2 (2) = 1 \text{ bit}$$

or

$$I(E) = \log_{10} \left[\frac{1}{P(E)}\right] = \log_{10} (2) = 0.301 \text{ Hartleys.}$$

An information-generating mechanism is an information source in which the successively emitted symbols are statistically independent of each other and is called a zero-memory information source. Each symbol from the fixed finite source alphabet

$$S = \left[S_1, S_2, S_3, \ldots\ldots\ldots\ldots\ldots, S_n\right]$$

arrives from the mechanism with no dependence whatsoever on the preceding symbol. The source (S) is completely described by the alphabet and probabilities with which the symbols occur - which are

$$P(S_1), P(S_2), P(S_3), \ldots\ldots\ldots\ldots\ldots, P(S_n) \quad .$$

Since this is a discrete information source, with all symbols being statistically independant, the information provided by the occurrence of the symbol S_j is given by

$$I(S_j) = \log_2 \left[\frac{1}{P(S_j)} \right] \text{ bits} .$$

The average amount of information per symbol can then be calculated as

$$\sum_{i=0}^{n} P(S_i) \cdot I(S_i) \quad \text{bits}$$

Average information per symbol is defined as the entropy of the source and may be interpreted as the amount of uncertainty of the observer before the communication is received.

Information theory sets out the guidelines for the quantification of the information contained in a given communication, through the application of which, various modes of communication may be compared for the most efficient. Abramson (1, p.13) presents a comparison of the spoken word of a radio announcer to that of one single television picture. Assuming that the radio announcer has a vocabulary of 10,000 words, of which he selects 1000 in a random fashion (which is of course not the case in the process of normal speech), the probability of him choosing any one particular 1000 sequence (E) is then

$$P(E) = \frac{1}{(10,000)^{1000}}$$

And the amount of information transmitted by any particular sequence (E) will be

$$I(E) = \log_2 \frac{1}{(1/10,000^{1000})} = \log_2 (10,000)^{1000}$$

$$= 13,000 \text{ bits}$$

The television picture is composed of a matrix 500 x 600 with each cell of the matrix (black and white television) having any one of 10 distinguishable grey scales possible. Therefore the base alphabet for the television is

$$(10)^{500 \times 600} = 10^{300,000}$$

Thus the probability of any one picture (E) occurring is

$$P(E) = \frac{1}{10^{300,000}}$$

And the information transmitted by one of these pictures is

$$I(E) = \log_2 \frac{1}{(1/10^{300,000})} = \log_2 10^{300,000} \cong 10^6 \text{ bits.}$$

Therefore, comparatively speaking, the information transmitted by one television picture far exceeds the information transmitted by 1000 spoken words.

Maps, as Sukhov (26) , and Gokhman and Mekler (12) point out, may also be compared in this manner. However, since maps are a combination of various communication techniques, { i.e. text, topographic line work (isolines, hachuring), thematic line work (roads, settlements), hydrographic line work (rivers, lakes, oceans), colour patterns (tonings, shadings), and map graticules } each of which may be considered as an information-generating mechanism, information content must be determined for each of these techniques considered as an information source. The map must be broken down into its component parts and analysed piece by piece, the information for the various sources is then summed to give the total.

However the effects of various tehniques working in conjunction with each other will be lost with this method. If for example five hundred foot colour changes are overlaid with ten foot contours of a topographic

surface it is possible to calculate the information transmitted by the contours and by the colour changes, however the information of the combination is difficult if not impossible to measure. It is known that the colours will enhance the contour levels and make the relief appear more definite - but the information transmitted by this technique is not a hard and fast unit which can be measured. The map therefore contains certain tangibles which may be quantified in terms of the information they provide, but the intangibles are inherent in the map and any information which they provide will be overlooked in objective measurements even though it may be of great value.

Analysing the information content of given display techniques will afford some insight into the efficiency of the displays, however within the cartographic process, display is simply the end result. If different maps vary in information it cannot be concluded that this has been solely determined by the display technique. The cartographic pyramid, which illustrates information flow and whose shape corresponds with information retention, shows that information is not only lost at the display stage but also at the collection and storage stages. A good display can do no more than to illustrate the information that has proceeded this far in the process; it cannot generate its own information but must be fed this from the lower stages. Thus analysis of information content must take place at the collection, storage and display stages in order to optimize the cartographic process.

The collection stage should give the base information set for the study and should be based upon that which the study is intended to examine. The collection stage is therefore the most important member of the group and all necessary information should enter here. Entrance at this point should be directly into storage, with the storage technique used being developed primarily to serve the information entering. However, since the information must exit from the storage stage to the display stage the storage technique should also accommodate the exiting of the information. Thus the storage technique assumes a dual role - it must not only

accept information from the collection stage with maximum retention but also give information to the display stage, ideally, exactly as much information as was accepted from the collection stage. If the storage and display stages can be created to carry all information collected, the cartographic pyramid will assume a new shape (Figure 1.2) - since all information collected from the original geographic region will appear on the reduced cartographic version. However since the cartographer does not wish to duplicate reality - only accentuate certain chosen attributes - the collection stage will

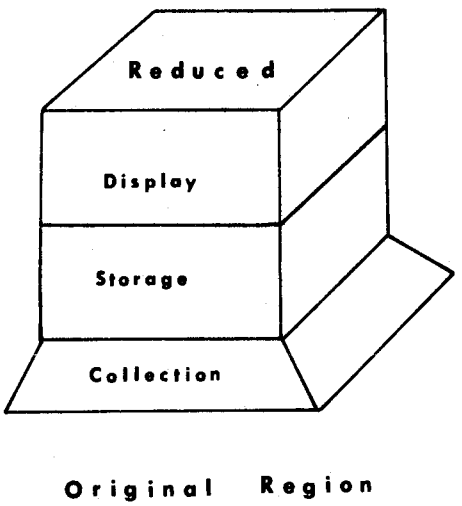


Figure 1.2 Cartographic Pyramid with information retention.

still show some loss of information. This cannot really be termed a loss, as it is a filtering or selective sampling, recording only those elements of reality which are deemed necessary to an understanding of that particular phenomena which is to be studied.

CHAPTER II: THE COLLECTION STAGE

Maps are two dimensional representations of three dimensional geographic areas or spatial phenomena. They are based on information selected from an infinite number of characteristics concerning the area in question. Since the map only represents reality in a reduced form it is imperative that the information selected from reality be characteristic of the reality. However, since no map is a representation of all of reality but only certain portions thereof, the information selected must only characterize these portions. For example, topographic maps, hydrographic maps, land use maps, settlement pattern maps or linguistic maps all represent different portions of reality and the information selected from reality must characterize the appropriate portion. Therefore, the type of map to be constructed or the focus of the information to be presented by the map, dictates what information shall be collected from the geographic area.

In the representation of surfaces (topographic, economic, or social), specific, characteristic points must be selected from the real surface. These points are functions of two variables

$$f(x,y) = z,$$

in which f is the function of surface behaviour from which z may be calculated - given x, y and the function. However, unlike mathematical surfaces, f is generally unknown for topographic surfaces. Therefore in the collection of information from a topographic surface it is necessary to collect the three values, x, y, and z, which will completely define the behaviour of the surface. Selection of these points may be carried out by any one of four methods;

1) regular grid overlay in the x, y plane, with collection of the z variable,

2) sampling varying x, y coordinates for a fixed z value with constant incremental increases in z (i.e. recording x, y coordinates of isolines),

3) collecting z values for random x, y coordinate locations,

or 4) collecting z values of specific points on the surface - where the points are determined by how characteristic they are of changes in the surface.

Any one of these methods will yield data characteristic of the surface and enable reproduction. However, since some collect more redundant information than do others, they require larger samples to achieve equality of information content.

Sampling a surface with the regular grid overlay technique involves superimposing a grid of constant x, y increments onto the surface and recording the z values for the grid nodes. In order that a statistically sound sampling be obtained, the theoretically "best" cell size may be calculated with the sampling theorem, which states that "if a function has no spectral components of frequency higher than W, then the value of the function is completely determined by a knowledge of its values placed ½ W apart. The lack of freedom of the function for variation from the prescribed path between sample points is a consequence of its lack of spectral components of frequency higher than W". (28). For example, if undulations of the surface are to be detected to within 100 units, it would be necessary that the grid cells be of 50 unit spacing or less.

Data collection based on regular spacing of the sample points should, if the sampling theorem is invoked, yield a statistically sound representation of the surface. Data collected in this manner tend to have the appearance of overlaying the surface with a "fish net", Figure 2.1, since each cell of the grid is only represented by the grid nodes.

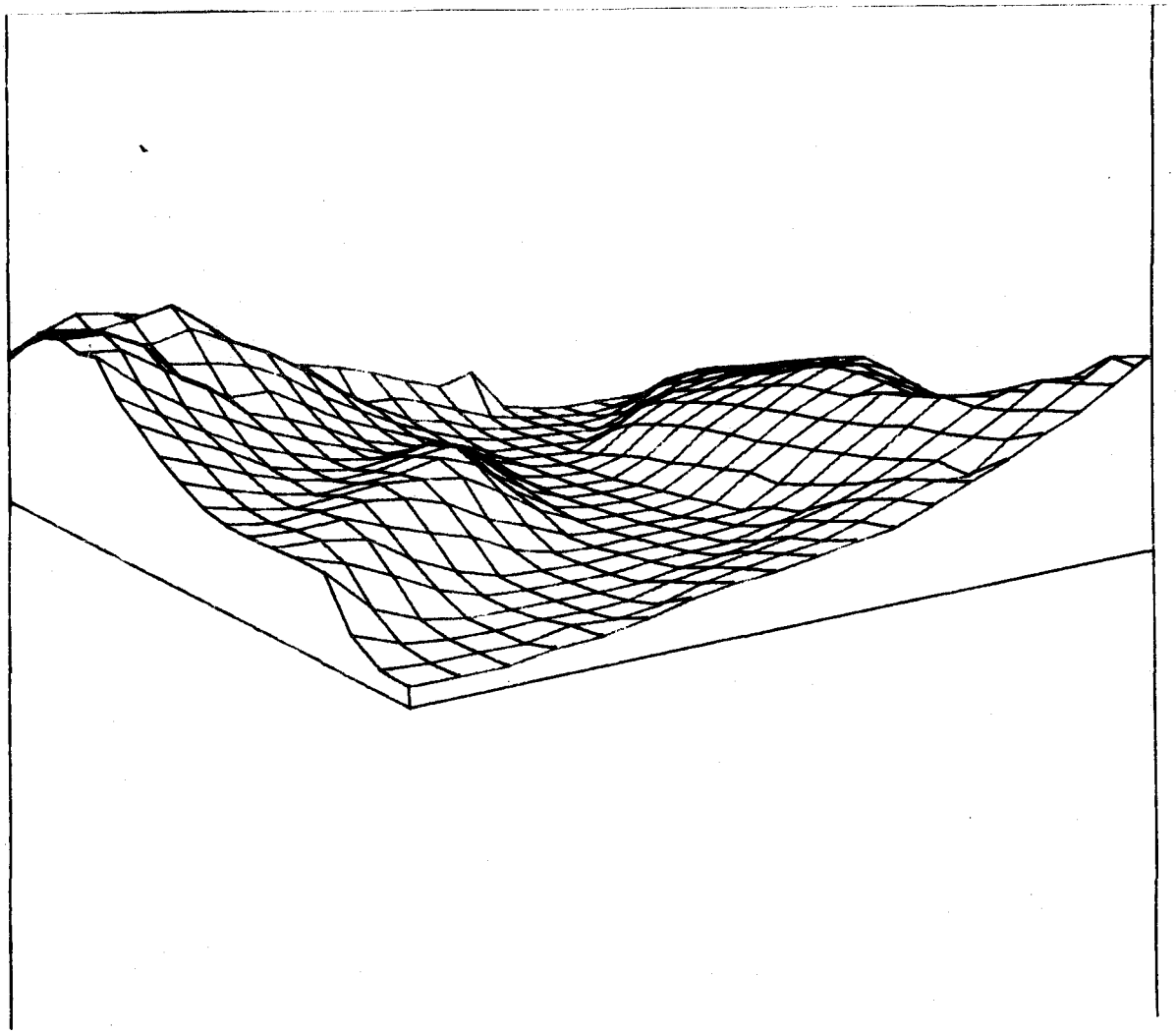


Figure 2.1 Regular Grid Data Collection. Program: VUBLOK; Production: D. Cochrane

Information collected in this manner is somewhat random, since the collection points (nodes of data collection) are not chosen for the surface in question but for the reproduction desired by the user. It is unlikely, unless the cell size chosen is extremely small, that the nodes will happen to correspond with breaks of the surface, or points which define changes in the surface. If the cell size is made small enough to ensure collecting all important points of the surface, the amount of data thus collected will be enormous, the majority being redundant. For example, if a data collection area is comprised of mountainous regions and plateau regions interspersed, the cell size will have to be extremely small to sufficiently cover the mountainous regions ensuring adequate data collection points. However, the plateaus will be represented by more than an adequate number of data points, each representing nearly the same surface function and many of which may be considered redundant. Collection of data by this method not only wastes a good deal of time in the collection stage, but since the data is carried through and displayed, waste is perpetuated throughout the process.

Sampling a surface by recording only the x, y coordinates of isolines, or, allowing the x, y values to vary while holding the z value fixed is a technique of prime importance to photogrameters. Through the use of optics, the photographer constructs, from aerial photographs, a three dimensional visual representation of the surface, from which he then records the x, y coordinates of the surface by a form of tracing. Sampling a surface by this technique involves recording the x, y coordinates of major breaks along contour lines, the interval at which data points are recorded is dependant upon the regularity of the surface. Smooth contours may be approximated by few data points, while rough and irregular contours will need more. However, as pointed out by Douglas and Peucker (11), attempts to automate this technique have lead to vast amounts of redundant information being recorded, since the machine does not intuitively "choose" points which represent major breaks of the surface. Automatic line followers record data at regular intervals

of either distance or time, and the sampling is therefore random since points are not chosen to represent the surface, only the output desired. Of course, the theoretically "best" interval may be chosen with the sampling theorem but this does not alleviate the problem of vast amounts of redundant information being recorded.

Redundant information may be screened from the raw data. Douglas and Peucker develop a technique which seems to relieve the data of only its redundant parts. Figure 2.2 shows the comparison between raw data and screened data - line a is constructed from 112 data points, while line b is constructed from only 19 data points.

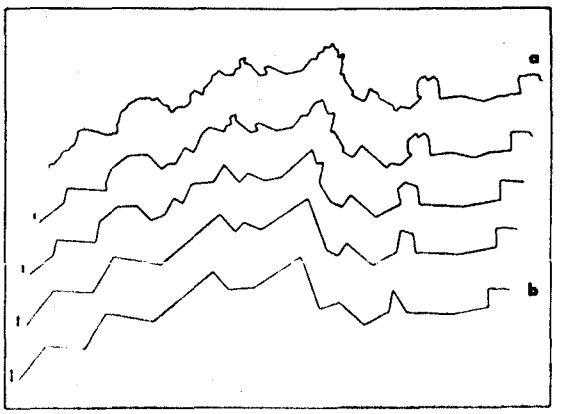


Figure 2.2 Line Reduction. Douglas and Peucker (11)

Information content of the message (line) has been retained, only the redundancy has been removed. Of course line a or line b is only a generalized representation of the real line since specific breaks have not necessarily been recorded. However, this technique of choosing a vast number of data points and subsequently screening out only the redundant information will approximate methods of recording only points of high information content. As the number of data points chosen increases, so does the probability of choosing points of high information content -

points representing changes in the surface function.

A surface is a continuum of points each of which is characteristic of its neighbourhood. A neighbourhood may be defined mathematically as "for any real number c, a neighbourhood of c is that open interval containing c". (14 , p. 74) The size of the interval of which the data point is characteristic is of course dependant on the nature of the surface or the surface function. For example, the neighbourhood of which a point on an infinite level plane is characteristic is an infinite neighbourhood, whereas if the surface is extremely irregular the neighbourhood of characterisation becomes smaller. The size of the neighbourhood will vary in relation to the irregularity of the surface in a given distance. The autocorrelation function shows that the larger a neighbourhood becomes, the more probable it is that any point chosen at random from the real surface will vary from the data point for which the neighbourhood is being drawn.

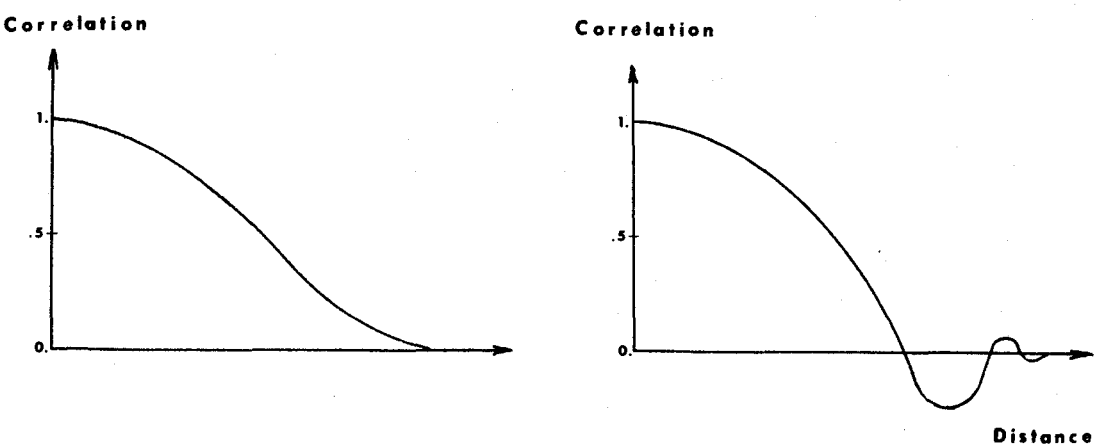


Figure 2.3 Autocorrelation Function

Figure 2.3 illustrates the autocorrelation function - correlation of the surface at any given data point is plotted against distance from the data point in question. The correlation between the surface and the data point decreases as the distance increases, maximum correlation is attained at the data point where the distance is equal to zero.

Neighbourhoods, or regions of homogeneous information, are comprised of characteristically similar data but similar data carries redundant information, unnecessary to the description of the surface. Changes in neighbourhood however are of high information content, as in the statement "the wrong side of the tracks"; the tracks are an important socio-economic boundary relaying a great deal of information to the observer. A line or boundary has a certain extension or neighbourhood within which a point may lie and still be descriptive of the change between neighbourhoods. If a point is chosen at random from a surface then the probability that this point will lie within a neighbourhood is proportional to the area of the neighbourhood, and the probability that the point will lie on a neighbourhood boundary is proportional to the "area" of the boundary. Thus if the probability of choosing a point within a neighbourhood (WN) is equal to P(WN) and the probability of choosing a point on a neighbourhood boundary (NB) is equal to P(NB) then

$$P(WN) > P(NB) \qquad 1$$

since the area (WN) will generally exceed the "area" (NB). The information content of the points so chosen is

$$I(WN) = \log \left[\frac{1}{P(WN)} \right]$$

$$I(NB) = \log \left[\frac{1}{P(NB)} \right]$$

from 1

$$\left[\frac{1}{P(WN)}\right] < \left[\frac{1}{P(NB)}\right]$$

and

$$\log\left[\frac{1}{P(WN)}\right] < \log\left[\frac{1}{P(NB)}\right]$$

which implies directly that

$$I(WN) < I(NB)$$

The information provided by a point chosen from within a neighbourhood will be less than that provided by a point chosen on a neighbourhood boundary - assuming of course that 1 is true. Thus, if the neighbourhood of a point is known, and it should be, simply from the regularity of the surface, sampling of the surface for the retention of high information content need only consider the characteristics of points which represent changes of neighbourhood. This of course is the method of the surveyor who records only breaks of the surface, which he has approximated with polygons (neighbourhoods).

Points characteristic of changes in neighbourhoods have been called "surface specific" points by Warntz (29). He outlines three distinct groupings of points and lines; peaks and pits, passes and pales, ridge lines and course lines. With this set all high level changes in surface behaviour (neighbourhood changes) may be noted. A peak is a point of local maximum and a pit a point of local minimum ; both are, within some relative neighbourhood, surrounded by closed contours. A pass and a pale are points of inflection in the surface profile and have no closed contours in their neighbourhoods. A pass occurs at the junction between ridge lines and course lines, while a pale occurs at the high point between pits, and is itself a low point of the ridge between the pits. A ridge is the line of meeting of two upwardly sloping surfaces and conversely a course is a line of meeting of two downwardly sloping

surfaces. Ridge lines, which connect points of local maxima, are separated by course lines, which connect points of local minima.

Surface specific points are points of high information content, not only as changes in neighbourhoods but also as specific points within groupings of neighbourhoods. The knowledge that a point is a peak contains the information that this point is a local maximum, the first derivative of the surface function is zero at this point, and this point represents a change of surface characteristics for at least two neighbourhoods (usually more). Entire groups of neighbourhoods can be reconstructed from the surface specific points, since not only do these points designate changes in neighbourhoods, they are characteristic of the neighbourhoods of which they form a part. These points define the spatial relationships of neighbourhoods through their relative positions on the surface (21, p.29).

Sampling only surface specific points is not a new technique - surveyors have been collecting surface detail in this manner for centuries. However, within the field of computer cartography, even though data have been collected in this manner it is normally interpolated to a regular grid for storage. Symap, one of the forerunners in computer cartographic processing, accepts as input irregularly collected data points but transforms the data onto a regular grid for any subsequent operations. Interpolation disperses the information content of a data point to the grid nodes. Information regarding the true, surface relationship of a point, which is characteristic of a neighbourhood, to other neighbouring specific points is lost. All information which is implicit to a point's position, and, all information to be derived from a point's interaction with its neighbouring specific points - also implicit by position - is gone. Certainly a new type of implicitness is gained, that of the position of the grid nodes, but this is for the interpolated surface, not the true geographic surface for which the data was collected. Surveyors, on the other hand, utilize this information. Once data concerning the surface specific points has

been collected the surveyor plots the points and connects them (even if only mentally) into a rough triangular network. From this network he then begins to draw his conclusions about the surface, whether it be for subdivisions, engineering structures (roads, bridges, buildings), or simply to map the surface - his decisions are always based on the surface specific points and their neighbourhoods. Once such data has been collected it is imperative that every detail be retained; information content should not suffer simply to add to the ease of collection or handling.

CHAPTER III: THE STORAGE STAGE

Computer representation and manipulation of surfaces involves processing vast amounts of data, which, if left in a raw unstructured form would be generally inaccessible. Data sets are comprised of "elementary data items" (10, p.118) which are singular pieces of information (i.e.x-coordinate of data point). A collection of all elementary data items concerning one data point is termed a record, and the collection of records into logical units is a file. "The arrangement and interrelation of records in a file form a data structure" (10, p.119) . The information to be retrieved and the type of processing to be carried out dictates the type of data structure best suited to a given problem. As Dodd (10) points out there are many types of data structures and many combinations thereof.

As an example of a data structuring problem, consider all the names and phone numbers of the greater Vancouver region recorded as one name, address and phone number (elementary data items) per data card (record). If these cards were then shuffled randomly they would constitute a very raw data set, and, there would exist many possible structurings.

a). alphabetize the records by surname,
b). alphabetize the records by given names,
c). sequence the records by phone numbers,
d). regionalize on the basis of city blocks and order by house number,
or e). sort by street then order by house number.

Any of these methods would yield a data structure, however the speed of computer access to the elementary data items would vary for each.

The alphabetized structure of (a) would be very efficient for retrieval of phone numbers if given the surname, but, the structure of (e) would only impede the same search.

The functions of a data structure are numerous and depend entirely on the problem to be solved. Surface storage requires data structures which are capable of a). minimizing space requirements
b). allowing fast access for retrieval, updating, and storage
and c). retention of maximum spatial information content.

However, simply attaining each of these does not guarantee a good data structure - these characteristics must compensate and complement each other, attaining the best from each in the totality.

As mentioned previously, retention of maximum spatial information content is of prime importance, since this is the purpose of the data structure in surface manipulation. There would be absolutely no benefit in creating a fast, small data structure containing relatively little information.

Surface representation is best accomplished by the collection of irregular data points, surface specific points. These points carry information describing the spatial relationships of certain surface features and this information must be retained by the data structure. Any data structure designed to accommodate this information must have the following characteristics.

Neighbourhood relationships should remain with the data, to the point of making them explicit by actually recording within the data structure the fact that two or more certain neighbours are adjacent and have a given relationship between them. Pointers between neighbours such as, point A is a neighbour of point B which is in turn a neighbour of point C, should be established. The surface specific points should be related

to each other within the structure, since a good deal of information is herein contained.

Linked or list organization is a type of data structure in which the links "are used to divorce the logical organization from the physical organization" (10, p.122) of the structure. The links, or as they are more generally called pointers, connect logically adjacent parts of the data set which are stored in physically non-adjacent areas of storage, thus allowing the computer to process a record which is in a physically different area of storage than that adjacent to the preceding record. The pointers between records may be part of the actual record or they may simply be computer addresses calculated at the time of processing (hash coding technique). For example in Figure 3.1 if the data were to be organized alphabetically it would be structured and processed as it is in the table.

Label	Name	Address	Phone #	Pointer
21	Anderson, Q. J.	1715 W. Hastings St.	397-0421	37
37	Bambi, D.R.	307 Cordova St.	472-9152	67
12	Doe, J.	1219 E. South St.	281-0415	21
67	Smith, E.J.	2102 S. West St.	972-0916	999
03	Vanline, W.K.	1605 N. Smith St.	104-3721	12

Figure 3.1 List Processing Records

But if it was desired to process the names sequentially by phone number from lowest to highest it would not be necessary to even check the phone numbers, processing would start at Vanline and proceed as shown in Figure 3.2

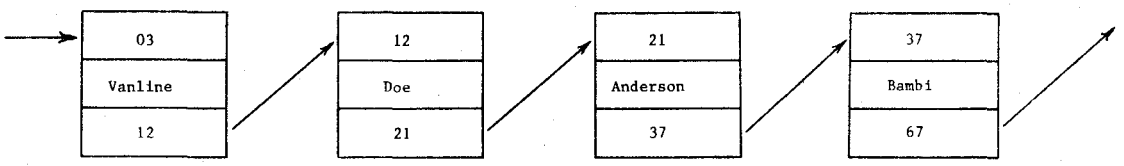


Figure 3.2 Data Flow, List Processing

Vanline's pointer is 12 which means the next record to be processed has a label[1] of 12 and is Doe, which has a pointer of 21 which points to Anderson, etc. This example demonstrates how the list organization functions, with one label and one pointer per record. The phone numbers become logically adjacent (processing will be in the logical order) even though there may be a vast physical distance between the records.

---

[1] label = key

A record may of course be logically adjacent to more than one other record, since the adjacency premise may differ - i.e. phone number, street address, or Christian name. For example the pointers established in Figure 3.1 have an adjacency premise of phone numbers, the phone numbers increase as the pointer list increases. However, if the premise were to be surnames, the order of processing would have to be

21, 37, 12, 67, 03

which would indicate a different pointer for each record. Obviously, if the premise were addresses another group of pointers would be created. Thus each record may have more than one pointer, with each pointer being dependent upon a different adjacency premise. In a record's pointer list the first pointer may point to the adjacent record based on phone numbers, the second to the adjacent based on street address and the third based on Christian name. Of course each record may have only one label since the record is processed by its label and multiple labels would only increase processing time in checking the label list for the given label. Data, representing surface structures, will have multiple pointers, each surface specific point must be connected to more than one other surface specific point. A surface cannot be represented by points which are functions of, or connected to, single other points.

The data structure chosen in this research was one in which multiple pointers were used to connect surface specific points. The method of the surveyor was followed; data were collected irregularly for the breaks of the surface, or the boundaries of neighbourhoods (Appendix II, page 85). The irregular data were then triangulated with the criteria for triangle formation being that the plane of the triangle most closely approximate the real surface. The criterion for closeness was that the triangle chosen (many triangulation schemes are possible from a given number of points (23 ,Chapter 4)) should have the least absolute distance between the plane of the triangle and the real

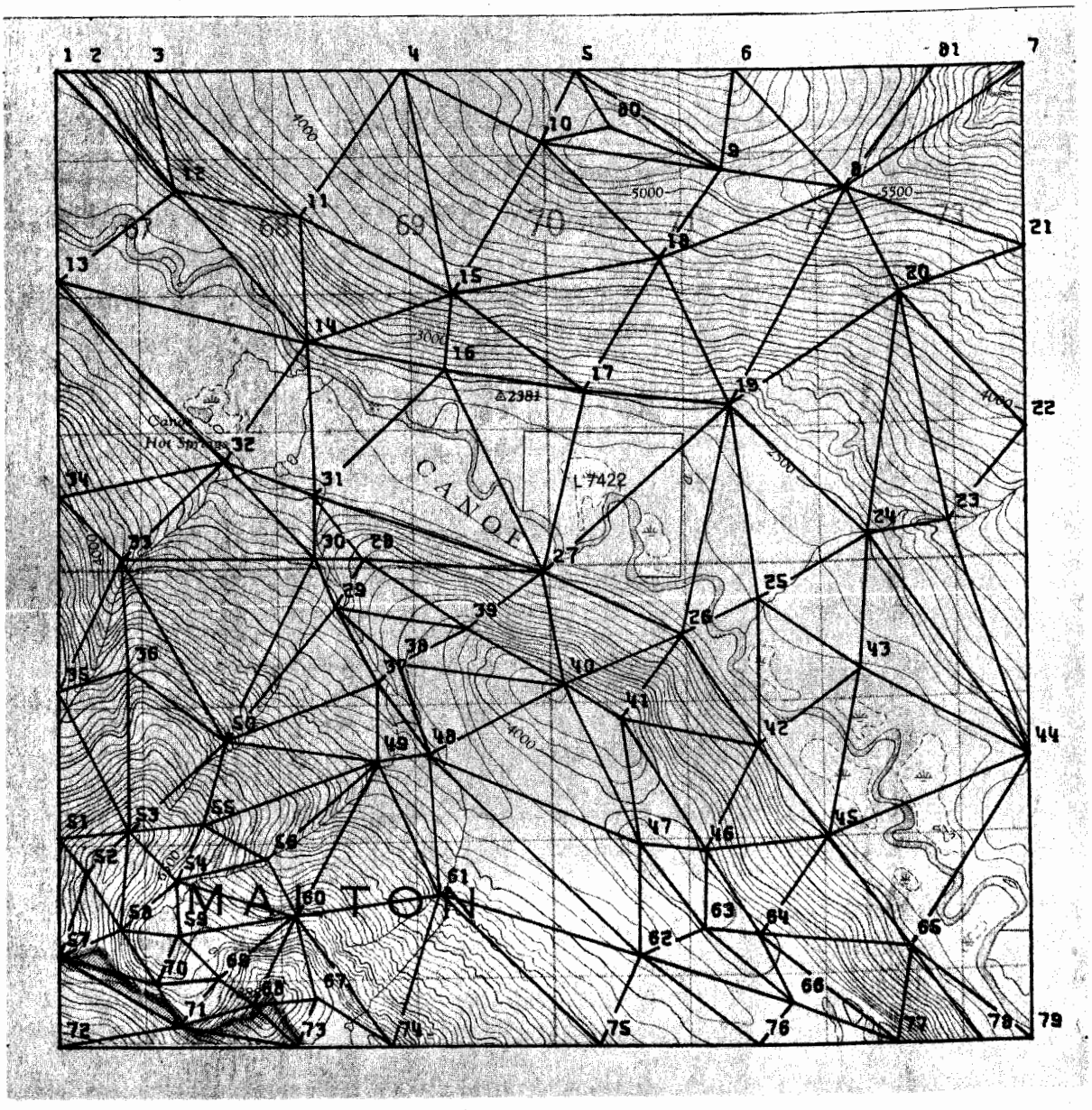


Figure 3.3 Triangulation Scheme, Canoe River Valley

surface. Since the data points for this structure were derived from a map it was relatively simple to choose the plane of least deviation from the surface[2]. Figure 3.3 illustrates the triangulated data points overlaid on the original surface.

The data structure is constructed from the triangulated network at the time of recording the data values. Recording is done according to the following rules:

1) sequentially (starting at 1) label all data points (nodes of the network) up to N, the number of points in the data set.

The following are done for each data point in order, one at a time (Format is outlined in Appendix II).

2) record the label (from above) and the x, y, z coordinate of the data point in question.

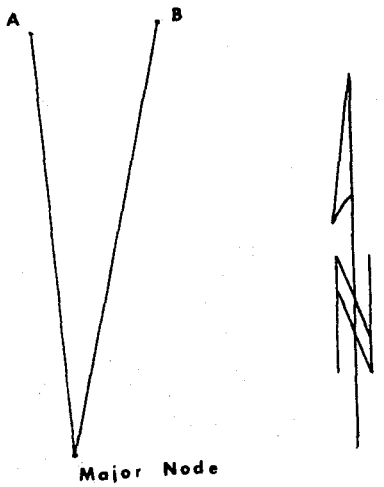


Figure 3.4 Initial Pointer Choice

---

[2] An automated triangulation procedure is being developed.

3) starting at the pointer closest to north and proceeding in a clockwise direction (Figure 3.4) record the labels of the points to which the point in question is connected.

4) if a border (outside extremity of the data set) is encountered between two pointers, record a value of 32000 in the pointer list.

5) if the number of pointers radiating from the point in question is greater than seven (>7), then make the seventh pointer in the list a negative integer.

The absolute value of the integer shall depend upon, i) the number of data points in the set (N) and ii) the number of times, for previous records, the number seven has been surpassed. If, for the data set in question, this is the 1st node for which the number seven has been surpassed then the integer shall have a value of N + 1. If, for the data set in question, this is the jth node for which the number seven has been surpassed then the integer shall have an absolute value of N + j.

If the number of pointers does not exceed seven then return to number 2), otherwise continue.

8) the (N + j)th record must now be filled in. Give it a label of (N + j) and record exactly the same x, y, z coordinates as that of the node in 2).

9) continue recording the pointers as above with the first pointer of the (N + j)th record being the seventh pointer clockwise around the point in question.

10) If the number of pointers happens to exceed thirteen use the same method as above to record them with the label of the new record being (N + j + 1) or (N + 2).

11) When all pointers of the point in question are exhausted (for the number of pointers greater than seven) make the last pointer in the list a negative integer. The absolute value of this integer shall be the label of the point in question (that recorded in number 2)).

Continue this process for all N data points, with the resultant (N + k) records being the data structure representing the surface, where k is the number of nodes with more than seven neighbours.

The data structure so constructed will represent the surface not only relative to an absolute origin, but will also contain the relative relationships between neighbours. Processing the records of this structure will be carried out between logical records and not necessarily between physically adjacent records. All the neighbours of one point may easily be recognized by the explicit relationships designated by the pointers, which in itself is not an advantage over the implicit relationship between grid nodes. However, the nodes are surface specific points rather than simple grid node samplings which do not necessarily characterize the surface.

CHAPTER IV: THE DISPLAY STAGE

Information extraction is the prime objective of all data collections. Data display, for the purpose of extracting information, may be carried out in any one of the following forms;

1) tabulate the observed data,

2) statistically analyze the data and tabulate the results,

3) graphically represent the observed data,

or 4) graphically represent the statistically analyzed data.

In all forms of display the main concern should be to maximise the amount of information obtainable from the results (table, graph, map, etc.) of the study. The purpose of any study and the predetermined use to which information gathered will be applied will generally dictate which display method is best for the data in question. Representation of a surface is dependant upon the total data, information being drawn from the relative positions of data points. Therefore, it is imperative that the data be observed as a whole. The parts of the data structure have no absolute meaning and can only be judged by comparison to their neighbours. For example, it means nothing to say a certain data point has an x, y, z value of (100, 200, 300). The point could be a peak of great relative significance or a point on a relatively flat surface. Surface data should be presented to the observer in a manner such that all neighbourhood relations may be easily observed.

Cartographers, through centuries of experimentation, have developed numerous methods for the graphical representation of surfaces. In 1701 Edmund Halley (in27) published the first isoline

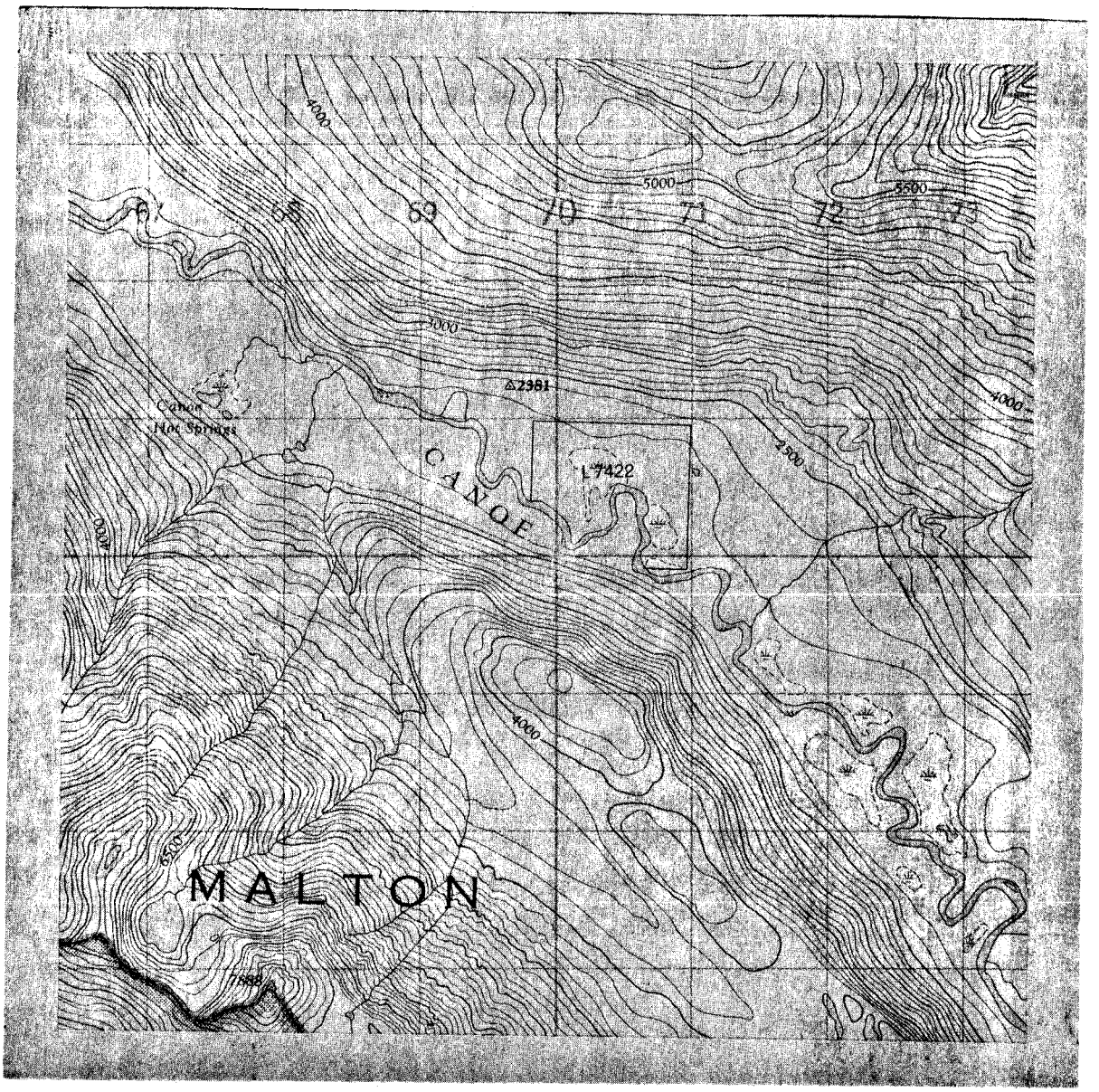


Figure 4.1 Canoe River Valley, Original Topographic Map

map, of isogones or lines of equal magnetic declination. By 1972 when Norman Thrower (27) published his book "Maps & Man" he presented forty-three types of isograms as being a selection "from a much larger number of terms for special forms of the isogram" (27, p 164). During the nineteenth century block diagrams or three quarter view projections were developed and used primarily for illustrating sub-surface structures of the earth in relation to the surface functions. Senefelder (27), in the late eighteenth century, developed a technique of using multiple colours or tonal ranges for the illustration of topographic relief. Hachuring and relief shading of the surface was expanded by Tanaka (27) in the early twentieth century and more recently automated by Yoeli (33). Historically, cartographers have always strived for the "best" graphical representation of the data with which they have been presented.

The focal point of this research has been the representation of surfaces; sampling, storage and display. In order that the data stored in the irregular data structure be accurately portrayed it was necessary to create a new system of display routines that were centered around the data structure. Three display techniques were chosen; contour lines, block diagrams and relief shading. One system of subroutines was then written for each of these techniques with the resultant programs being as follows; ISOKONT, a program for the production of contour maps, EYEBALL, for the production of block diagrams or, three dimensional views, and GREYSHDE, for the production of shaded relief maps. Using the irregular data structure and relying on numerous subroutines (Appendix I) these routines will ideally accomplish the desired result of displaying that information which is collected and stored with as little information loss as possible.

Contouring

ISOKONT is a program for displaying isolines of elevation (contours) from the data of the irregular structure. The basic idea of ISOKONT is very simply as follows;

1) Knowing the z value of the highest data point in the entire structure, calculate for the user-defined contour interval the value of the maximum contour level (i.e. if maximum z = 510 and contour interval = 50, then maximum contour level equals 500).

2) Pass to SCHNET the maximum contour level and the value of the highest data point. SCHNET will then, as described in Appendix I, calculate and plot all the intersection points of this contour level with the edges of the data structure.

3) Check to see if all intersections of the particular contour with the edges have been found. This check is accomplished by setting up a bit or binary matrix in which each entry in the matrix (each cell) is simply a binary figure '1' or '0'. The cell location is defined by the nodes at the end of each edge. For example, the edge formed by the two nodes 56 and 101 will have a matrix location of row 56 by column 101. Each time a new contour level is started the entire matrix is set to '0'. When an edge is passed through by a particular contour, that cell location defined by the nodes of the edge is set to '1'. Thus, in order to ensure that a particular edge has been crossed by the contour, simply check the bit location.

4) Decrease the value of the contour level by the value of the contour interval. If any edges are above the contour level they may be eliminated from any further checks since there is no possibility of intersection with the contour.

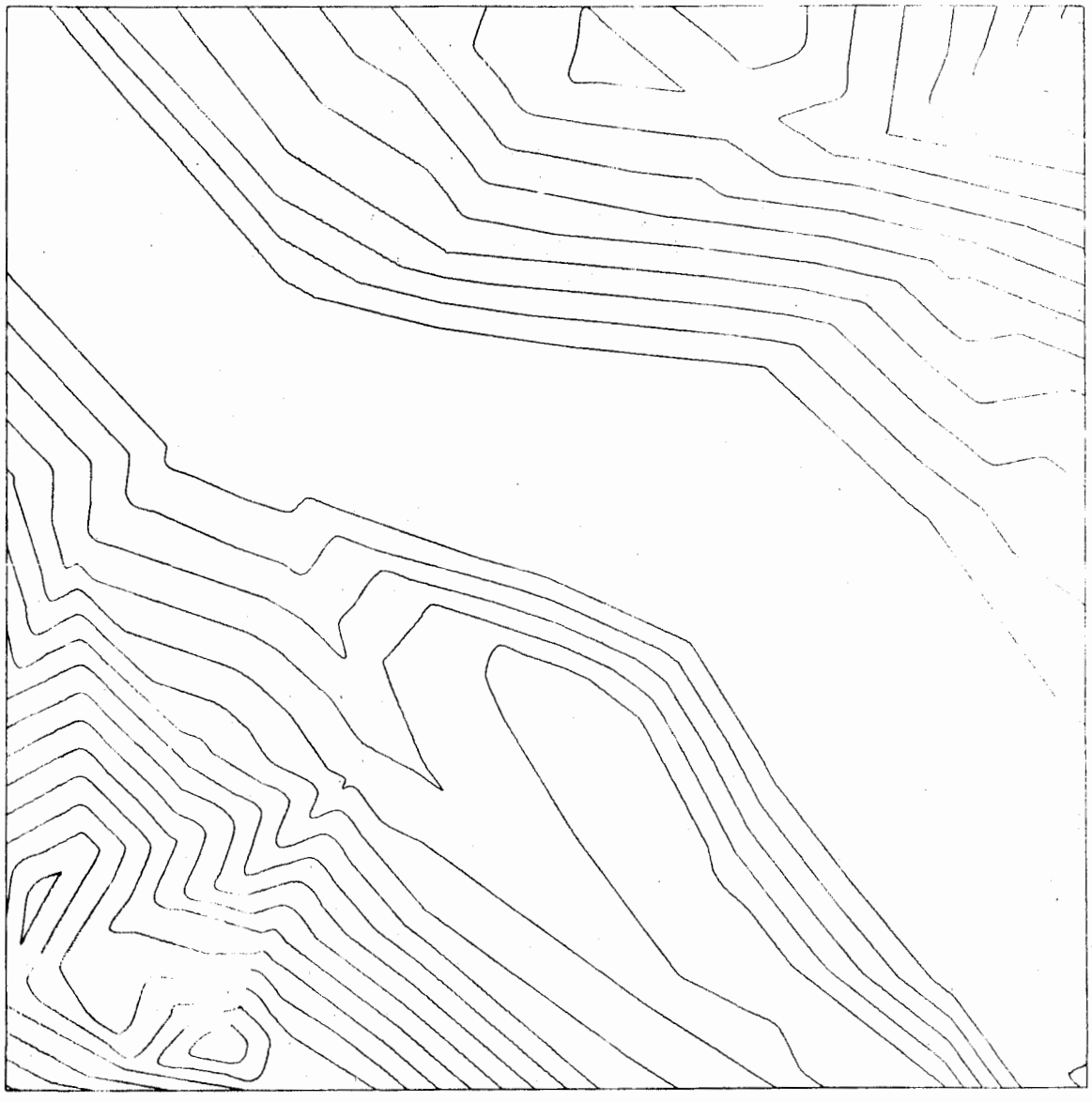


Figure 4.2 Canoe River Valley, 100 Meter Contours. Program: ISOKONT

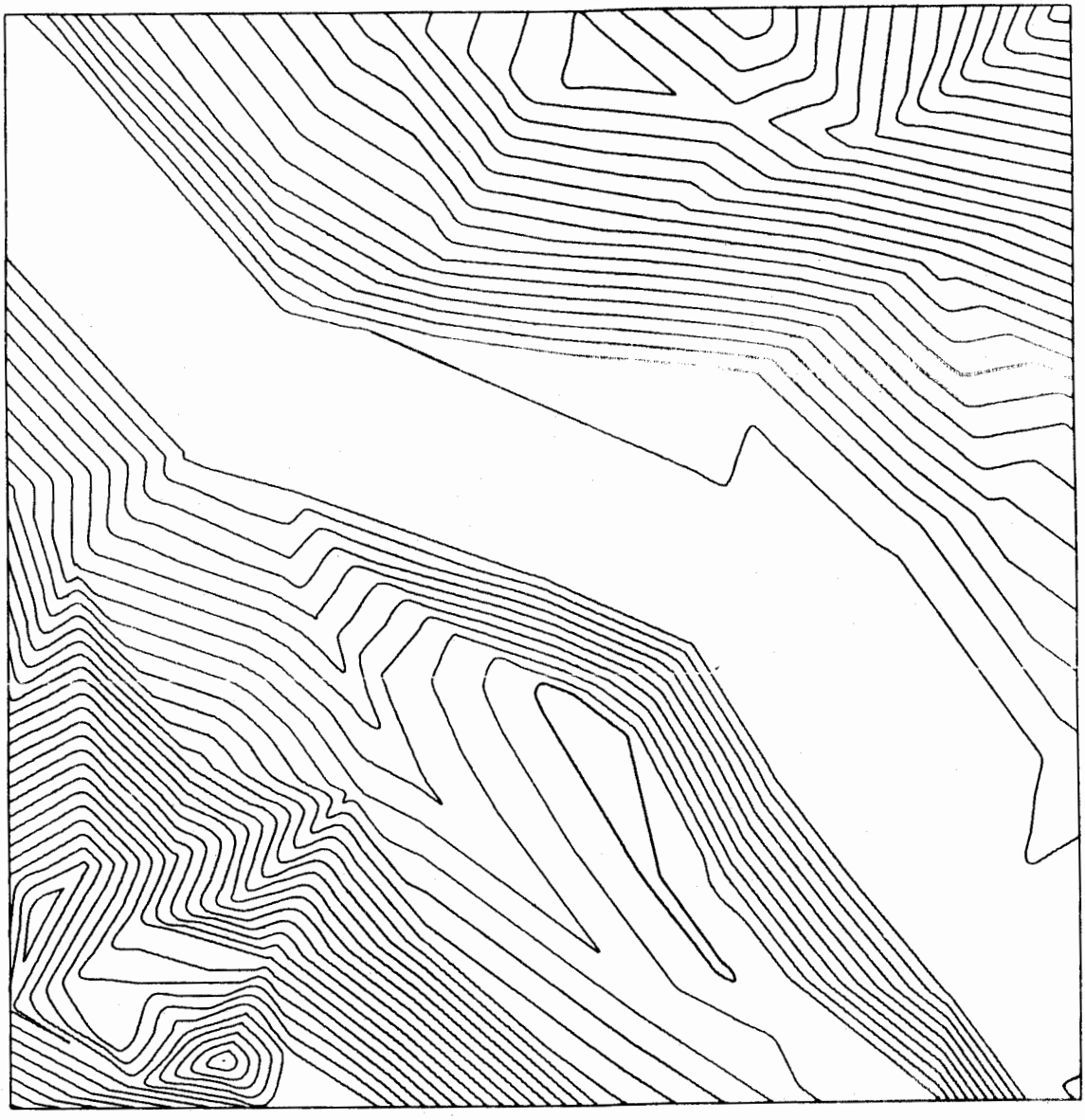


Figure 4.3 Canoe River Valley, 50 Meter Contours. Program: Isokont

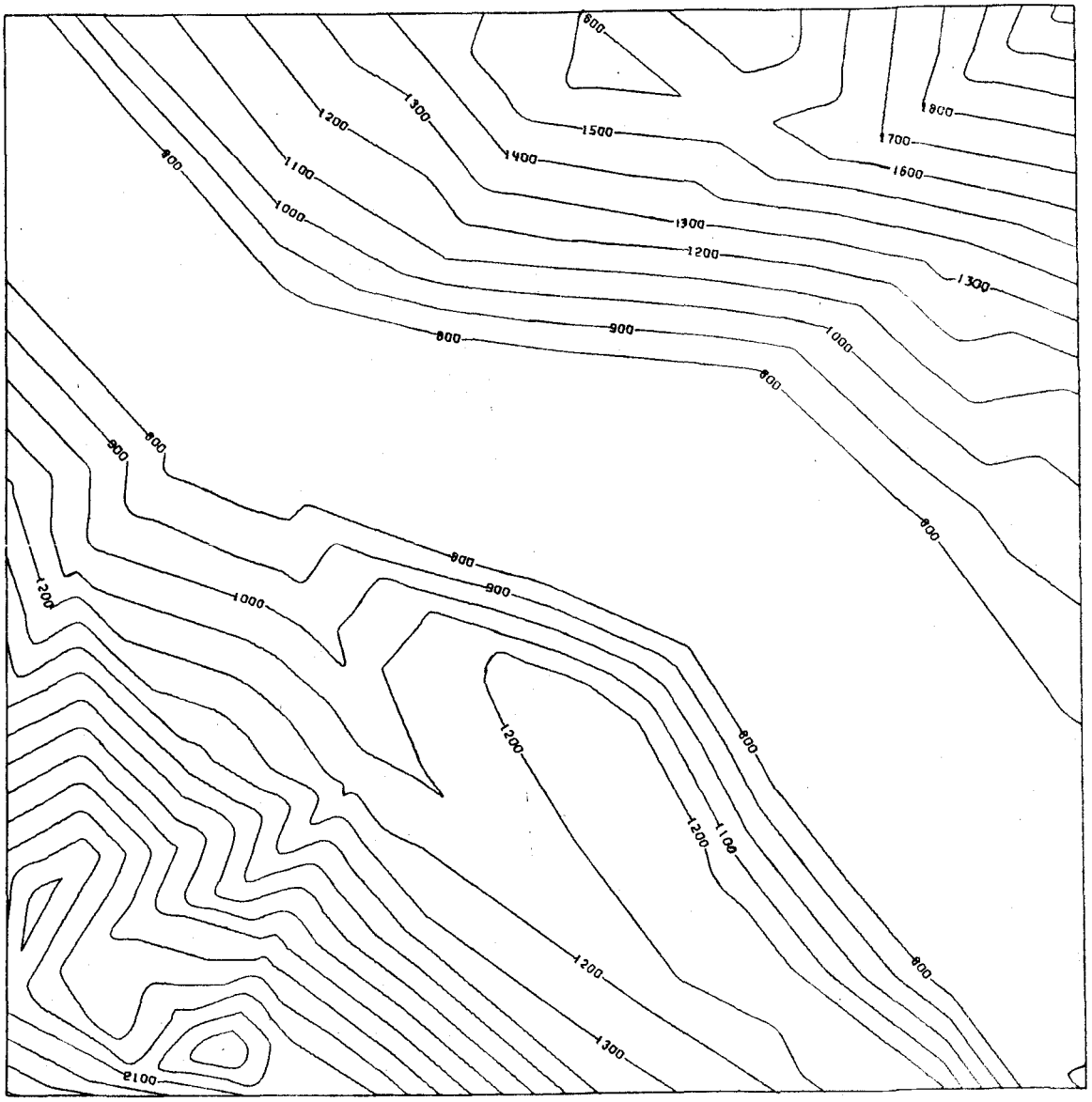


Figure 4.4 Canoe River Valley, 100 Meter Contours. Program: ISOKONT

5) Pass to SCHNET the value of the contour level and any data point which is above this level, but has not been eliminated in Step 4). SCHNET will again calculate and plot the intersections of this contour level with the edges of the data set.

6) Continue steps 3), 4), and 5) until the minimum value of the data set is greater than the value of the contour level.

A listing of ISOKONT may be seen in Figure I.2. ISOKONT will produce maps of any desired interval. Figure 4.2 and Figure 4.3 show contour maps of the Canoe Valley (original topographic map, Figure 4.1). Figure 4.2 shows contours plotted with a one hundred meter contour interval while Figure 4.3 is of a fifty meter interval. Figure 4.4 illustrates the option of having the contour intervals plotted on the contour map. The data set for these particular maps contain only eighty one points - selected by examination of the surface, and choosing only surface specific points.

Block Diagrams

EYEBALL is a routine which calculates and plots various perspective view diagrams of the irregular data structure. For a user-defined viewpoint (point in space from which the user wishes to view the surface) and central point (center of view the user wishes in the data structure) the routine will construct, according to the option specified; true perspective projections, perspective in x and orthographic in y, perspective in y and orthographic in x, or true orthographic projections. Any viewpoint above the minimum z value of the surface may be specified and in fact the viewpoint may actually rest on the surface, allowing the user to "walk" over the surface (Figure 4.5). Also, since the projected surface is composed of surface profiles taken at specified regular intervals from the viewpoint, the user is afforded flexibility in determining the extent of detail on the finished projection (compare Figure 4.6 to Figure 4.7).

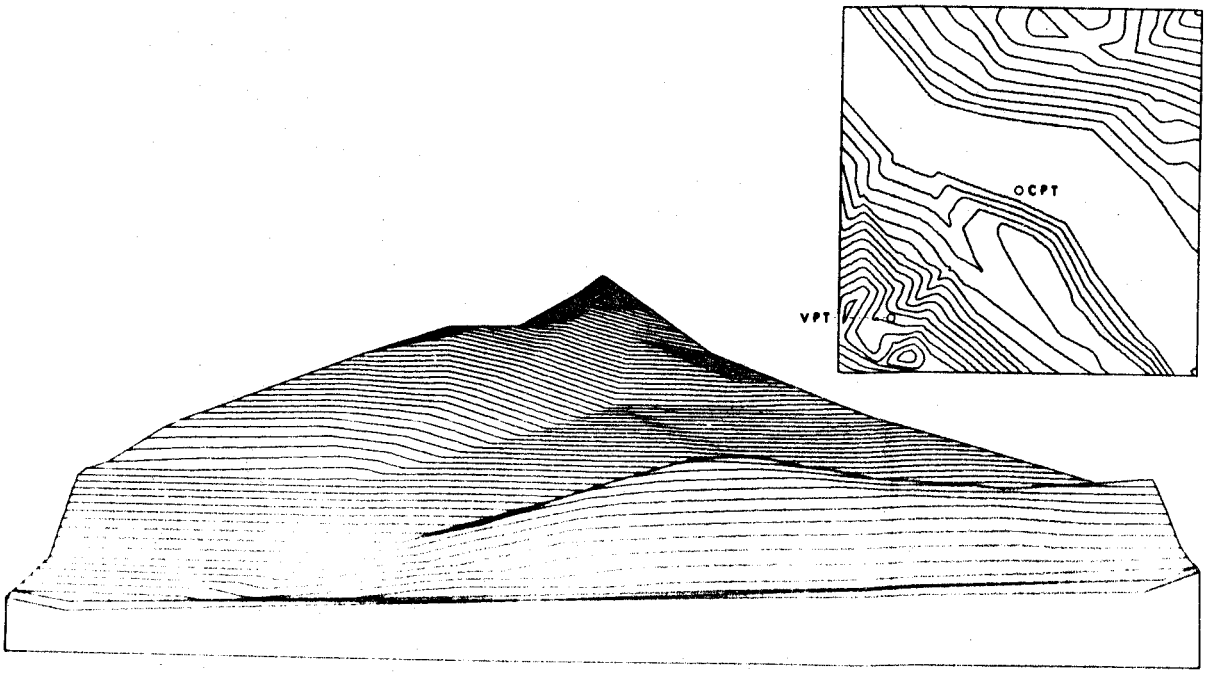


Figure 4.5 Canoe River Valley, Viewpoint Within the Data Structure. Profiling begins at one half the distance between the viewpoint and the central point. Program: EYEBALL

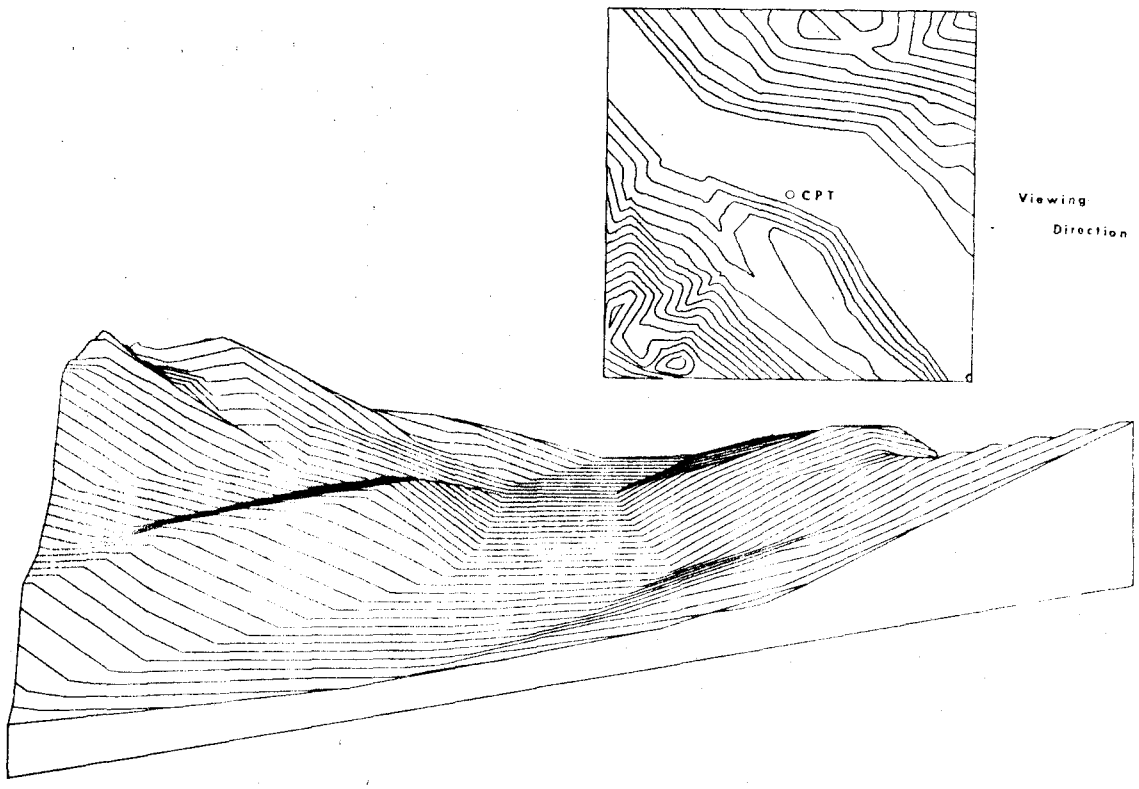


Figure 4.6 Canoe River Valley, 200 Unit Profile Spacing.
Program: EYEBALL

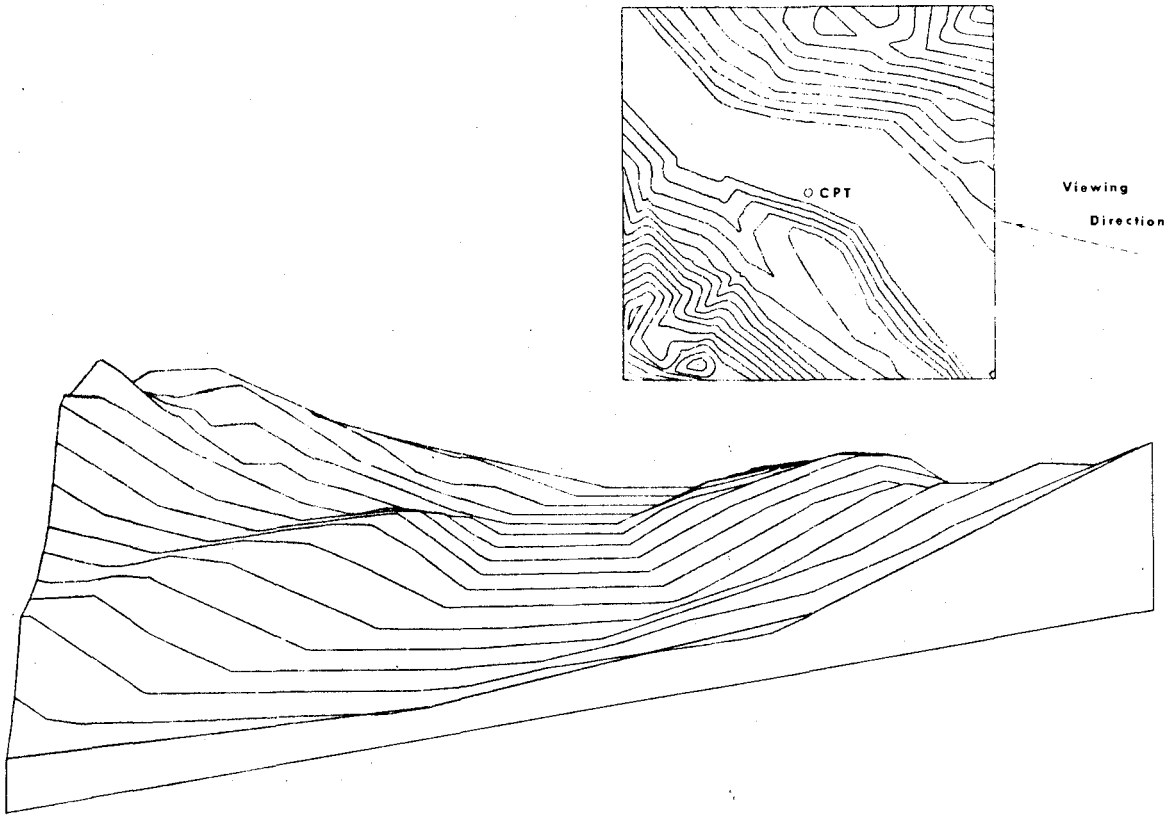


Figure 4.7 Canoe River Valley, 500 Unit Profile Spacing.
Program: EYEBALL

A perspective view of a surface must show only those areas of the surface which would be seen by the eye. It is therefore necessary for any projection routine to eliminate those portions of the surface obscured by surface elements in closer proximity to the viewpoint. A surface point may be defined as hidden if and only if a line segment connecting the surface point and the viewpoint intersects the surface at some point. Kubert, Szabo, and Giulieri (17) outline an algorithm for the construction of this line segment for each surface element and the subsequent test for intersections. EYEBALL eliminates hidden surface elements by moving a visibility profile through the surface at right angles to the line of sight and away from the viewpoint.

The main routine of the EYEBALL system is simply a program to start the operation and to stop it: CONVER. The EYEBALL system operates on the irregular data structure and performs the operations as follows:

1) The entire data set is first assigned new x, y coordinates by transforming and rotating the data such that the viewpoint becomes the new x, y origin and the central point lies on the x axis, Figure 4.8. The z values (height) are left unchanged. Any coordinates now referred to will be the new coordinates.

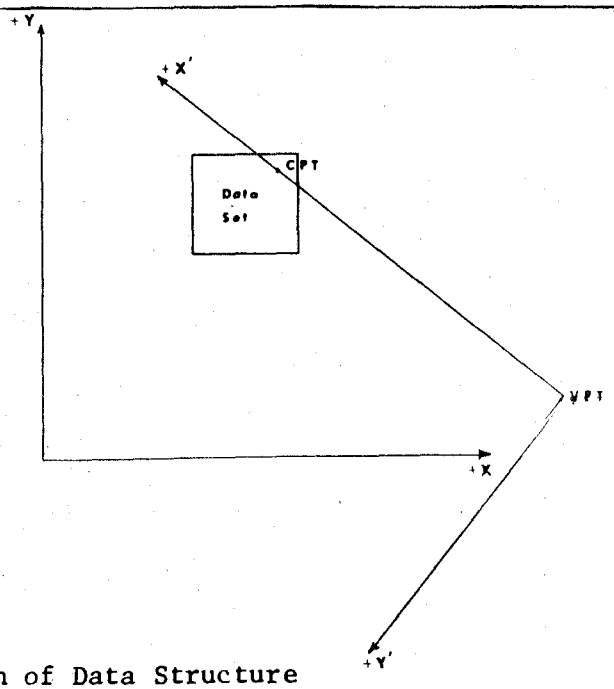


Figure 4.8 Rotation of Data Structure

2) Beginning at either,

i) the minimum x value, or
ii) one half the distance from the viewpoint to the central point, whichever is greater, construct profiles through the data set, at right angles to the x axis and of a user-defined distance apart.

These profiles, Figure 4.9, are constructed by first calculating the endpoints, which are simply the intersection points of the boundary of the data set with a line of given x position and infinite y length, points A and B of Figure 4.9. The coordinates of the endpoints of the desired profile are then passed to SCHNET, which determines the intersection points of this line with the data set and returns the x, y, and z coordinates of these intersections.

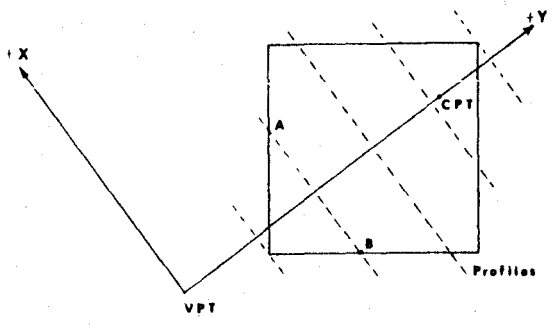


Figure 4.9 Profiling of Data Structure

3) The profiles are then projected onto an imaginary plane which is constructed between the central point and the viewpoint.

4) The profile closest to the viewpoint is then called the horizon and is tested against the second closest profile to the view-

point to see if the second closest profile is visible. The highest visible points (of the horizon and the profile) then become the new horizon, Figure 4.10. This new horizon is then compared to the next profile away from the viewpoint and a new horizon can be formed. Each time a comparison is done the resultant horizon is plotted, whether it is entirely new or just the same as the preceding horizon. This process continues until the last profile, which is less than (by less than one cell width) or equal to the maximum x value of the data set, has been compared to the next to last horizon and has been plotted.

A listing of the program EYEBALL may be seen in Figure I.3.

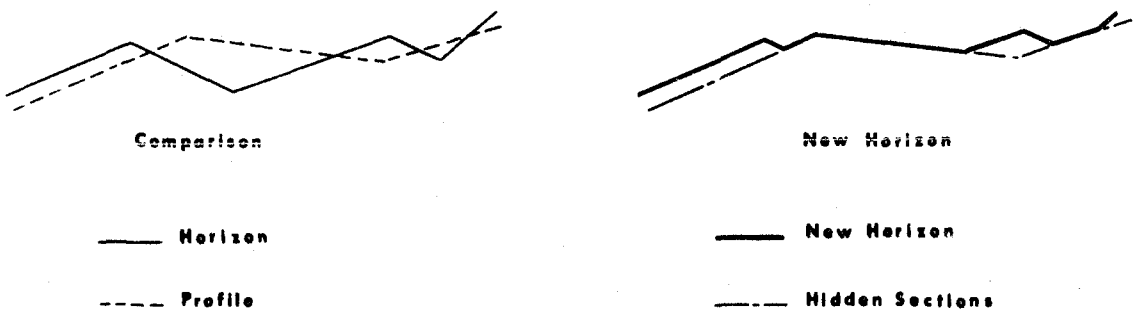


Figure 4.10 Horizon and Profile Comparison

Some of the plots which result from various viewpoints being used with EYEBALL are as follows;

Above each plot it is noted on a contour map the direction from which the view was taken.

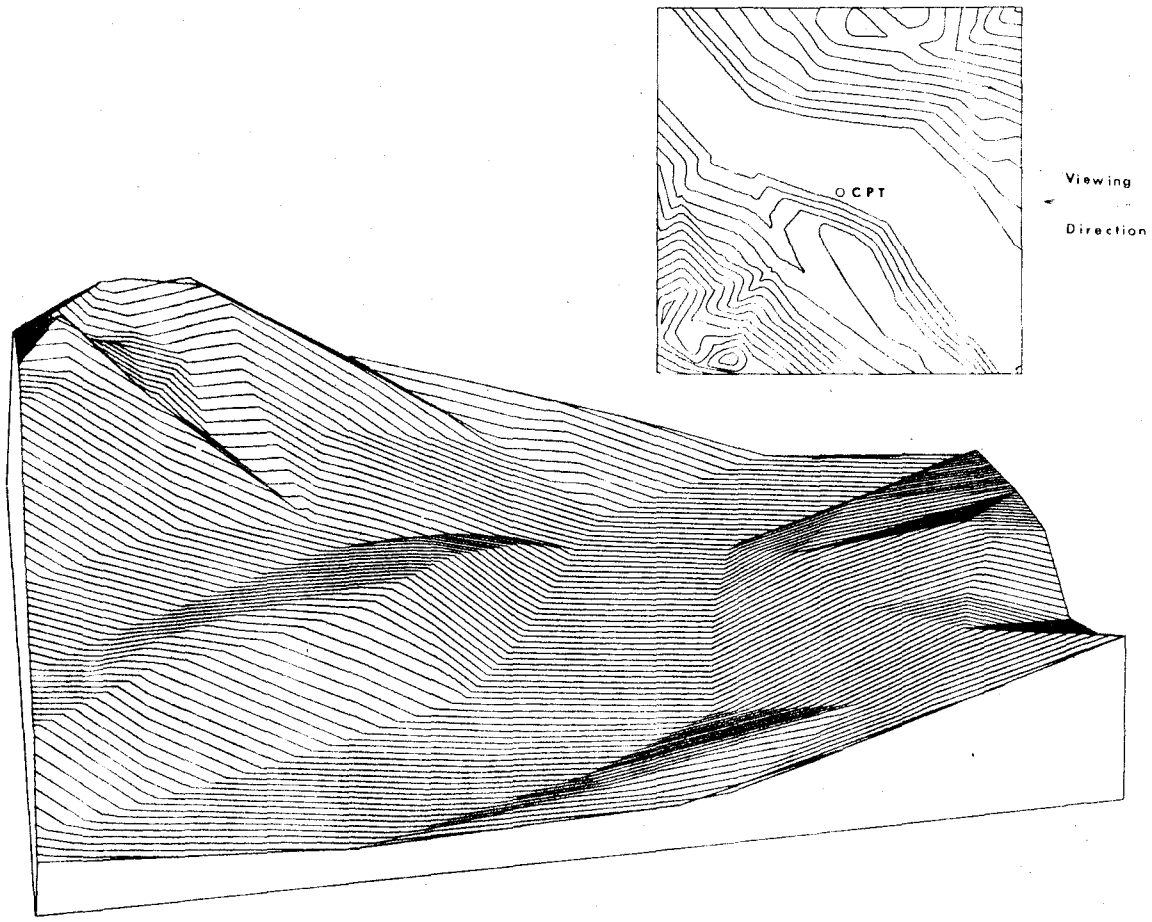


Figure 4.11 Canoe River Valley, Orthographic Projection.
Program: EYEBALL

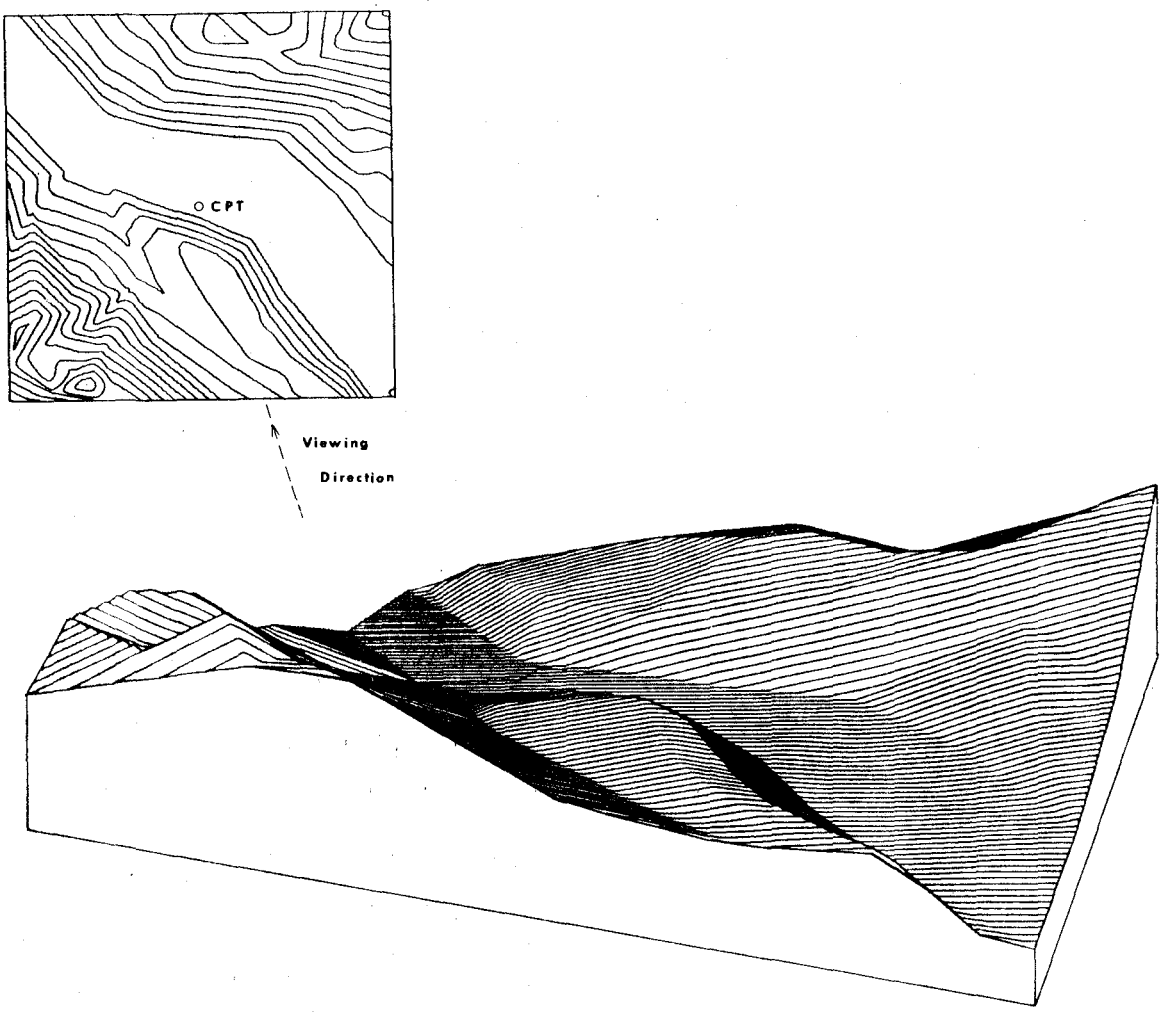


Figure 4.12 Canoe River Valley, True Perspective Projection.
Program: EYEBALL

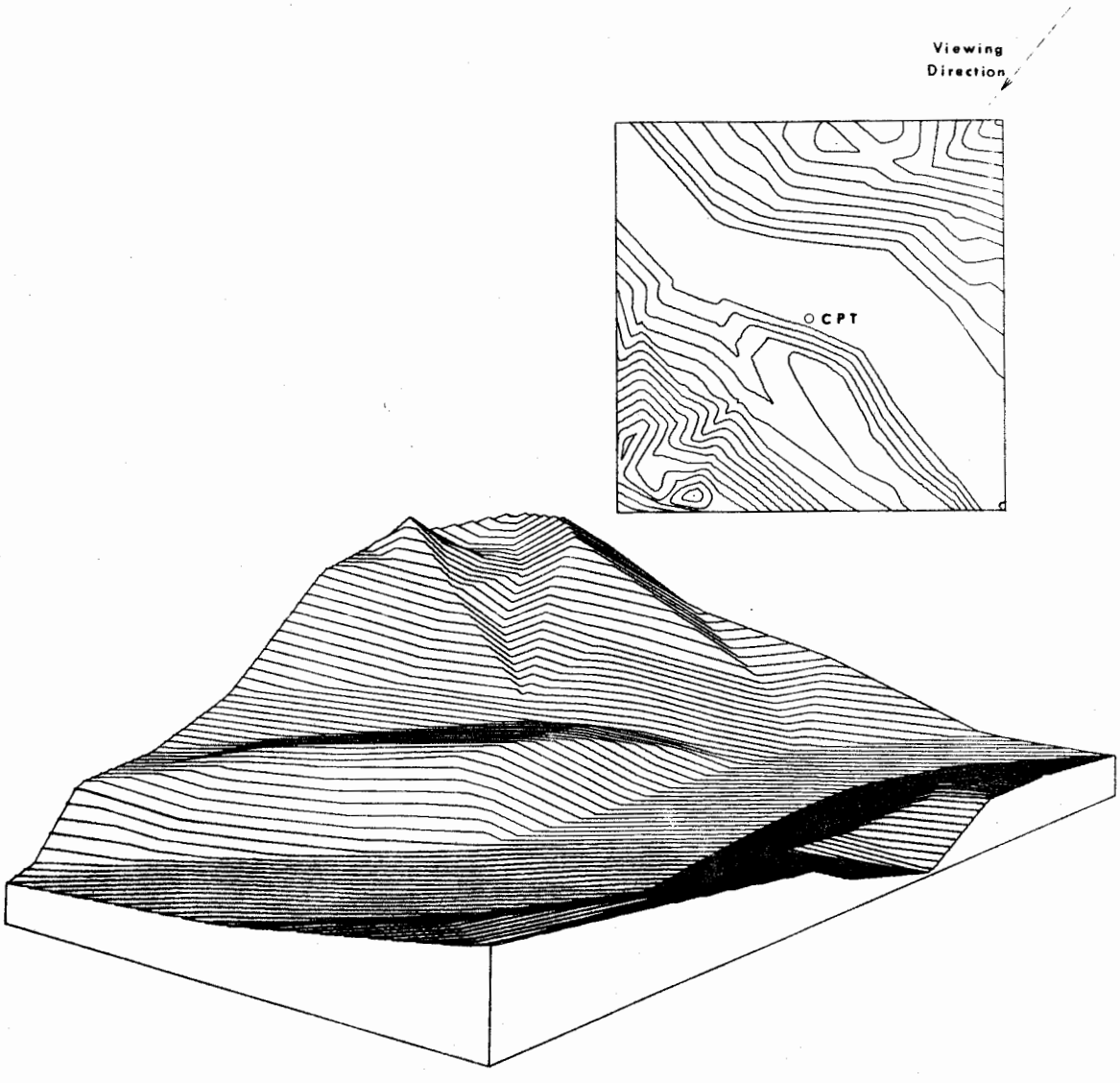


Figure 4.13 Canoe River Valley, Orthographic Projection.
Program: EYEBALL

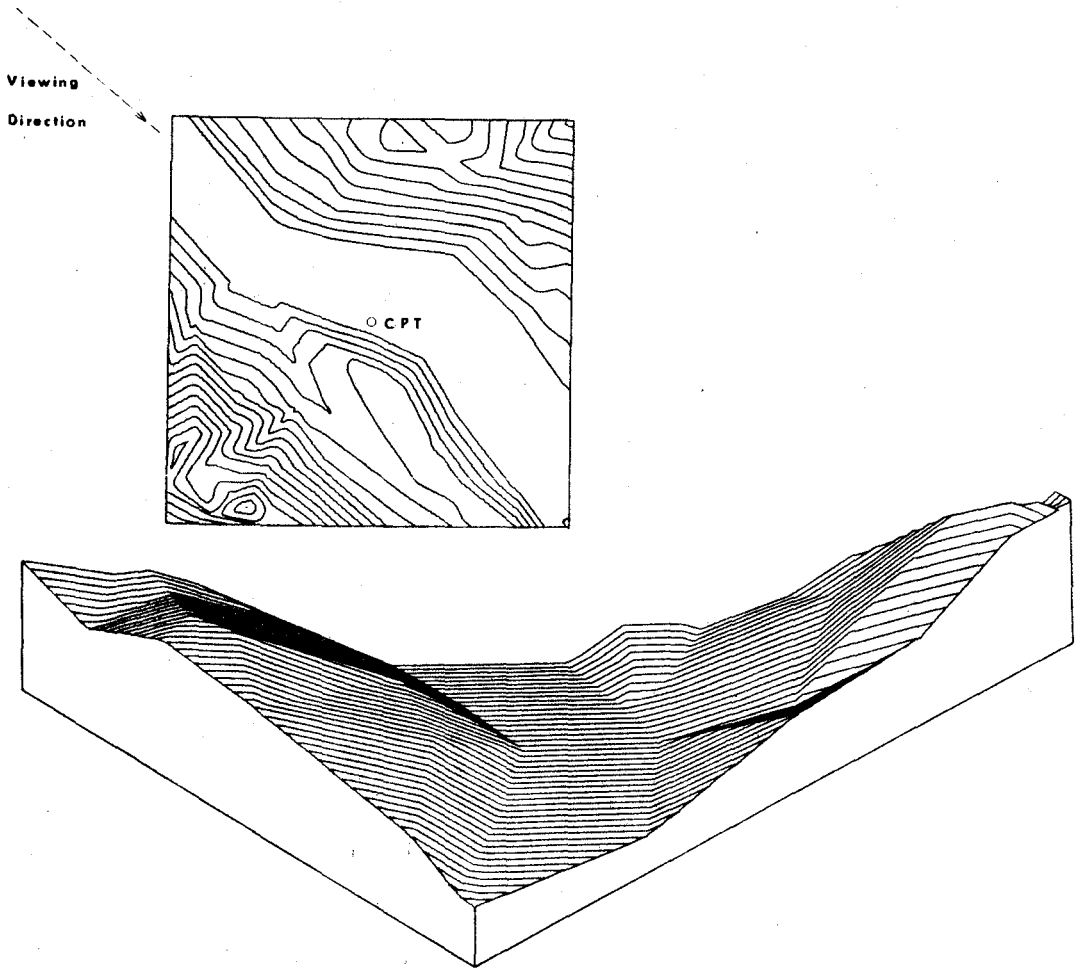


Figure 4.14 Canoe River Valley, True Perspective Projection. This view is of the north west half of the data structure only. Program: EYEBALL

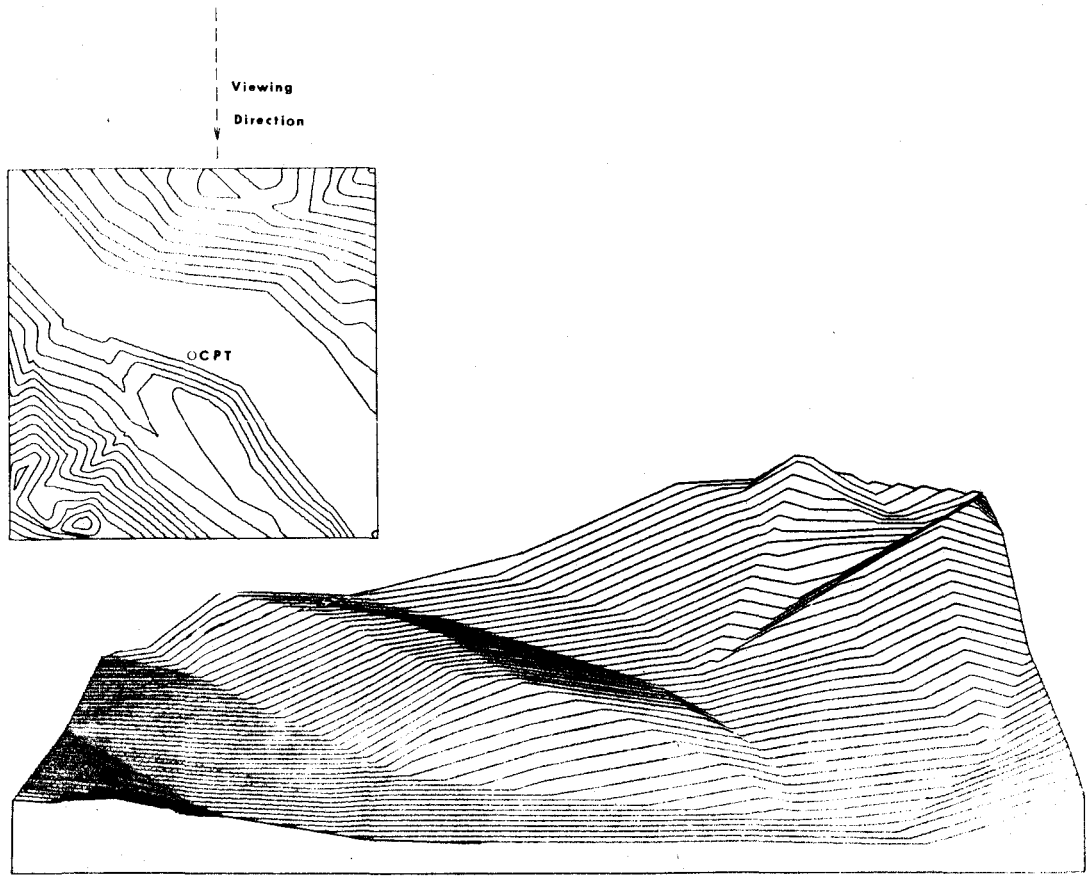


Figure 4.15 Canoe River Valley, Orthographic Projection.
Program: EYEBALL

Figure Number	Description
4.11	Orthographic view of the surface with 100 unit spacing of the profiles.
4.12	True perspective projection with 100 unit spacing of profiles.
4.13	Orthographic view with 100 unit spacing of profiles.
4.14	True perspective projection of only half the data structure (north west corner only) with 100 unit spacing of profiles.
4.15	Orthographic view with 100 unit spacing of profiles.

Analytical Hill Shading

The GREYSHDE system is used for the construction of shaded relief models of the surface as developed by Yoeli (31). The method is somewhat similar to that of Yoeli except that Yoeli calculates the shading of unit squares of the data set and GREYSHDE calculates shading for entire triangular polygons of the data structure. The irregular data structure being comprised of triangular polygons would make it impractical to redivide the surface into smaller regular polygons since the resultant grid cells would be of the same surface function as the larger triangles. Thus the only practical method by which to shade the surface is using triangles.

As Yoeli points out it is impossible to create an exact representation of the earth's relief on a map. However by illuminating a surface with a point light source and reproducing the shading which appears on the surface, a close approximation to reality may be realized. Yoeli begins

his derivation of the theoretical shading of a surface by using Wiechel's equation for light intensity at any point on a surface.

$$\cos(e) = \cos(a) \cdot \cos(b) + \sin(a) \cdot \sin(b) \cdot \cos(c)$$

In which:

e is the angle between the light ray and the normal to the surface at the point in question.

b the slope angle of the surface.

a the complement of the angle of slope of the light ray.

c the angle between the surface and a plane perpendicular to the light ray.

Yoeli uses vector analysis to apply the equation to an area element defined by two vectors in the plane of illumination. The vectors are

$$\vec{a}\ (a_x, a_y, a_z) \quad \text{and} \quad \vec{b}\ (b_x, b_y, b_z).$$

Then, assuming that the length of the light vector $\vec{S}$ (light ray) is one unit the equation

$$\cos(e) = \frac{S_x(a_y b_z - a_z b_y) + S_y(a_z b_x - a_x b_z) + S_z(a_x b_y - a_y b_x)}{\sqrt{(a_y b_z - a_z b_y)^2 + (a_z b_x - a_x b_z)^2 + (a_x b_y - a_y b_x)^2}}$$

is derived. This is a simplification of Wiechel's formula for the light intensity of a rectangular surface element.

As mentioned above, it would be impractical to divide the triangular

polygons into rectangles or squares. Each triangle of the irregular data set is defined by three edges or vectors which are in the plane of the polygon they define, thus any two edges of any particular triangle can be thought of as the vectors $\vec{a}$ and $\vec{b}$.

Given the coordinates of points L, M and N in Figure 4.16 as
$L = (l_1, l_2, l_3)$, $M = (m_1, m_2, m_3)$, and $N = (n_1, n_2, n_3)$
it is a simple matter to calculate the components of the vectors $\vec{a}$ and $\vec{b}$.

$$a_x = n_1 - l_1 \qquad a_y = n_2 - l_2 \qquad a_z = n_3 - l_3$$

$$b_x = m_1 - l_1 \qquad b_y = m_2 - l_2 \qquad b_z = m_3 - l_3$$

If the assumption is then made that the light source is in the northwest and at an angle of inclination of 45^o the components of the vector $\vec{S}$ may be easily found. Having found the values of S_x, S_y, S_z together with the components of $\vec{a}$ and $\vec{b}$, cos e may be solved for.

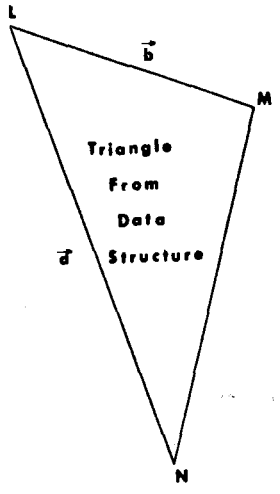


Figure 4.16 Vectors of a Typical Triangle

'Cos e' is the value of the light intensity over the given surface element. The light intensity over the surface varies from white with full illumination to black with no illumination. Full illumination results when the angle of illumination corresponds to the surface normal with light rays hitting the surface perpendicularly yielding a value of 1.00 for cos e. When the surface normal corresponds to the angle of incidence of the light ray and is in the same direction as the light ray, the value of cos e is -1.00; however this rarely occurs in nature (i.e. an overhanging cliff, or double valued surface function). Surface slopes for topographic surfaces are rarely over forty-five degrees, which inclined away from the light ray will give cos e a value of 0.00. Since for slopes of over forty-five degrees away from the light ray, the surface would appear very heavily in shade, all values of cos e less than or equal to 0.00 may be considered as black. Thus the range of cos e from -1.00 to 1.00 will correspond to a surface shading range of black to white with all cos e values less than or equal to zero having a black shading. For the purposes of GREYSHDE this range was divided into ten groups and assigned shading symbols from a light grey to black, Figure 4.17.

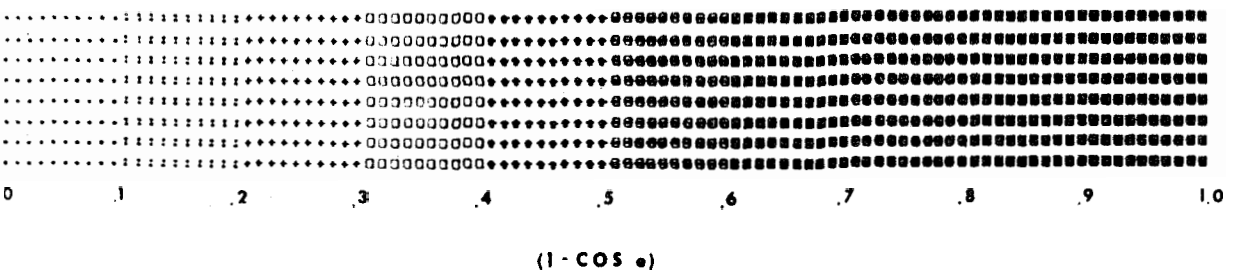


Figure 4.17 Shading Symbol Range

These symbols when used relative to each other give a good impression of surface relief and irregularity.

The algorithm for GREYSHDE is as follows;

1) Read in irregular data structure.

2) Call TRIGLE, a subroutine which forms the vertices of all triangles into three one dimensional arrays. For example, if the jth triangle encountered has the vertex numbers of 12, 13, 14 then array one contains 12 in the jth position, array two contains 13 in the jth position and array three contains 14 in the jth position.

3) Call SHADE, a subroutine which calculates the grey shading of every triangle in the data structure by the method outlined above.

4) Using SCHNET calculate the x, y coordinates of the intersection points of the data structure and a line of constant y value passed through the data structure from minimum x to maximum x. The starting, or initial, y value of the line is the maximum y value of the data.

5) Moving from the minimum x value of the intersection points found, to the maximum, pair the intersection points as follows;

(1st, 2nd), (2nd, 3rd), (3rd, 4th), ..., ((n - 1)st, nth).

6) Calculate the x coordinate of the midpoint of each pair and determine within which triangle of the data structure each midpoint lies.

7) Assign the shading values of the appropriate triangles to the sections of the line represented by each intersection pair. Therefore, from the first intersection point to the second intersection point the line will be assigned a shading value of the triangle within which the line between the intersection points lies.

8) Assemble and output a line of print (for the line printer), each character of which is based upon the shading value it is to represent, Figure 4.17.

9) Decrease the y value of the line (as calculated in 4) by a certain interval and calculate the intersection points of this line with the data structure.

10) Go back to step 5) and continue this process until the minimum y value of the data structure is encountered by the line moving downward through the structure.

A listing of the GREYSHDE program may be seen in Figure I.4. The product of GREYSHDE is a shaded relief map (Figure 4.18) which is a display of a much more general nature than either the contour map or the perspective view. However, the shaded relief map does carry information of a much higher level and constitutes generalities rather than particulars.

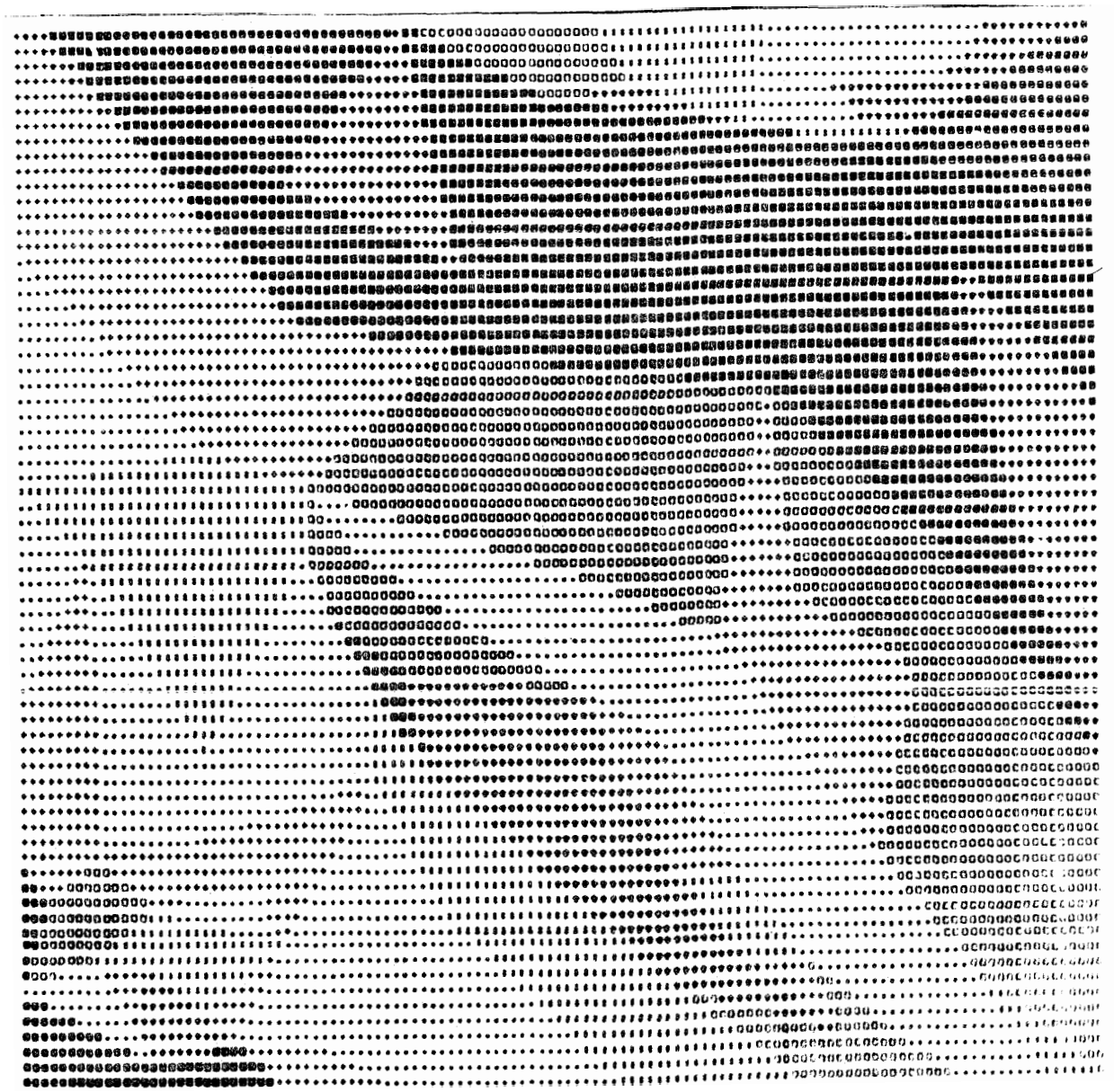


Figure 4.18 Canoe River Valley, Shaded Relief Map.
Program: GREYSHDE

CHAPTER V: CONCLUSION

Nearly every field of academic work has felt the impact of the computer. With its extensive storage facilities, rapid access to data and high speed of calculation the computer has the possibility of revolutionizing cartography. As a communicative process, the object of cartography is to convey to an independant observer as much information concerning a geographic area or spatial phenomena as is possible. Information is generally lost at each of three stages within the process; collection, storage and display. This thesis has been an endeavor to illustrate how, with an alternative approach to data collection, and through implementation of the addressing facilities offered by the computer, retention of information can be maximized throughout the process. The approach recommended as an alternative to regular grid collection and/or storage is that of collection of surface specific points and storage of the absolute coordinates of these points and their spatial relationships to neighbours with a linked list data structure.

One important innovation of the data structure developed is its ability to convey and retain information about the relation between surface specific points. This is important because a surface function is dependant upon the relation and interaction between points for the majority of surface information. A regular grid will of course yield this information, however the amount of necessary surface data must be increased along with the processing time. The linked list data structure appears, by the tests run thus far, to maximize information retention while saving on data storage space and processing time.

The systems for the display of the irregular data structure have been designed to give the user flexibility in viewing the surface. From

the filtered or smoothed representation afforded by GREYSHDE to the minute isoline detail of ISOKONT, the user may choose the amount of information to be displayed. GREYSHDE, through use of the triangle, or neighbourhood, as the lowest level data item, filters or channels the information into groups, giving the user insight to the relationships of neighbourhoods. GREYSHDE gives no absolute surface values, only the surface function relations. EYEBALL allows the user to view the surface from a normal perspective, again no absolute specifics of the surface appear on the reproduction, only the relationships between neighbourhoods. However, these relationships may be further enhanced by a change of profile density, a change of viewing position or a change of projection. The most detailed display is given by ISOKONT, with the user dictating the amount of information to be on the finished plot simply through adjustment of the contour interval. These systems allow the information of the surface specific points related through position to be plotted with a high degree of information retention. The displays are, as nearly as possible, that information which has been collected, no loss can be attributed to storage or output (display) techniques.

The principles developed in this thesis should serve as a basis to ongoing research for the development of efficient geographic data structures. Geography, being a study of dynamic phenomena, needs data structures capable of handling ever changing spatial distributions. Data structures which are able to cope with the interactions and inter-relationships between groups of data. The data structure herein developed may not be capable of solving these problems but should help in the development of new structures; if only to point out a need for development.

APPENDIX I: PROGRAMS AND SUBROUTINES

The three display routines outlined in Chapter IV were dependant upon numerous subroutines for their efficient operation. One of these was of major importance and will be discussed in some detail while the rest are self explanatory and may be seen in Figure I.2 thru Figure I.23.

SCHNET is a routine for searching and developing a path through the irregular data structure. The path chosen by SCHNET is based on several possible criteria, one of which must be supplied by the invoking program. The following criteria were chosen for SCHNET;

a) The path of the Jth isoline - find the intersection points of a path created by a plane of height J intersecting the data structure. SCHNET will return the x, y coordinates of the intersection points.

b) The path of a vertical plane - find the intersection points of a vertical plane passing through the data structure. SCHNET will return the x, y, z coordinates of the path. See Figure I.1.

c) The data point closest to a given x, y coordinate. Given any x, y coordinate, preferably within the data set boundaries, SCHNET will return the label of the data point which is closest to this point.

The path following algorithm for a network was first developed by Dayhoff (9) and used for contouring a regular grid pattern in X-Ray crystallography. The algorithm, as used by SCHNET, is basically the same as Dayhoff's except that minor changes have been made to accommodate both a new data structure and the criteria of a vertical plane. Basically SCHNET tests each edge in a systematic manner for an intersection with the path. A few definitions are necessary prior to a complete

description of the algorithm:

RP - reference point, normally the major node of the edge being tested. Swinging always occurs around the RP.

SP - sub point, is the secondary node of the edge that is being tested. Always the changing point, or the point which is being changed in the swinging procedure.

EP - edge point, is the point which is being calculated as the intersection between the path and the edge defined by the nodes RP and SP.

The algorithm for SCHNET is;

1) SCHNET is passed, from the invoking procedure, the first RP to be used, the value of which is dependant upon the criteria chosen. For the criteria of following a contour path, RP is chosen as any data point which is above the value of the contour level (path to be followed) but has not been eliminated due to subsequent processing (see write-up of ISOKONT). For the criteria of following the path of a vertical plane (straight line) RP is chosen as the data point closest to one end of the vertical plane (i.e. the vertical plane becomes a line in the x, y plane defined by two points at the ends of the line) This point may be found with a subsequent call to SCHNET using criteria c) from above. For the criteria of finding the data point closest to a given x, y coordinate location the RP used may be the label of any data point.

2) SP is chosen as the data point pointed to by the first (sequentially measured) pointer in RP's pointer list.

3) The edge formed by RP - SP is checked for an intersection (EP) with the path.

4) a) if one exists it is calculated and stored. SP is then set to the data value pointed to by the next (sequentially) pointer in RP's pointer list - 3) is repeated unless of course the next pointer in RP's list is zero or blank - indicating that all RP's neighbours have yielded intersection points. Which for the contour path indicates a closed continuous contour around RP in which case control is returned to the invoking procedure. For the vertical plane criteria this is impossible unless of course RP is on a boundary of the data set and then one of RP's pointers would have a value of 32000 which would indicate to SCHNET that this condition may occur and to check for it.

b) i) if no EP is found, and, no EP has yet to be found for this particular RP, then SP is set to the data value pointed to by the next (sequentially) pointer of RP's pointer list. 3) is repeated unless of course the next pointer in RP's list is zero or blank.- indicating that all RP's neighbours have been checked and found to yield no path intersections. Control is returned to the invoking routine which must give SCHNET a new RP.

ii) But, if no EP is found and one was found for the last SP (i.e. 4a was performed for the last pointer in RP's list) then 6 is performed.

6) RP is set to the data value of SP and SP is set to the data value pointed to by the pointer in the new RP's list immediately following the pointer to the old RP.

7) The edge RP - SP is checked for an EP. One must exist since the path has entered the triangle and unless it stopped inside the triangle (the end of a straight line) it must leave the triangle through this side (since it did not leave through the side defined by the nodes old RP, SP.

8) SP is set to the data value pointed to by the next pointer in RP's list.

9) Go back to statement 3).

This process of swinging through RP's and SP's will continue until one of the stopping mechanisms is encountered. The stopping mechanism for a contour path could be one of two conditions;

a) the contour is a continuous closed curve which eventually returns to the initial RP.

b) the contour encounters one of the boundaries of the data set and must stop, in which case SCHNET transfers the path following procedure back to the beginning of the path at the initial RP. SCHNET then continues to follow the path in the opposite direction until another border is reached in which case the contour has been completed and control is returned to the invoking procedure.

The stopping mechanism for a straight line may be one of two events occuring

a) a boundary of the data set is reached, but since the initial RP was one end of the line or where the line entered the data set all intersections have been found and control is returned to the invoking procedure.

b) an end point of the line has been reached and since the initial RP was one end of the line or where the line entered the data set,

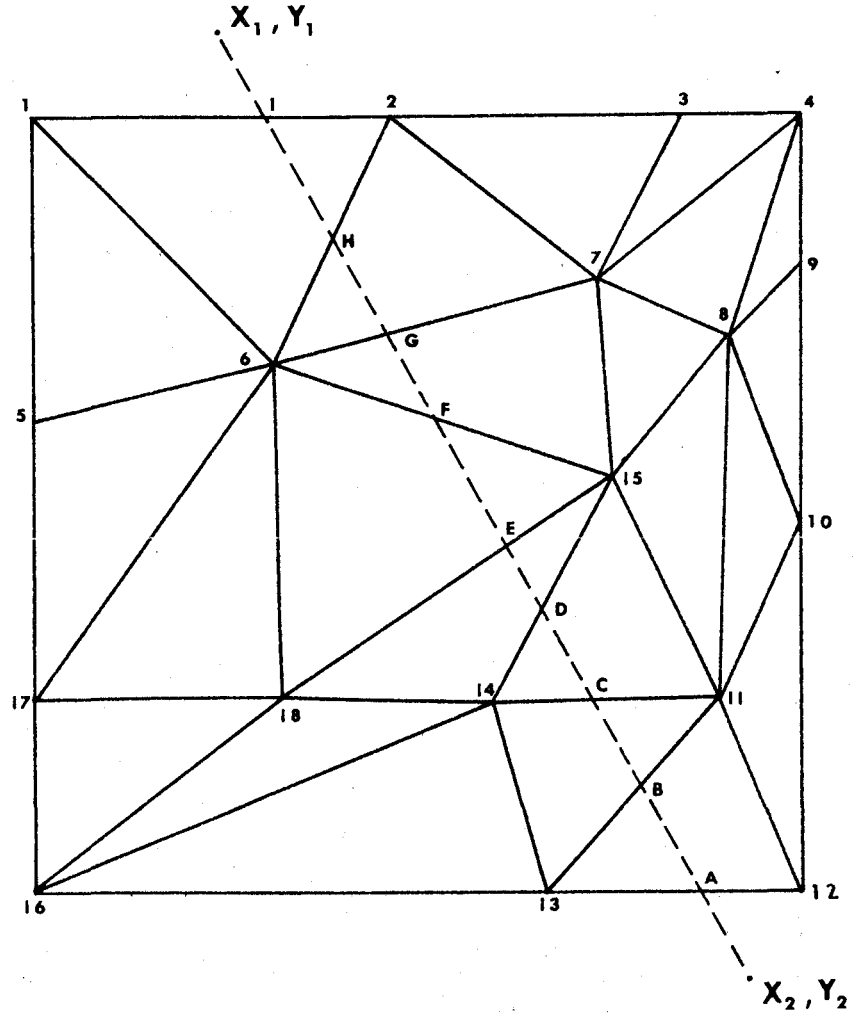


Figure I.1 Vertical Plane Through the Data Structure. SCHNET operates on the data structure and the vertical plane to find the intersection points of the two.

control is returned to the invoking procedure.

A sample SCHNET run as illustrated in Figure I.1 is as follows;

Initially RP is 1

RP	SP	EP
1	2	I
1	6	None
6	2	H
6	7	G
6	15	F
6	18	None
18	15	E
18	14	None
14	15	D
14	11	C
14	13	None
13	11	B
13	12	A
13	32000	None

The final SP being 32000 indicates to SCHNET that a border has been reached. The values of EP as intersection coordinates are then returned to the invoking procedure.

The listing for SCHNET may be seen in Figure I.7 thru Figure I.11 .

The programs of the display system use the subroutines as follows;

EYEBALL	GREYSHDE	ISOKONT
READER1	READER1	READER1
CONVER	MAXMIN	MAXMIN
CENTRE	CENTRE	SORTER
MAXMIN	TRIGLE	SCHNET
ROTATE	SETIS	WORK1
STRIP	RESET	BORDER
SCHNET	SHADE	PLOTS[2]
WORK1	POINTS	FACTOR[2]
WORK2	SCHNET	PLOT[2]
SORTER	STRIP	NUMBER[2]
PRJN1[1]	OUTSID	
RIDGE[1]	PNPOLY[1]	
PLOTS[2]		
PLOT[2]		
FACTOR[2]		

The subroutine SCHNET always uses the following subroutines;

ISON
LAND
LOR
FINDIT
ANGFND
MARKIT
SMTHLN[1]
AINTER[1]
CROSS[1]

---

1 Douglas, David, H. "Graphic and Cartographic Subroutine Library," unpublished but used for teaching purposes at The University of Ottawa.

2 Software subroutine package for Simon Fraser University Calcomp plotter.

```
ISOKONT:PROC OPTIONS(MAIN);
/*
    THIS PROGRAM IS DESIGNED FOR THE DRAWING OF CONTOUR MAPS OF
    DATA STORED IN AN IRREGULAR MANNER AND TRIANGULATED.
    INPUT: (ON THE FILE SYSIN)
          COL  1- 5 THE NUMBER OF DATA POINTS IN THE INPUT DATA SET
                    INCLUDING DUMMY POINTS, (I.E. THE TOTAL NUMBER
                    OF DATA POINTS TO BE CONSIDERED FOR THE MAP)
                    AS AN INTEGER NUMBER.
          COL  6-10 THE NUMBER OF DATA POINTS IN THE INPUT DATA SET
                    MINUS THE NUMBER OF DUMMY POINTS, AS AN INTEGER
                    NUMBER.
          COL 11-15 THE MINIMUM Z VALUE IN THE DATA SET, AS A REAL
                    NUMBER.
          COL 16-20 THE MAXIMUM Z VALUE IN THE DATA SET, AS A REAL
                    NUMBER.
          COL 21-25 THE CONTOUR INTERVAL DESIRED ON THE OUTPUT MAP
                    AS A REAL NUMBER.
          COL 26-30 THE FACTOR BY WHICH ALL VALUES TO BE PLOTTED MUST
                    BE MULTIPLIED IN ORDER THAT THEY WILL FIT ON THE
                    OUTPUT PAPER. DEFAULT FOR THIS VALUE IS TO FIT
                    THE PLOT ON 10 INCH PAPER.
                    AS A REAL NUMBER.
          COL 31-35 AN INTEGER VALUE WHICH TELLS THE PROGRAM IF THE
                    USER WANTS TO HAVE THE CONTOUR VALUES PRINTED
                    ON THE OUTPUT MAP OR NOT.
                    ANY NON-ZERO INTEGER INDICATES THAT THE USER
                    DESIRES THE CONTOUR VALUES PRINTED.
                    ZERO INDICATES NO CONTOUR VALUES TO BE
                    PRINTED.
  */
DCL
      (X(*),Y(*),Z(*)) CTL BIN FLOAT,
      XFACT STATIC BIN FLOAT,
      A(*) CTL BIN FLOAT,
      NSORT(*) CTL BIN FIXED (31),
      NPT(*,*) CTL BIN FIXED(31),
      (RP,CONT,X1,Y1,X2,Y2,CRIT,XMIN,XMAX,YMIN,YMAX)
       STATIC BIN FLOAT,
      (NR,N,NCALL,N1,J,NCON,I,NWHAT,L1,L2,NRIT,LET)
       STATIC BIN FIXED(31),
     (READER1,SORTER) ENTRY EXTERNAL,
     (DATABK) FILE RECORD KEYED DIRECT ENV(REGIONAL(1));
      OPEN FILE(DATABK) DIRECT INPUT;
      CALL PLOTS(1,J);
      GET EDIT(N,N1,ZMIN,ZMAX,CONI,XFACT,LET)
            (COL(1),7 F(5,0));
      ALLOCATE X(N),Y(N),Z(N),NPT(N,07),NSORT(N1);
      NRIT=0;
      CALL READER1(X,Y,Z,NPT,N,DATABK);
      NWHAT=0;
      CALL MAXMIN(X(1),XMIN,N,J,NWHAT);
      NWHAT=1;
      CALL MAXMIN(X(1),XMAX,N,J,NWHAT);
      CALL MAXMIN(Y(1),YMAX,N,J,NWHAT);
      NWHAT=0;
      CALL MAXMIN(Y(1),YMIN,N,J,NWHAT);
      IF XFACT=0. THEN DO;
      TMAX=MAX((XMAX-XMIN),(YMAX-YMIN));
      XFACT=9./TMAX;
      END;
      CALL FACTOR(XFACT);
/*  THE DATA POINTS ARE FIRST SORTED ACCORDING TO THEIR Z VALUES */
      CALL SORTER(Z(1),NSORT(1),N1);
      NCON=ZMAX/CONI;
      TOP=NCON*CONI;
/*  THE FINDING OF CONTOURS BEGINS AT THE HIGEST VALUE AND WORKS ITS
    WAY DOWN  */
      DO CONT = TOP TO ZMIN BY (-CONI);
      NCALL=0;
      DO J = N1 TO 1 BY -1;
      IF     Z(NSORT(J))<CONT THEN GO TO NEXT;
      NR=NSORT(J);
      RP=Z(NR);
      CALL SCHNET(X(1),Y(1),Z(1),NPT(1,1),RP,CONT,NR,N,X1,X2,Y1,Y2,
                  NRIT,NCALL,L1,L2,XFACT,LET);
AGAIN:END;
NEXT: END;
      CALL BORDER(XMAX,XMIN,YMAX,YMIN,XFACT);
      PUT SKIP LIST('SUCCESSFUL COMPLETION OF CONTOUR MAP');
      END ISOKONT;
```

Figure I.2 ISOKONT

```
EYEBALL:PROC OPTIONS(MAIN);

EYEBALL IS A PROGRAM WRITTEN FOR THE PURPOSE OF
PLOTTING THREE DIMENSIONAL VIEWS OF THE SURFACE
REPRESENTED BY THE IRREGULAR DATA STRUCTURE.

DATA:
    ON THE UNIT SYSIN

N     - THE NUMBER OF POINTS IN THE DATA STRUCTURE,
        BOTH REAL AND DUMMY.
NPROJ - TYPE OF PROJECTION REQUIRED
        1 - TRUE PERSPECTIVE PROJECTION
        2 - PERSPECTIVE IN X; ORTHOGRAPHIC IN Y
        3 - ORTHOGRAPHIC IN X; PERSPECTIVE IN Y
        4 - TRUE ORTHOGRAPHIC PROJECTION

FORMAT 2 F(5,0)

ON THE UNIT FT06F001

CARD 1 FORMAT 3F(10.0)

VPT - A THREE MEMBER ARRAY CONTAINING THE
      X,Y AND Z COORDINATES RESPECTIVELY OF
      THE VIEWPOINT DESIRED BY THE USER.

CARD 2 FORMAT 3F(10.0)

CPT - A THREE MEMBER ARRAY CONTAINING THE
      X,Y AND Z COORDINATES RESPECTIVELY OF
      THE CENTRAL POINT DESIRED BY THE USER.

CARD 3 FROMAT F(5.0)

CELLS - THE SPACING OF THE PROFILES DESIRED BY
        THE USER.
                                                */
DCL

(X(*),Y(*),Z(*))CTL BIN FLOAT,
NPT(*,*)CTL BIN FIXED(31),
(READER1) ENTRY EXTERNAL,
(N,NPROJ) BIN FIXED(31),
(DATABK) FILE RECORD KEYED DIRECT ENV(REGIONAL(1));
GET EDIT(N,NPROJ)
(COL(1),2 F(5,0));
ALLOCATE X(N),Y(N),Z(N),NPT(N,07);
CALL PLOTS(I,J);
CALL READER1(X,Y,Z,NPT,N,DATABK);
CALL CONVER (X(1),Y(1),Z(1),NPT(1,1),N,NPROJ);
END EYEBALL;
```

```
READER1:PROC (X,Y,Z,NPT,N,DATABK);

/*
     READER IS A SUBROUTINE DESIGNED FOR THE PURPOSE OF READING IN DATA
          STORED IN AN IRREGULAR FASHION WHICH HAS BEEN TRIANGULATED
          AND WRITTEN ON DISK STORAGE AS A PL/1 REGIONAL 1 TYPE FILE
          THE X,Y,Z COORDINATES ARE READ IN AND STORED IN ONE
          DIMENSIONAL ARRAYS, WHILE THE POINTERS OF EACH RECORD ARE
          READ IN AND STORED IN A TWO DIMENSIONAL ARRAY.
          THE WHOLE DATA SET IS READ IN AT ONCE AND STORED
          IN ARRAYS.
     PARAMETERS:
          X,Y,Z ARE CONTROLLED ARRAYS OF N VARIABLES WHICH RETURN
                THE X,Y,Z COORDINATES OF THE DATA SET RESPECTIVELY.
          N     IS THE NUMBER OF ELEMENTS IN THE DATA SET, INCLUDING
                DUMMY POINTS.
          NPT   IS A CONTROLLED ARRAY CONTAINING (N,07) VARIABLES,
                WHICH RETURNS THE POINTERS OF EACH MAJOR POINT AS
                THEY ARE PROCESSED. N REPRESENTS THE MAJOR NODE.
          DATABK IS THE DD NAME OF THE INPUT DATA SET.
   */
DCL
        (X(N),Y(N),Z(N))CTL BIN FLOAT,
        NPT(N,07)CTL BIN FIXED(31),
        (DATABK) FILE RECORD KEYED DIRECT ENV(REGIONAL(1)),
        1 CARD,
          2 CX DEC FLOAT,
          2 CY DEC FLOAT,
          2 CZ DEC FLOAT,
          2 NODE(7) DEC FIXED(5),
        2 DUMMY CHAR(45),
        IN STATIC DEC FIXED(4),
        (N,I,J,K)BIN FIXED(31),
        KEY PICTURE 'XXXXXXX';
        NPT=0;
DATA:   DO IN = 1 TO N;
        NODE=0.;
        KEY=IN;
        READ FILE(DATABK) INTO (CARD) KEY(KEY);
        X(IN)=CX;
        Y(IN)=CY;
        Z(IN)=CZ;
        DO I = 1 TO 7;
NO:     NPT(IN,I)=NODE(I);
AGAIN:  END;
END:    END DATA;
        END READER1;
```

Figure I.3 EYEBALL, READER1

```
GRSHDE:   PROC OPTIONS(MAIN);

/*
      PURPOSE OF THIS PROGRAM IS TO CONSTRUCT SHADED RELIEF
      MAPS OF THE DATA STORED IN THE IRREGULAR DATA
      STRUCTURE.

      INPUT:

      COL 1- 5 THE NUMBER OF DATA POINTS IN THE INPUT DATA
               SET BOTH REAL AND DUMMY.
      COL 6-10 THE SPACING OF THE RASTER SCAN LINES IN THE
               DATA SET.

                                                          */
   DCL
        (X(*),Y(*),Z(*),XP(*),YP(*),GREY(*),OUTLIN(*)) CTL BIN FLOAT,
        (NPT(*,*),NV1(*),NV2(*),NV3(*),NGRLIN(*),NSORT(*)) CTL BIN
           FIXED(31),
        (BV1(*),BV2(*),BV3(*))CTL BIT(1),
        (N,J,NNN,NC,NTRI,NLINE,NPOI,I,K,KM1,M) BIN FIXED(31),
        (READER1,TRIGLE) ENTRY EXTERNAL,
        SYM(8) CHAR(1) INITIAL('.',':','-','+','O','X','V','M'),
        NOUT(*,*)CTL CHAR(1),
        (DATABK) FILE RECORD KEYED DIRECT ENV(REGIONAL(1));
        GET EDIT(N,CELL)
                (COL(1),F(5,0),F(5,1));
        ALLOCATE X(N),Y(N),Z(N),XP(N),YP(N),GREY(3*N),OUTLIN(N),
                 NPT(N,07),NV1(3*N),NV2(3*N),NV3(3*N),NGRLIN(3*N),
                 NSORT(N),BV1(3*N),BV2(3*N),BV3(3*N);
        CALL READER1(X,Y,Z,NPT,N,DATABK);
        NNN=1;
        CALL MAXMIN(Y(1),YMAX,N,J,NNN);
        CALL MAXMIN(X(1),XMAX,N,J,NNN);
        NNN=0;
        CALL MAXMIN(X(1),XMIN,N,J,NNN);
        CALL MAXMIN(Y(1),YMIN,N,J,NNN);
        PUT PAGE;
        PAG=12.;
        CHAR=PAG*10.;
        NCL=CHAR;
        NCL=NCL+1;
        ALLOCATE NOUT(6,NCL);
        XDIF=XMAX-XMIN;
        XCELL=XDIF/CHAR;
        CALL CENTRE(X(1),Y(1),NC,N);
        CALL TRIGLE(NPT,NV1,NV2,NV3,N,NTRI);
        CALL SETIS(NTRI);
        CALL SHADE(X(1),Y(1),Z(1),NV1(1),NV2(1),NV3(1),N,NTRI,
                   GREY(1));
        NLINE=1;
        WRKLIN=YMAX-CELL/2.;
        NPOI=N;
START:  DO I = NPOI TO 1 BY -1;
        IF Y(NSORT(I))<(WRKLIN-CELL/2.) THEN GO TO GO;
        DO J = 1 TO NTRI;
        IF NV1(J)=NSORT(I) THEN DO;
                           BV1(J)='1';
                           END;
                           ELSE;
        IF NV2(J)=NSORT(I) THEN DO;
                           BV2(J)='1';
                           END;
                           ELSE;
        IF NV3(J)=NSORT(I) THEN DO;
                           BV3(J)='1';
                           END;
                           ELSE;
        IF BV1(J)='1' & BV2(J)='1' & BV3(J)='1' THEN DO;
                           CALL RESET(J);
                           END;
                           ELSE;
        END;
        END;
GO:     NPOI=I+1;
        CALL POINTS(X(1),Y(1),Z(1),NPT(1,1),XMIN,XMAX,WRKLIN,
                    N,XP(1),YP(1),NC,K);
        CALL OUTSID (X(1),Y(1),XP(1),YP(1),NGRLIN(1),K,N,NV1(1),
                     NV2(1),NV3(1),NTRI);
        KM1=K-1;
        DO M = 1 TO KM1;
        OUTLIN(M)=GREY(NGRLIN(M));
        END;
        M1=1;
        M2=2;
        M=1;
        NCOL=1;
        NOUT=' ';
        XM=XMIN+XCELL/2.;
FIRST:  IF XP(M1)>XM THEN DO;
                     XM=XM+XCELL;
                     NOUT(*,NCOL)=' ';
                     NCOL=NCOL+1;
                     GO TO FIRST;
                     END;
         ELSE;
AGAIN:   IF OUTLIN(M1)<=.10 THEN DO;
                NOUT(1,M)=SYM(1);
                GO TO MOVE;
                END;
         IF OUTLIN(M1)<=.20 THEN DO;
                NOUT(1,M)=SYM(2);
                GO TO MOVE;
                END;
         IF OUTLIN(M1)<=.30 THEN DO;
                NOUT(1,M)=SYM(2);
                NOUT(2,M)=SYM(3);
                GO TO MOVE;
                END;
         IF OUTLIN(M1)<=.40 THEN DO;
                NOUT(1,M)=SYM(4);
                GO TO MOVE;
                END;
         IF OUTLIN(M1)<=.50 THEN DO;
```

Figure I.4 GREYSHDE

```
                NOUT(1,M)=SYM(5);
                GO TO MOVE;
                END;
        IF OUTLIN(M1)<=.60 THEN DO;
                NOUT(1,M)=SYM(5);
                NOUT(2,M)=SYM(3);
                GO TO MOVE;
                END;
        IF OUTLIN(M1)<=.70 THEN DO;
                NOUT(1,M)=SYM(5);
                NOUT(2,M)=SYM(3);
                NOUT(3,M)=SYM(2);
                GO TO MOVE;
                END;
        IF OUTLIN(M1)<=.80 THEN DO;
                NOUT(1,M)=SYM(5);
                NOUT(2,M)=SYM(6);
                GO TO MOVE;
                END;
        IF OUTLIN(M1)<=.90 THEN DO;
                NOUT(1,M)=SYM(5);
                NOUT(2,M)=SYM(6);
                NOUT(3,M)=SYM(7);
                GO TO MOVE;
                END;
        IF OUTLIN(M1)> .90 THEN DO;
                NOUT(1,M)=SYM(5);
                NOUT(2,M)=SYM(6);
                NOUT(3,M)=SYM(7);
                NOUT(4,M)=SYM(8);
                NOUT(5,M)=SYM(2);
                NOUT(6,M)=SYM(3);
                GO TO MOVE;
                END;
MOVE:     XM=XM+XCELL; M=M+1;
          IF XP(M2)<XM THEN GO TO NEXTP;
          NOUT(*,M)=NOUT(*,M-1);
          GO TO MOVE;
NEXTP:    M1=M1+1;
          M2=M2+1;
          IF M2>K THEN GO TO OUTPUT;
          IF XP(M2)<XM THEN GO TO NEXTP;
          GO TO AGAIN;
OUTPUT:   DO J = 1 TO 6;
          PUT SKIP(0) EDIT((NOUT(J,LN) DO LN = 1 TO NCL))
                      (COL(1),(NCL) (A));
          END;
NEXCL:    KK=KK+1;
          PUT SKIP(1);
        WRKLIN=WRKLIN-CELL;
        IF WRKLIN>=YMIN THEN GO TO START;
        ELSE;
        END GRSHDE;
```

Figure I.5 GREYSHDE Continued

```
TRIGLE: PROC(NPT,NV1,NV2,NV3,N,NTRI);

/*
     THE PURPOSE OF THIS SUBROUTINE IS TO LOAD THE
     VERTICES OF THE TRIANGLES INTO THREE ONE DIMENSIONAL
     ARRAYS. THE ORDER OF TRIANGLE PROCESSING IS
     SEQUENTIAL. THE THREE VERTICES OF THE JTH TRIANGLE
     ENCOUNTERED WILL LEAVE TRIGLE LOCATED IN THE JTH
     POSITION OF THE ARRAYS NV1, NV2, NV3 RESPECTIVELY.

     PARAMETERS:

         NPT   AN (N,07) ARRAY CONTAINING THE POINTERS
               OF THE DATA STRUCTURE UPON ENTRY INTO
               TRIGLE.

         NV1   ARRAYS OF LENGTH 3*N CONTAINING THE
         NV2   VERTICES OF ALL TRIANGLES UPON
         NV3   LEAVING TRIGLE.

         N     THE NUMBER OF REAL AND DUMMY POINTS
               IN THE DATA STRUCTURE.

         NTRI  THE NUMBER OF TRIANGLES PROCESSED BY
               TRIGLE.

                                                    */
    DCL
       NPT(N,07) CTL BIN FIXED (31),
       (NV1(3*N),NV2(3*N),NV3(3*N))CTL BIN FIXED(31),
       TRIP BIT(1),
       (N,I,J,K,II,KK,ILAST,NTRI) BIN FIXED (31);
       II=1;
       I=1;
       K=1;
START: J=1;
       TRIP='0';
       KK=I;
LOOK:  IF (NPT(I,J)=32000) THEN GO TO NEXTP;
       IF (NPT(I,J)=0)     THEN GO TO NEXTR;
       IF (NPT(I,J)<0)     THEN DO;
                            ILAST=ABS(NPT(I,J));
                            NLAST=NPT(I,J-1);
                            GO TO KICKI;  END;
       IF (NPT(I,J)<II)    THEN GO TO NEXTP;
       IF TRIP='1' THEN GO TO REC;
       IF (NPT(I,J+1)=32000)THEN GO TO NEXTP2;
       IF (NPT(I,J+1)=0)    THEN GO TO NEXTR;
       IF (NPT(I,J+1)<0)    THEN DO;
                            ILAST=ABS(NPT(I,J+1));
                                 NLAST=NPT(I,J);
                                 GO TO KICKI;
                                 END;
       IF (NPT(I,J+1)<II)   THEN GO TO NEXTP2;
REC:   NV1(K)=II;
       NV2(K)=NPT(I,J);
       IF TRIP='1' THEN DO;
                        NV3(K)=NLAST;
                        GO TO UP;
                        END;
       NV3(K)=NPT(I,J+1);
UP:    K=K+1;
       IF TRIP='1' THEN TRIP='0';
       J=J+1;
       IF J>7 THEN GO TO NEXTR;
       GO TO LOOK;
NEXTP: J=J+1;
       IF J>7 THEN GO TO NEXTR;
       GO TO LOOK;
KICKI: IF KK=I  THEN DO;
                     I=ILAST;
                     IF NLAST>II THEN TRIP='1';
                     J=1;
                     GO TO LOOK;
                     END;
                     ELSE DO;
                     I=KK+1;
                     II=I;
                     GO TO START;
                     END;
NEXTP2:
       J=J+2;
       IF J>7 THEN GO TO NEXTR;
       GO TO LOOK;
NEXTR: II=II+1;
       I=I+1;
       IF II>N THEN GO TO OVER;
       GO TO START;
OVER:  NTRI=K-1;
       END TRIGLE;
```

Figure I.6 TRIGLE

```
C***** SCHNET
C
C       THE PURPOSE OF THIS SUBROUTINE IS TO FOLLOW A PATH THROUGH
C       THE IRREGULAR DATA STRUCTURE. GIVEN A CRITERIA, SUCH AS,
C       A PATH OF EQUAL HEIGHT (ISOLINE), FIND THE X,Y COORDINATES
C       OF THIS PATH THROUGH THE STRUCTURE.
C
C   PARAMETERS:
C       X       N LENGTH ARRAY OF X COORDINATES OF DATA POINTS
C       Y       N LENGTH ARRAY OF Y COORDINATES OF DATA POINTS
C       Z       N LENGTH ARRAY OF Z COORDINATES OF DATA POINTS
C       NPT     A 7 BY N ARRAY OF POINTERS  OF THE DATA SET
C       RP      THE Z VALUE OF A NODE WHICH IS AT ONE END OF AN EDGE
C               CUT BY THE ISOLINE TO BE FOLLOWED. THIS SIMPLY GIVES
C               SCHNET A PLACE TO START IN THE DATA SET.
C       CONT    THE VALUE OF THE ISOLINE BEING FOLLOWED
C       NR      THE NODE NUMBER OF THE NODE FOR WHICH RP IS THE Z VALUE
C       N       NUMBER OF POINTS IN THE DATA SET-BOTH REAL AND DUMMY
C       X1      THE X COORDINATE OF ONE END OF THE STRAIGHT LINE FOR
C               WHEN NRIT=1
C       Y1      THE Y COORDINATE OF ONE END OF THE STRAIGHT LINE FOR
C               WHEN NRIT=1
C       X2      THE X COORDINATE OF THE OTHER END OF THE STRAIGHT LINE
C               FOR WHEN NRIT=1
C       Y2      THE Y COORDINATE OF THE OTHER END OF THE STRAIGHT LINE
C               FOR WHEN NRIT=1
C       NRIT    THE CRITERIA OF THE PATH BEING FOLLOWED
C               NRIT=0 TELLS SCHNET THAT AN ISOLINE IS TO BE FOLLOWED
C                      AND THE Z VALUES ARE TO BE COMPARED
C               NRIT=1 TELLS SCHNET THAT A STRAIGHT LINE IS TO BE
C                      FOLLOWED AND THE X,Y VALUES ARE TO BE COMPARED
C               NRIT=2 TELLS SCHNET THAT THE USER IS PASSING IN A
C                      STRAIGHT LINE-BUT WISHES TO KNOW ONLY THE
C                      NODE NUMBER WHICH IS CLOSEST TO ONE END OF THE
C                      LINE (X2,Y2)
C       NCALL TELLS SCHNET HOW MANY TIMES IT HAS BEEN CALLED
C       L1      THE NODE NUMBER CLOSEST TO X2,Y2 FOR NRIT=2
C       L2      THE NODE NUMBER SECOND CLOSEST TO X2,Y2 FOR NRIT=2
C       XFACT THE FACTOR BY WHICH THE WHOLE DATA SET MUST BE
C               MULTIPLIED IN ORDER TO FIT ON TEN INCH PLOTTER
C               PAPER
C       LET     IF EQUAL TO 0-NO CONTOUR VALUES WILL APPEAR ON THE
C               PLOTTED OUTPUT
C               IF NOT EQUAL TO 0-THE VALUES OF THE PLOTTED CONTOURS
C               WILL BE ON THE OUTPUT AT REGULAR INTERVALS


      SUBROUTINE SCHNET(X,Y,Z,NPT,RP,CONT,NR,N,X1,X2,Y1,Y2,NRIT,NCALL
     1,L1,L2,XFACT,LET)
      DIMENSION X(N),Y(N),Z(N),NPT(07,N),PX(200),PY(200),XN(200)
      COMMON/PLT/ZN(200),YN(200),NNXN,AA(200)
      COMMON/ANGLE/ LLAB,LAB,TEST,MUL,CAL
      LOGICAL INFLAG,OTFLAG,BORDER,THRU,BORD,NBORD,NIXO,LLAB,TEST,CAL
      KL=-1
      IF(NCALL.EQ.0) ITEMP=ISON(N,N,KL)
      NBOR=0
      KK=NR
      EPS=.01
      MUL=1
      CAL=.FALSE.
      BORD=.FALSE.
      NBORD=.FALSE.
      OTFLAG=.FALSE.
      LLAB=.FALSE.
      TEST=.FALSE.
      K=1
      J=1
      NCIR=0
C  IN WORKING WITH A STRAIGHT LINE (I.E. NRIT=1) THE EDGE CLOSEST
C  TO ONE END OF THE LINE MUST BE PASSED TO SCHNET.
C  THIS EDGE CAN BE FOUND WITH A SUBSEQUENT CALL TO SCHNET WITH
C  NRIT=2
      IF(NRIT.NE.1) GO TO 31
      NBOR=1
      NQ=L1
      GO TO 6
C FIND THE FIRST POINTER IN THE LIST OF NR WHICH IS NOT TO A BORDER
31    NP=NPT(J,NR)
      IF(NP.NE.32000) GO TO 310
      BORD=.TRUE.
      GO TO 33
310   IF(NP.EQ.0) GO TO 999
      IF(NP.GT.0) GO TO 3101
      NR=IABS(NP)
      IF(NR.EQ.KK)GO TO 999
      J=1
      GO TO 31
3101  KL=0
      ITEMP=ISON(NR,NP,KL)
      IF(NRIT.NE.0) GO TO 32
      IF    (Z(NP).LE.CONT) GO TO 32
      IF(NP.EQ.NR) GO TO 999
33    J=J+MUL
      IF(J.GT.7) GO TO 999
      GO TO 31
32    IF(ITEMP.EQ.0) GO TO 9
      GO TO 999
6     CALL FINDIT(J,NR,NQ,N,NPT)
      NCIR=0
65    KL=0
```

Figure I.7 SCHNET

```
      ITEMP=ISON(NR,NPT(J,NR),KL)
      IF(ITEMP.NE.0) GO TO 99
9     INFLAG=.FALSE.
      BORDER=.FALSE.
C EACH NEW POINTER FROM A MAJOR NODE IS STARTED HERE
7     CONTINUE
72    IF(J.GT.7.OR.J.LT.1)J=1
      NP=NPT(J,NR)
      IF(NP.NE.32000) GO TO 71
C WHAT TO DO IF THE LINE BEING FOLLOWED RUNS INTO A BORDER
      IF(NRIT.EQ.0)GO TO 720
      IF(NIXG.EQ.1)GO TO 2
      IF(NR.NE.KK.AND.K.GT.2) GO TO 999
      IF(NBOR.EQ.0)GO TO 9930
      GO TO 999
720   BORDER=.TRUE.
      IF(NBOR.NE.0) GO TO 991
      GO TO 993
C WHAT IF NR RUNS OUT OF POINTERS IN ITS LIST?
71    IF(NP.EQ.0)GO TO 99
      IF(NP.GT.0)GO TO 74
      IF(.NOT.LLAB) GO TO 7111
      LLAB=.FALSE.
      GO TO 7112
7111  LLAB=.TRUE.
7112  LAB=NR
      TEST=.TRUE.
      NR=IABS(NP)
      J=1
      GO TO 7
74    SP=Z(NP)
      NP2=NP
C THE PROGRAM SPLITS HERE FOR CONTOURS AND FOR STRAIGHT LINES
      IF (NRIT.NE.0) GO TO 21
      IF(RP.NE.CONT)  GO TO 1
      IF(K.GT.1) GO TO 5
      NR=NR1
      NP=NP1
5     XN(K)=X(NR)
      YN(K)=Y(NR)
      KL=1
      ITEMP=ISON(NR,NP,KL)
55    K=K+1
      GO TO 4
1     OTFLAG=.FALSE.
8     IF(SP.GT.CONT) GO TO 2
C IF THERE IS A CONTOUR BETWEEN NR AND NP THEN CALCULATE THE
C INTERSECTION POINT OF IT WITH THE TRIANGLE EDGE
      IF(K.GT.1) GO TO 81
      NR1=NR
      NP1=NP
81    RATIO=(RP-CONT)/(RP-SP)
      XN(K)=X(NR)+(X(NP)-X(NR))*RATIO
      YN(K)=Y(NR)+(Y(NP)-Y(NR))*RATIO
      KL=1
      ITEMP=ISON(NR,NP,KL)
85    K=K+1
      INFLAG=.TRUE.
      IF(K.LE.200) GO TO 3
      THRU=.TRUE.
      GO TO 15
C THIS PORTION FINDS OUT IF THERE IS AN INTERSECTION BETWEEN THE
C STRAIGHT LINE AND THE EDGE BETWEEN NR AND NP, AND CALCULATES IT IF
C THERE IS ONE
21    OTFLAG=.FALSE.
      BORD=.FALSE.
      NIXO=.FALSE.
      CALL CROSS(X(NR),Y(NR),X(NP),Y(NP),X1,Y1,X2,Y2,A,B,NIX)
      NIXG=NIX
      IF(NIX.EQ.1) GO TO 2
      KL=0
      ITEMP=ISON(NR,NP,KL)
      IF(NIX.EQ.3.AND.ITEMP.EQ.1)GO TO 999
      IF(NIX.EQ.4) NBORD=.TRUE.
      IF(NRIT.EQ.1) GO TO 201
      L1=NR
      L2=NP
      IF(NIX.EQ.2) GO TO 250
      GO TO 204
201   IF(NIX.EQ.4)GO TO 204
      IF(NIX.EQ.2) GO TO 250
C FOR A SIMPLE INTERSECTION OF TWO LINES CONTINUE
      CAL=.FALSE.
      YN(K)=B
      AA(K)=A
      DIST1=X(NP)-X(NR)
      DIST2=Y(NP)-Y(NR)
      DEL=SQRT(DIST1*DIST1+DIST2*DIST2)
      DIST1=A-X(NR)
      DIST2=B-Y(NR)
      DEL1=SQRT(DIST1*DIST1+DIST2*DIST2)
      ZN(K)=Z(NR)-(DEL1*(Z(NR)-Z(NP)))/DEL
      GO TO 208
C IF ONE LINE OVERLAPS OR LIES ON TOP OF THE OTHER THEN CONTINUE
204   IF(NRIT.NE.1) GO TO 208
211   YN(K)=Y(NR)
      ZN(K)=Z(NR)
      K=K+1
      YN(K)=Y(NP)
      ZN(K)=Z(NP)
```

Figure I.8 SCHNET Continued

```
      INFLAG=.TRUE.
208   K=K+1
213   KL=1
      ITEMP=ISON(NR,NP,KL)
      IF (NIX.NE.4) GO TO 205
C FIND OUT IF ONE OF THE OVERLAPPING LINES IS A BORDER
      IF(NPT(J-MUL,NR).EQ.32000) GO TO 203
      IF(NPT(J+MUL,NR).NE.32000) GO TO 205
C FIND OUT IF EITHER OF THE ENDPOINTS FOR WHICH WE ARE SEARCHING
C IS BETWEEN NR AND NP
203   X5=X2+100.
      Y5=Y2
2075  CALL CROSS(X(NR),Y(NR),X(NP),Y(NP),X2,Y2,X5,Y5,A,B,NIX)
      IF(NIX.NE.4) GO TO 2065
      X5=X2
      Y5=Y2+100.
      GO TO 2075
2065  IF(NIX.EQ.2) GO TO 999
C FIND OUT IN WHICH DIRECTION WE ARE MOVING ALONG THE LINE WE ARE
C TRYING TO FOLLOW
2050  IF(ABS(X(NR)-X2).LE.EPS) GO TO 173
      IF(ABS(X2-X(NR)).LT.ABS(X2-X(NP)))GO TO 170
      NQ=NR
      NR=NP
      J=2
      IF(NPT(2,NR).NE.32000) GO TO 171
      J=3
      IF(NPT(3,NR).EQ.NQ)J=1
171   GO TO 72
170   J=J+1
      IF(J.GT.7)J=1
      IF(NPT(J,NR).EQ.32000)GO TO 170
      GO TO 72
173   IF(ABS(Y2-Y(NR)).LT.ABS(Y2-Y(NP))) GO TO 172
      NQ=NR
      NR=NP
      J=2
      IF(NPT(2,NR).NE.32000) GO TO 174
      J=3
      IF(NPT(3,NR).EQ.NQ)J=1
174   GO TO 72
172   J=J+1
      IF(J.GT.7)J=1
      IF(NPT(J,NR).EQ.32000) GO TO 172
      GO TO 72
205   INFLAG=.TRUE.
      GO TO 3
C IF THE ENDPOINTS OF THE LINES HAPPEN TO COINCIDE OR IF ONE OF THE
C POINTS LIES ON THE OTHER LINE BETWEEN ITS ENDPOINTS THEN CONTINUE
250   YN(K)=B
      AA(K)=A
      IF(ABS(B-Y(NR)).GT.EPS)GO TO 251
      IF(ABS(A-X(NR)).GT.EPS)GO TO 251
      ZN(K)=Z(NR)
      GO TO 2510
251   IF(ABS(B-Y(NP)).GT.EPS)GO TO 252
      IF(ABS(A-X(NP)).GT.EPS)GO TO 252
      ZN(K)=Z(NP)
2510  IF(ABS(B-Y1).GT.EPS)   GO TO 2601
      IF(ABS(A-X1).GT.EPS)   GO TO 2601
2511  NIXO=.TRUE.
      IF(CAL) GO TO 260
      IF(K.GT.2) GO TO 260
      CALL ANGEND(X1,X2,Y1,Y2,X,Y,NPT,N,NR,J)
      NP=NPT(J,NR)
      GO TO 69
2601  IF(ABS(B-Y2).GT.EPS)   GO TO 260
      IF(ABS(A-X2).GT.EPS)   GO TO 260
      GO TO 2511
252   IF(ABS(B-Y1).GT.EPS) GO TO 253
      IF(ABS(A-X1).GT.EPS)GO TO 253
      NIXO=.TRUE.
      DIST1=X(NP)-X(NR)
      DIST2=Y(NP)-Y(NR)
      DEL=SQRT(DIST1*DIST1+DIST2*DIST2)
      DIST1=A-X(NR)
      DIST2=B-Y(NR)
      DEL1=SQRT(DIST1*DIST1+DIST2*DIST2)
      ZN(K)=Z(NR)-(DEL1*(Z(NR)-Z(NP)))/DEL
      GO TO 260
253   IF(ABS(B-Y2).GT.EPS)GO TO 254
      IF(ABS(A-X2).GT.EPS)GO TO 254
      NIXO=.FALSE.
      DIST1=X(NP)-X(NR)
      DIST2=Y(NP)-Y(NR)
      DEL=SQRT(DIST1*DIST1+DIST2*DIST2)
      DIST1=A-X(NR)
      DIST2=B-Y(NR)
      DEL1=SQRT(DIST1*DIST1+DIST2*DIST2)
      ZN(K)=Z(NR)-(DEL1*(Z(NR)-Z(NP)))/DEL
      GO TO 260
254   CONTINUE
260   KL=1
      ITEMP=ISON(NR,NP,KL)
      K=K+1
      INFLAG=.TRUE.
      GO TO 3
C MOVE THE POINTER OVER ONE POSITION IN NR'S LIST, WHERE THE DIRECTION
C DEPENDS ON THE VALUE OF MUL (I.E. + OR - )
2     OTFLAG=.TRUE.
```

Figure I.9 SCHNET Continued

```
3       J=J+MUL
        IF(.NOT.BORDER) GO TO 35
        IF(NBOR.EQ.0) GO TO 993
35      CONTINUE
        IF(INFLAG.AND.OTFLAG) GO TO 4
        NNN=0
        IF(J.LE.0) CALL FINDIT(J,NR,NNN,N,NPT)
        GO TO 7
4       IF(K.GE.200) GO TO 10
C IF THE FOLLOWING TWO IF STATEMENTS ARE TRUE THEN THE CONTOUR IS A
C CONTINUOUS LINE AND HAS NOW BEEN CONNECTED TO ITS STARTING POINT
C AND SO WE CAN PLOT THE VALUES
        IF(NR.NE.NR1)GO TO 69
        IF(NP.NE.NP1)GO TO 69
        IF(NRIT.NE.0) GO TO 69
        IF(K.LT.2) GO TO 69
        GO TO 10
69      RP=SP
        IF(.NOT.LLAB) GO TO 6910
        IF(.NOT.TEST) GO TO 6910
        TEST=.FALSE.
        LLAB=.FALSE.
        NQ=LAB
        NR=NP
        GO TO 6
6910    NQ=NR
        NR=NP
        GO TO 6
10      XN(K)=XN(1)
        THRU=.FALSE.
        YN(K)=YN(1)
15      IF(THRU) K=K-1
        IF(NRIT.NE.0) GO TO 999
        CALL MARKIT(XN,YN,K,LSIG,XFACT,CONT,LET)
        IF(K.LT.200) GO TO 999
        K=1
        GO TO 69
99      IF(NRIT.NE.0) GO TO 9917
        IF(NBOR.GT.0) GO TO 991
        IF(BORDER)GO TO 993
        IF (.NOT.BORD)GO TO 98
        THRU=.TRUE.
        GO TO 15
98      IF(NR.EQ.KK) GO TO 10
        IF(NP.NE.0) GO TO 996
        IF(NCIR.GT.0) GO TO 996
        NCIR=1
9917    J=1
        GO TO 7
996     THRU=.TRUE.
        IF(K.GT.1) GO TO 15
        GO TO 999
993     IF(K.LT.2) GO TO 999
C IF A BORDER HAS BEEN REACHED THEN WE MUST SAVE THE VALUES OF ALL
C INTERSECTIONS AND GO BACK TO THE NR AT WHICH WE ENTERED THE DATA SET
        KM1=K-1
        KM2=K-2
        DO 992 I=1,KM2
9935    PX(I)=XN(KM1)
        PY(I)=YN(KM1)
992     KM1=KM1-1
        INE=K-1
        RP=Z(NR1)
        MUL=-1
9921    NQ=NP1
        NR=NR1
        TEST=.FALSE.
        LLAB=.FALSE.
        K=2
        NBOR=1
        CALL FINDIT(J,NR,NQ,N,NPT)
        IF(J.GT.1) GO TO 3
        NNN=0
        CALL FINDIT(J,NR,NNN,N,NPT)
        GO TO 7
9930    KL=-1
        ITEMP=ISON(N,N,KL)
        NBOR=1
        INFLAG=.FALSE.
        K=1
        IF(.NOT.LLAB) GO TO 881
        TEST=.FALSE.
        LLAB=.FALSE.
        NQ=LAB
        NR=NP2
        CALL FINDIT(J,NR,NQ,N,NPT)
        GO TO 7
881     NQ=NR
        NR=NP2
        CALL FINDIT(J,NR,NQ,N,NPT)
        GO TO 7
991     KM1=K-1
C ONCE A BORDER HAS BEEN REACHED FOR THE SECOND TIME IN ONE RUN WE
C KNOW THAT THE PARTICULAR CONTOUR WE HAVE BEEN WORKING ON HAS
C EXITED FROM THE DATA SET AND WE CAN THEREFORE CONCATENATE THE TWO
C GROUPS IN EITHER DIRECTION FROM THE ORIGINAL NR AND PLOT THE
C RESULTANT VECTORS OF VALUES.
        DO 995 I=1,KM1
        PX(INE)=XN(I)
        PY(INE)=YN(I)
```

Figure I.10 SCHNET Continued

```
995     INE=INE+1
        INE=INE-1
        CALL MARKIT(PX,PY,INE,LSIG,XFACT,CONT,LET)
999     NCALL=1
        IF(NRIT.EQ.0) GO TO 1000
        NNXN=K-1
        IF(NNXN.LT.1) GO TO 3
        IF(NNXN.NE.1) GO TO 1000
        IF(NRIT.NE.1) GO TO 1000
        NP=NP2
        GO TO 69
1000    RETURN
        END
```

Figure I.11 SCHNET Continued

```
C***** CONVER
C
C       THE PURPOSE OF THIS SUBROUTINE IS TO CALCULATE
C       AND PLOT THE HORIZONS OF EYEBALL. ACTUALLY
C       CONVER SIMPLY SETS UP THE CALCULATIONS, WHICH
C       ARE THEN PERFORMED BY VARIOUS OTHER SUBROUTINES.
C       CONVER IS NECESSARY SINCE THE DATA COMES IN TO
C       THE SYSTEM IN PL-1 AND THE SUBROUTINES ARE ALL
C       IN FORTRAN.
C
C       PARAMETERS:
C
C       X,Y,Z THE COORDINATES OF THE IRREGULAR DATA
C             STRUCTURE STORED IN N LONG ARRAYS.
C       NPT   THE POINTERS OF THE DATA STRUCTURE
C             STORED IN A 7 BY N ARRAY.
C       N     THE TOTAL NUMBER OF POINTS IN THE DATA
C             STRUCTURE, BOTH REAL AND DUMMY.
C       NPROJ THE TYPE OF PROJECTION DESIRED. SEE EYEBALL
C             FOR DETAILS.
C
C
      SUBROUTINE CONVER (X,Y,Z,NPT,N,NPROJ)
      COMMON/RDGLIN/XRIG(200),YRIG(200),LR
      COMMON/PLT/ZN(200),YN(200),NNXN,XN(200)
      COMMON/DSTPLN/D
      COMMON/PLTDTL/IPROJ,WIDE,BRDDRW,CRSECT,SECTOR,SLICE
      DIMENSION X(N),Y(N),Z(N),VPT(3),CPT(3),BX(10),BY(10),NPT(07,N),
     1XB(4),YB(4),PX(4),PY(4),PZ(4),VPN(3),CPN(3),DX(200),DY(200),
     2NN(200)
      LOGICAL WIDE,BRDDRW,CRSECT,SECTOR,SLICE,END1,END2
      EPS=.005
      EPS2=10.
      CRSECT=.FALSE.
      WIDE=.FALSE.
      END1=.TRUE.
      END2=.TRUE.
      IPROJ=NPROJ
      READ(5,100)(VPT(I),I=1,3)
      READ(5,100)(CPT(I),I=1,3)
      READ(5,101) CELLS
100   FORMAT(3F10.0)
101   FORMAT(F5.2)
      CALL CENTRE(X,Y,NC,N)
C
C  THIS SECTION SETS UP THE BORDER
C
      NNN=0
      CALL MAXMIN(X,XB(1),N,J,NNN)
      NNN=1
      CALL MAXMIN(X,XB(2),N,J,NNN)
      NNN=0
      CALL MAXMIN(Y,YB(1),N,J,NNN)
      NNN=1
      CALL MAXMIN(Y,YB(3),N,J,NNN)
      XB(4)=XB(1)
      XB(3)=XB(2)
      YB(2)=YB(1)
      YB(4)=YB(3)
      LL=4
      LN=0
      DO 22 LVP=1,3
      VPN(LVP)=VPT(LVP)
22    CPN(LVP)=CPT(LVP)
      CALL ROTATE(XB,YB,LL,VPN,CPN,LN)
      AX=10.
      AY=5.
      IPEN=-3
      CALL PLOT(AX,AY,IPEN)
      FAC=.25
      CALL FACTOR(FAC)
      LN=1
      CALL ROTATE(X,Y,N,VPT,CPT,LN)
      NNN=0
      CALL MAXMIN(X,XMIN,N,J,NNN)
      PX(1)=X(J)
      PY(1)=Y(J)
      PZ(1)=Z(J)
      NNN=1
      CALL MAXMIN(X,XMAX,N,J,NNN)
      PX(2)=X(J)
      PY(2)=Y(J)
      PZ(2)=Z(J)
      NNN=0
      CALL MAXMIN(Y,YMIN,N,J,NNN)
      PX(3)=X(J)
      PY(3)=Y(J)
      PZ(3)=Z(J)
      NNN=1
      CALL MAXMIN(Y,YMAX,N,J,NNN)
      PX(4)=X(J)
      PY(4)=Y(J)
      PZ(4)=Z(J)
      NNN=4
      NSEC=1
      CALL PRJN1(VPT,CPT,PX,PY,PZ,NNN,NSEC,NNN,NSEC)
      DMY1=ABS(PY(2)-PY(1))
      DMY2=ABS(PY(4)-PY(3))
      DMX1=ABS(PX(2)-PX(1))
      DMX2=ABS(PX(4)-PX(3))
```

Figure I.12 CONVER

```
      AMA=MAX1(DMY1,DMY2,DMX1,DMX2)
      FAC=10.0/AMA
      CALL FACTOR(FAC)
C
CROSS WITH BORDER
C
       WKPT=XMIN
       HALFO=CPT(1)/2.
       NCOU=0
       IF(WKPT.LT.HALFO) WKPT=HALFO
2      CALL CROSS(WKPT,YMAX,WKPT,YMIN,XB(1),YB(1),XB(2),YB(2),XC,YC,NIX)
       NCOU=NCOU+1
       ND=NC
       K=0
       IF(NIX.EQ.1) GO TO 10
       IF(NIX.NE.4) GO TO 5
       GO TO 10
5      K=K+1
       BX(K)=XC
       BY(K)=YC
10     CALL CROSS(WKPT,YMAX,WKPT,YMIN,XB(2),YB(2),XB(3),YB(3),XC,YC,NIX)
       IF(NIX.EQ.1) GO TO 12
       IF(NIX.NE.4) GO TO 13
       GO TO 12
13     K=K+1
       BX(K)=XC
       BY(K)=YC
12     CALL CROSS(WKPT,YMAX,WKPT,YMIN,XB(3),YB(3),XB(4),YB(4),XC,YC,NIX)
       IF(NIX.EQ.1)GO TO 14
       IF(NIX.NE.4)GO TO 15
       GO TO 14
15     K=K+1
       BX(K)=XC
       BY(K)=YC
14     CALL CROSS(WKPT,YMAX,WKPT,YMIN,XB(4),YB(4),XB(1),YB(1),XC,YC,NIX)
       IF(NIX.NE.1) GO TO 145
       IF(K.EQ.0) GO TO 16
       GO TO 175
145    IF(NIX.NE.4) GO TO 17
       GO TO 175
17     K=K+1
       BX(K)=XC
       BY(K)=YC
175    NRIT=2
       NCALL=0
       NBOR=4
       IF(K.LE.2) GO TO 1753
       CALL STRIP(BX,BY,NBOR)
1753   IF(WKPT.EQ.XMIN) GO TO 7512
       IF(WKPT.EQ.XMAX) GO TO 7512
       AA=AMIN1(BY(1),BY(2))
       IF(AA.NE.BY(1)) GO TO 7511
       BY(1)=BY(1)-EPS2
       BY(2)=BY(2)+EPS2
       GO TO 7512
7511   BY(1)=BY(1)+EPS2
       BY(2)=BY(2)-EPS2
7512   CALL SCHNET(X,Y,Z,NPT,RP,CO,ND,N,X(ND),BX(1),Y(ND),BY(1),
     1NRIT,NCALL,L1,L2,XFO,LET)
       NCALL=0
       NRIT=1
       IF(ABS(BX(1)-BX(2)).GT.EPS) GO TO 751
       IF(ABS(BY(1)-BY(2)).GT.EPS) GO TO 751
       NNXN=1
       DO 749 I=1,N
       IF(ABS(X(I)-BX(1)).GT.EPS) GO TO 749
       IF(ABS(Y(I)-BY(1)).GT.EPS) GO TO 749
       ZN(1)=Z(I)
749    CONTINUE
       XN(1)=BX(1)
       YN(1)=BY(1)
       GO TO 97
751    CALL SCHNET(X,Y,Z,NPT,RP,C,L2,N,BX(1),BX(2),BY(1),BY(2),NRIT,
     1NCALL,L1,L2,XFO,LET)
99     DO 32 I=1,NNXN
32     XN(I)=WKPT
97     NDIM=200
98     NSEC=1
       CALL PRJN1(VPT,CPT,XN,YN,ZN,NNXN,NSEC,NDIM,NSEC)
       CALL STRIP(XN,YN,NNXN)
       CE=.75
       NNNN=NNXN
       CALL SMTHLN(XN,YN,NNNN,CE,LSIG)
       IF(NCOU.GT.1)GO TO 30
       IF(NNXN.NE.1) GO TO 9013
       XRIG(1)=XN(1)
       XRIG(2)=XN(1)
       YRIG(1)=YN(1)
       YRIG(2)=YN(1)
       GO TO 3111
9013   DO 31 I=1,NNXN
       XRIG(I)=XN(I)
31     YRIG(I)=YN(I)
3111   LR=NNXN
       ED1X=XN(1)
       ED1Y=YN(1)
       ED2X=XN(NNXN)
       ED2Y=YN(NNXN)
       GO TO 35
30     CALL RIDGE(XN,YN,ZN,NNXN)
```

Figure I.13 CONVER Continued

```
      IF(NCOU.NE.2) GO TO 372
      CALL SORTER(XRIG,NN,LR)
      DO 371 I=1,LR
      DX(I)=XRIG(NN(I))
371   DY(I)=YRIG(NN(I))
      DO 373 I=1,LR
      XRIG(I)=DX(I)
373   YRIG(I)=DY(I)
372   CONTINUE
      IF(ED1X.EQ.XRIG(1).AND.ED1Y.EQ.YRIG(1)) GO TO 3496
      IF(END1) GO TO 3493
      END1=.TRUE.
      ED1X=XRIG(1)
      ED1Y=YRIG(1)
      GO TO 3497
3493  IPEN=3
      CALL PLOT(ED1X,ED1Y,IPEN)
      IPEN=2
      CALL PLOT(XRIG(1),YRIG(1),IPEN)
      ED1X=XRIG(1)
      ED1Y=YRIG(1)
      GO TO 3497
3496  END1=.FALSE.
3497  CALL WORK2(XRIG,YRIG,LR,LSIG)
      IF(ED2X.EQ.XRIG(LR).AND.ED2Y.EQ.YRIG(LR)) GO TO 3498
      IF(END2) GO TO 3495
      END2=.TRUE.
      ED2X=XRIG(LR)
      ED2Y=YRIG(LR)
      GO TO 35
3495  IPEN=2
      CALL PLOT(ED2X,ED2Y,IPEN)
      ED2X=XRIG(LR)
      ED2Y=YRIG(LR)
      GO TO 35
3498  END2=.FALSE.
35    WKPT=WKPT+CELLS
      IF(WKPT.LE.XMAX) GO TO 2
16     RETURN
       END
```

Figure I.14 CONVER Continued

```
C***** ANGFND
C
C       THE PURPOSE OF THIS SUBROUTINE IS TO FIND THE EDGE OF THE
C       DATA SET WHICH IS CLOSEST TO A GIVEN LINE.
C       THIS SUBROUTINE IS USED WHEN A LINE IS PASSED TO SCHNET
C       WHICH PASSES DIRECTLY THROUGH A DATA POINT. OBVIOUSLY
C       IF IT IS ONLY KNOWN THAT THE LINE IS STRAIGHT THEN IT
C       IS IMPOSSIBLE TO KNOW WHICH EDGES OF THE TRIANGULAR NETWORK
C       THE LINE EXITS THROUGH, SINCE ALL TRIANGLES OF THIS GROUP
C       HAVE THIS ONE DATA POINT AS A VERTICE OF EACH.
C       THUS, WHAT THIS SUBROUTINE DOES IS TO FIND THE ANGULARLY
C       ADJACENT EDGE TO THE LINE IN QUESTION.
C
C       PARAMETERS:
C
C       X1,Y1,X2,Y2 THE X-Y COORDINATES OF THE ENDPOINTS OF THE LINE
C                   IN QUESTION.
C       X,Y         ARE VECTORS OF LENGTH N CONTAINING THE X-Y
C                   COORDINATES OF THE DATA SET.
C       NPT         A 7 BY N ARRAY OF THE POINTERS OF THE DATA SET.
C       N           THE NUMBER OF POINTS IN THE DATA SET.
C       NR          THE NUMBER OF THE DATA POINT WHICH IS BEING
C                   PASSED THROUGH.
C       NAN         THE POSITION OF THE ADJACENT POINTER IN NR'S LIST
C                   THIS IS WHAT ANGFND RETURNS.
C
      SUBROUTINE ANGFND(X1,X2,Y1,Y2,X,Y,NPT,N,NR,NAN)
      DIMENSION X(N),Y(N),NPT(7,N)
      DATA PI/3.141593/
      COMMON/ANGLE/ LLAB,LAB,TEST,MUL,CAL
      LOGICAL LLAB,TEST,CAL,ANGLL,ANGTE
      NAN=0
      LANG=LAB
      ANGLL=LLAB
      ANGTE=TEST
      CAL=.TRUE.
      PIB2=PI/2.
      PI2=2.*PI
      A1=X2-X1
      A2=Y2-Y1
      TANG=ATAN2(A1,A2)*(-1.)+PIB2
      IF(TANG.LT.0) TANG=PI2+TANG
      JT=0
      J=1
      L=NR
      A=X(NR)
      B=Y(NR)
5     NP=NPT(J,NR)
      IF(NP.EQ.32000)GO TO 2
      IF(NP.EQ.0)    GO TO 99
      IF(NP.LT.0)    GO TO 75
      C=X(NP)
      D=Y(NP)
      IF(JT.NE.0)GO TO 1
      JT=1
      K=NR
      NAN=J
      A1=C-A
      A2=D-B
      ANG=ATAN2(A1,A2)*(-1.)+PIB2
      IF(ANG.LT.0) ANG=PI2+ANG
      IF(MUL.LT.0)GO TO 3
      DANG=(TANG-ANG)
      IF(DANG.LT.0) DANG=DANG+PI2
      GO TO 2
3     DANG=ANG-TANG
      IF(DANG.LT.0) DANG=DANG+PI2
      GO TO 2
1     A1=C-A
      A2=D-B
      ANG=ATAN2(A1,A2)*(-1.)+PIB2
      IF(ANG.LT.0) ANG=PI2+ANG
      IF(MUL.LT.0)GO TO 4
      DDANG=TANG-ANG
      IF(DDANG.LT.0)DDANG=DDANG+PI2
      IF(DDANG.GE.DANG)GO TO 2
      DANG=DDANG
      K=NR
      NAN=J
      GO TO 2
4     DDANG=ANG-TANG
      IF(DDANG.LT.0)DDANG=DDANG+PI2
      IF(DDANG.GE.DANG)GO TO 2
      DANG=DDANG
      K=NR
      NAN=J
2     J=J+MUL
      IF(J.GT.7)GO TO 99
      GO TO 5
75    IF(L.NE.NR)GO TO 99
      IF(.NOT.LLAB)GO TO 751
      LLAB=.FALSE.
      GO TO 752
751   LLAB=.TRUE.
752   LAB=NR
      TEST=.TRUE.
      NR=IABS(NP)
      J=1
      GO TO 5
99    IF(NAN.EQ.0)WRITE(6,100)
100   FORMAT(' ','ANGFND DID NOT WORK')
      NR=K
      IF(NR.NE.L) RETURN
      LAB=LANG
      LLAB=ANGLL
      TEST=ANGTE
      RETURN
      END
```

Figure I.15 ANGFND

```
C***** MARKIT
C
C       THE PURPOSE OF THIS SUBROUTINE IS TO INSERT POINTS AT A GIVEN
C       DISTANCE FROM THE POINTS ALONG A CONTOUR LINE SO THAT
C       THE PROGRAM SMTHLN WILL NOT CAUSE CONTOURS TO CROSS,
C       FOR A MORE COMPLETE DESCRIPTION OF THIS SEE THE PROGRAM
C       WRITE-UP. THE SECONDARY PURPOSE OF THIS SUBROUTINE IS
C       TO CALCULATE THE POSITIONS OF SYMBOLS TO BE PLACED ALONG
C       THE CONTOURS.
C
C       PARAMETERS:
C                  X      THE X COORDINATES OF THE CONTOUR LINE
C                  Y      THE Y COORDINATES OF THE CONTOUR LINE
C                  M      THE NUMBER OF POINTS ON THE CONTOUR LINE
C                         (I.E. THE NUMBER OF X,Y COORDINATES)
C                  LSIG   A DUMMY PARAMETER TO ALLOW FOR LINE DOTTING
C                  XFACT  THE FACTOR BY WHICH ALL PLOTTED VALUES SHALL
C                         BE MULTIPLIED
C                  CONT   THE PARTICULAR CONTOUR BEING PLOTTED
C                  LET    A SIGNAL TO TELL THE SUBROUTINE IF THE USER
C                         DESIRES THE CONTOUR VALUES PLACED ON THE OUTPU
C                         PLOT.
C
C
      SUBROUTINE MARKIT (X,Y,M,LSIG,XFACT,CONT,LET)
      DIMENSION X(M),Y(M),XLIN(200),YLIN(200)
      COMMON/SYM/ CONTC,XH,THETA,CNT
      IF(M.LT.2) RETURN
      CONTC=CONT
      LSIG=0
      SUM=0.
      CELL=.01/XFACT
      CINT=(CELL*200.)*XFACT
      XH=.070/XFACT
      CL=ALOG10(CONT)
      IL=CL+1
      CL=IL
      CNT=CL*XH
      EF=.50/XFACT
      J=1
      N1=0
      DO 5 I=1,M
      I1=I+1
      IF(I1.GT.M) GO TO 1
      DIST=SQRT((X(I1)-X(I))**2+(Y(I1)-Y(I))**2)
      DISTF=XFACT*DIST
      SUM=SUM+DISTF
      IF (LET.EQ.0) SUM=0.
      IF(SUM.LT.CINT) GO TO 1
      IF(DISTF.LT.1.0) GO TO 1
      DX=X(I1)-X(I)
      DY=Y(I1)-Y(I)
      THETA=ATAN2(DY,DX)
      DN=DISTF/2.-(CNT/2.)*XFACT
      XN=DN*COS(THETA)
      YN=DN*SIN(THETA)
      XLIN(J)=X(I)+XN
      YLIN(J)=Y(I)+YN
      XNEW=(CNT*COS(THETA))+XLIN(J)
      YNEW=(CNT*SIN(THETA))+YLIN(J)
      LSIG=1
      CALL SMTHLN(XLIN,YLIN,J,CELL,LSIG)
      LSIG=0
      JJ=1
      XLIN(JJ)=XNEW
      YLIN(JJ)=YNEW
4     J=2
      SUM=0.
      GO TO 5
1     XLIN(J)=X(I)
      YLIN(J)=Y(I)
      IF(I1.GT.M) GO TO 5
      IF (DISTF.LT.0.5) GO TO 7
      CF=.25/XFACT
      J=J+1
      DX=X(I1)-X(I)
      DY=Y(I1)-Y(I)
      THETA=ATAN2(DY,DX)
      XN=CF*COS(THETA)
      YN=CF*SIN(THETA)
      XLIN(J)=X(I)+XN
      YLIN(J)=Y(I)+YN
      J=J+1
      XLIN(J)=X(I1)-XN
      YLIN(J)=Y(I1)-YN
7     J=J+1
      XLIN(J)=X(I1)
      YLIN(J)=Y(I1)
      IF (J.LT.196) GO TO 5
      N1=N1+1
      CALL SMTHLN(XLIN,YLIN,J,CELL,LSIG)
      J=1
5     CONTINUE
      IF(N1.LT.1) CALL SMTHLN(XLIN,YLIN,J,CELL,LSIG)
      RETURN
      END
```

Figure I.16 MARKIT

```
C***** ISON
C
C       THIS SUBROUTINE IS WRITTEN FOR THE PURPOSE OF SETTING UP AND
C       CHECKING A BIT PLANE. THE IDEA BEING THAT WE HAVE A LARGE MATRIX
C       WITH EVERY VALUE IN IT SET TO ZERO ( THE MATRIX IS N BY N ) TO
C       START WITH, IF A PARTICULAR OCCURANCE TAKES PLACE AT THE NI BY
C       NJ TH   POSITION (I.E. A CONTOUR GOES THROUGH THIS CELL) THEN
C       THE VALUE AT THAT POSITION IN THE MATRIX IS CHANGED FROM ZERO
C       TO ONE. WE CAN THEREFORE TELL IF THIS CELL HAS BEEN PROCESSED
C       PREVIOUSLY. THE REASON A BIT PLANE IS USED IS TO CONSERVE
C       STORAGE SPACE SINCE EACH MEMBER OF THE ARRAY IS SIMPLY
C       THE BIT  '0'  OR  '1'.
C
C       IF IND= -1 THEN CLEAR THE WHOLE ARRAY
C                0 THEN CHECK THE FLAG AND RETURN EITHER A ZERO OR A ONE
C                  FOR ISON
C               +1 THEN CHECK THE FLAG AND SET IT TO ONE
C
C                     - C A U T I O N -
C             THIS SUBROUTINE HAS THE TWO MACHINE DEPENDANT
C          FUNCTION SUBPROGRAMS  'AND' AND  'OR'.   IT MAY NOT
C          RUN PROPERLY ON MACHINES OTHER THAN THE IBM 370-155.
C
      FUNCTION ISON( NI , NJ , IND )
      DIMENSION IARRAY( 1000 )
      DIMENSION IMASK( 32 )
      DATA IMASK /Z1, Z2, Z4, Z8, Z10, Z20, Z40, Z80,
     1 Z100, Z200, Z400, Z800, Z1000, Z2000, Z4000,
     2 Z8000, Z10000, Z20000, Z40000, Z80000,
     3 Z100000, Z200000, Z400000, Z800000,
     4 Z1000000, Z2000000, Z4000000, Z8000000,
     5 Z10000000, Z20000000, Z40000000, Z80000000 /

      IF( IND .NE.(-1)) GO TO 1
      DO 100 I = 1 , 1000
100   IARRAY( I ) = 0
      NN = NI
      ISON=0
      RETURN
1     CONTINUE
      IPOS = ( NJ - 1 ) * NN + NI
      IWORD = IPOS / 32
      IBIT = IPOS - IWORD * 32  + 1
      IWORD = IWORD + 1
      ITEMP = LAND( IARRAY( IWORD ) , IMASK( IBIT )  )
      ISON = 1
      IF( IND .EQ. 0 ) GO TO 200
       IF( ITEMP .NE. 0 ) RETURN
      ISON = 0
      IARRAY( IWORD ) = LOR( IARRAY( IWORD ) , IMASK( IBIT )  )
      RETURN
200   CONTINUE
       IF( ITEMP .EQ. 0 ) ISON= 0
      RETURN
      END
```

```
LOR     CSECT
        USING LOR,15
        SAVE  (14,10),T,*
        L     2,0(0,1)      LOAD ADDRESS OF ARGUMENT LIST IN R2
        L     0,0(2)
        L     2,4(0,1)      LOAD ADDRESS OF SECOND ARGUMENT INTO R2
        O     0,0(0,2)      LOGICAL OR
        LM    1,10,24(13)
        RETURN (14,15),T,RC=0
```

```
LAND    CSECT
        USING LAND,15
        SAVE  (14,10),T,*
        L     2,0(0,1)
        L     0,0(0,2)
        L     2,4(0,1)
        N     0,0(0,2)      LOGICAL AND
        LM    1,10,24(13)
        RETURN (14,15),T,RC=0
        END
```

Figure I.17 ISON, LOR and LAND

```
C***** WORK1 (FOR EYEBALL)
C
C       THE PURPOSE OF THIS SUBROUTINE IS TO PASS
C       INFORMATION CONCERNING THE INTERSECTION POINTS
C       AS FOUND BY SCHNET BACK TO CONVER. THIS IS
C       ACCOMPLISHED THROUGH THE COMMON BLOCK.
C
C
        SUBROUTINE WORK1(XLIN,YLIN,J,LSIG)
        DIMENSION XLIN(200),YLIN(200)
        COMMON/PLT/ZN(200),YN(200),NNXN,XN(200)
        DO 1 I=1,J
        XN(I)=XLIN(I)
1       YN(I)=YLIN(I)
        NNXN=J
        RETURN
        END
```

```
C***** WORK2 (FOR EYEBALL)
C
C       THE PURPOSE OF THIS SUBROUTINE IS TO PLOT THE
C       HORIZONS OF EYEBALL AS THEY ARE PASSED FROM
C       CONVER.
C
C
        SUBROUTINE WORK2(XP,YP,N,LSIG)
        DIMENSION XP(N),YP(N)
        IPEN=3
        CALL PLOT(XP(1),YP(1),IPEN)
        IPEN=2
        DO 1 I=2,N
1       CALL PLOT(XP(I),YP(I),IPEN)
        RETURN
        END
```

Figure I.18 WORK1, WORK2 and FINDIT

```
C***** FINDIT
C
C       THIS SUBROUTINE FINDS A PARTICULAR POINTER IN A LIST
C       OF POINTERS CALLED NPT
C
C   PARAMETERS:
C       J- IS THE POSITION OF THE PARTICULAR POINTER IN THE ARRAY NPT
C          AND J IS RETURNED BY FINDIT.
C          IF THE POINTER CAN'T BE FOUND J=-1 IS RETURNED
C       N- IS THE VARIABLE DIMENSION OF NPT(20,N)
C       NR- IS THE MAJOR NODE NUMBER AT WHICH THE SCAN OR SEARCH IS
C           ROTATING.
C       NQ- IS THE POINTER WHICH IS BEING SOUGHT IN THE VECTOR NPT.
C
C
        SUBROUTINE FINDIT(J,NR,NQ,N,NPT)
        COMMON/ANGLE/ LLAB,LAB,TEST,MUL
        LOGICAL LLAB
        DIMENSION NPT(07,N)
        K=0
        J=1
        IF(NR.LT.0) NR=IABS(NR)
        NR1=NR
1       IF(NPT(J,NR).NE.NQ) GO TO 2
        IF(NQ.NE.0) GO TO 3
        J=J-1
        IF(K.GT.0)J=J-1
4       GO TO 3
2       J=J+1
        IF(J.GT.7)GO TO 6
        NPJ=NPT(J,NR)
        IF(NPJ.GT.0)GO TO 1
        IF(NPJ.LT.0)GO TO 5
        IF(NQ.EQ.0) GO TO 1
        GO TO 6
5       NPJ=IABS(NPJ)
        IF(NQ.EQ.0.AND.K.GT.0) GO TO 1
        IF(NPJ.EQ.NR2)GO TO 7
        NR=NPJ
        LLAB=.TRUE.
        K=1
        J=1
        GO TO 1
7       NR=NR2
        J=6
        GO TO 3
6       IF(NQ.NE.0)GO TO 8
        J=7
        GO TO 3
8       IF(NQ.NE.NR3) GO TO 9
        NQ=NR2
        J=1
        K=0
        GO TO 1
9       J=-1
3       NR2=NR1
        NR3=NR
        RETURN
        END
```

```
C***** SORTER
C
C       THIS SUBROUTINE IS USED TO SORT VALUES OF A ONE
C       DIMENSIONAL ARRAY WITHOUT MOVING THE MEMBERS BUT ONLY
C       RECORDING THEIR SORTED POSITIONS INTO A ONE DIMENSIONAL
C       INTEGER ARRAY.
C
C       A IS THE ARRAY TO BE SORTED
C       N IS THE NUMBER OF ELEMENTS IN A
C       IJ IS THE INTEGER ARRAY IN WHICH THE SORTED POSITIONAL VALUES
C          ARE RETURNED.
C
C     REFERENCE:
C               SHELLSORT  BY  J. BOOTHROYD
C               ALGORITHM  #201
C               COLLECTED ALGORITHMS FROM THE CACM
C
C
      SUBROUTINE SORTER(A,IJ,N)
      DIMENSION A(1),IJ(1)
      DO 1 I=1,N
1     IJ(I)=I
      M=N
10    M=M/2
      IF(M.EQ.0) RETURN
      K=N-M
      DO 30 J=1,K
      I=J
20    IM=I+M
      IF(A(IJ(IM)).GT.A(IJ(I)))GO TO 30
      KK =IJ(IM)
      IJ(IM)=IJ(I)
      IJ(I)=KK
      I=I-M
      IF(I.GT.0) GO TO 20
30    CONTINUE
      GO TO 10
      END
```

```
C***** STRIP
C
C       THE PURPOSE OF THIS SUBROUTINE IS TO FIND AND REMOVE
C       THE REDUNDANT HALVES OF CONTIGUOUS EQUAL PAIRS OF NUMBERS
C       IN THE VECTORS XN AND YN.
C       XN(I-1) MUST EQUAL XN(I)
C                 AND
C       YN(I-1) MUST EQUAL YN(I)
C       BEFORE XN(I) AND YN(I) WILL BE REMOVED
C       THE OLD VECTORS HAVE LENGTHS OF NNXN WHICH IS CHANGED
C       AND BECOMES THE LENGTH OF THE NEW VECTORS
C
C
      SUBROUTINE STRIP(XN,YN,NNXN)
      DIMENSION NN(200),XNN(200),YNN(200),XN(200),YN(200)
      CALL SORTER(XN,NN,NNXN)
      DO 4 I=1,NNXN
      XNN(I)=XN(NN(I))
4     YNN(I)=YN(NN(I))
      DO 5 I=1,NNXN
      XN(I)=XNN(I)
5     YN(I)=YNN(I)
      EPS=.001
      J=1
      NN(1)=0
      DO 1 I=2,NNXN
      IF(ABS(XN(I)-XN(I-1)).GT.EPS) GO TO 2
      IF(ABS(YN(I)-YN(I-1)).GT.EPS) GO TO 2
C
C   ADJACENT POINTS ARE EQUAL
C
      NN(I)=1
      GO TO 1
2     NN(I)=0
1     CONTINUE
      DO 3 I=1,NNXN
      IF(NN(I).NE.0) GO TO 3
      XN(J)=XN(I)
      YN(J)=YN(I)
      J=J+1
3     CONTINUE
      NNXN=J-1
      RETURN
      END
```

Figure I.19 SORTER and STRIP

```
C***** ROTATE
C
C     THIS SUBROUTINE ACCEPTS TWO ONE DIMENSIONAL VECTORS X AND Y
C     AND A VIEWPOINT AND CENTRAL POINT. IT THEN MAKES THE VIEWPOINT
C     INTO THE ORIGIN AND MAKES THE CENTRAL POINT LIE ALONG THE
C     X AXIS.
C
C
      SUBROUTINE ROTATE(X,Y,N,VPT,CPT,IDUM)
      DIMENSION X(N),Y(N),VPT(3),CPT(3),CPN(3)
      XMR=CPT(1)-VPT(1)
      YMR=CPT(2)-VPT(2)
C
C  TRANSFORM AND ROTATE , WHERE R IS THE RADIUS OF ROTATION
C
      DO 11 I=1,N
      X(I)=X(I)-VPT(1)
11    Y(I)=Y(I)-VPT(2)
      R=SQRT(XMR*XMR+YMR*YMR)
      COSAN=XMR/R
      SINEA=(-1)*YMR/R
      DO 1 I=1,N
      XI=X(I)
      YI=Y(I)
      X(I)=XI*COSAN-YI*SINEA
1     Y(I)=XI*SINEA+YI*COSAN
      IF(IDUM.EQ.0) RETURN
      DO 2 I=1,2
      CPN(I)=CPT(I)-VPT(I)
2     VPT(I)=0.0
      CPT(1)=CPN(1)*COSAN-CPN(2)*SINEA
      CPT(2)=CPN(1)*SINEA+CPN(2)*COSAN
      RETURN
      END
```

```
C***** CENTRE
C
C       THE PURPOSE OF THIS SUBROUTINE IS TO FIND AND RETURN IN
C       NCEN THE LABEL OF THE DATA POINT WHICH IS CLOSEST TO THE
C       ARITHMETIC MEAN OF BOTH X AND Y. OR IN OTHER WORDS TO
C       RETURN THE LABEL OF THE POINT CLOSEST TO THE CENTER OF THE
C       X,Y DATA SET. X AND Y ARE VECTORS OF LENGTH N.
C
C
      SUBROUTINE CENTRE(X,Y,NCEN,N)
      DIMENSION X(N),Y(N)
      L=1
      CALL MAXMIN(X,XMAX,N,J,L)
      L=0
      CALL MAXMIN(X,XMIN,N,J,L)
      L=1
      CALL MAXMIN(Y,YMAX,N,J,L)
      L=0
      CALL MAXMIN(Y,YMIN,N,J,L)
      XCEN=(XMAX-XMIN)/2.
      YCEN=(YMAX-YMIN)/2.
      XD=ABS(X(1)-XCEN)
      YD=ABS(Y(1)-YCEN)
      DO 1 I=2,N
      IF(ABS(X(I)-XCEN).GE.XD)GO TO 1
      IF(ABS(Y(I)-YCEN).GE.YD)GO TO 1
      XD=ABS(X(I)-XCEN)
      YD=ABS(Y(I)-YCEN)
      NCEN=I
1     CONTINUE
      RETURN
      END
```

Figure I.20 ROTATE and CENTRE

```
C***** MAXMIN
C
C       THIS SUBROUTINE IS USED TO FIND EITHER THE MAXIMUM OR THE
C       MINIMUM OF THE ARRAY  X  , WHERE N IS THE LENGTH OF  X .
C       IF NWHAT.EQ.1 THEN THE VALUE OF THE MAXIMUM MEMBER OF  X
C       IS RETURNED IN  A  AND THE POSITION OF  A  IN THE ARRAY  X
C       IS RETURNED IN  J.
C       IF NWHAT.NE.1 THEN THE VALUE OF THE MINIMUM MEMBER OF  X  IS
C       RETURNED IN  A  AND THE POSITION OF  A  IN THE ARRAY  X  IS
C       RETURNED IN  J.
C
C
      SUBROUTINE MAXMIN(X,A,N,J,NWHAT)
      DIMENSION X(N)
      IF(NWHAT.EQ.1) GO TO 2
      A=X(1)
      J=1
      DO 1 I=2,N
      IF(X(I).GE.A) GO TO 1
      A=X(I)
      J=I
1     CONTINUE
      GO TO 999
2     A=X(j)
      J=1
      DO 3 I=2,N
      IF(X(I).LE.A) GO TO 3
      A=X(I)
      J=I
3     CONTINUE
999   RETURN
      END
```

```
C***** POINTS
C
C       THE PURPOSE OF THIS SUBROUTINE IS TO FIND AND RETURN, IN
C       THE ARRAYS XP,YP, THE K INTERSECTION POINTS OF THE LINE
C       FROM (XMIN,WRKLIN) TO (XMAX,WRKLIN) WITH THE IRREGULAR
C       DATA SET.
C
C       PARAMETERS:
C
C       X       AN N LONG VECTOR OF X COORDINATES OF THE DATA SET
C       Y       AN N LONG VECTOR OF Y COORDINATES OF THE DATA SET
C       Z       AN N LONG VECTOR OF Z COORDINATES OF THE DATA SET
C       NPT     A (7,N) ARRAY OF POINTERS OF THE DATA SET
C       XMIN    THE MINIMUM X VALUE IN THE DATA SET
C       XMAX    THE MAXIMUM X VALUE IN THE DATA SET
C       WRKLIN THE Y COORDINATE OF THE PARTICULAR LINE THROUGH
C               THE DATA SET.
C       N       THE NUMBER OF REAL AND DUMMY POINTERS IN THE DATA
C               SET
C       XP      THE X COORDINATES OF THE INTERSECTION POINTS
C       YP      THE Y COORDINATES OF THE INTERSECTION POINTS
C       NC      THE LABEL OF THE CENTRAL POINT IN THE DATA SET
C       K       THE NUMBER OF INTERSECTION POINTS RETURNED
C
C
      SUBROUTINE POINTS(X,Y,Z,NPT,XMIN,XMAX,WRKLIN,N,XP,YP,NC,K)
      COMMON/PLT/ ZN(200),YN(200),NNXN,XN(200)
      DIMENSION X(N),Y(N),Z(N),NPT(07,N),XP(N),YP(N),DX(200),DY(200)
      NRIT=2
      NCALL=0
      ND=NC
      CALL SCHNET(X,Y,Z,NPT,RP,CO,ND,N,X(ND),XMIN,Y(ND),WRKLIN,NRIT,
     1NCALL,L1,L2,XFO,LET)
      NCALL=0
      NRIT=1
      CALL SCHNET(X,Y,Z,NPT,RP,CO,L2,N,XMIN,XMAX,WRKLIN,WRKLIN,NRIT,
     1NCALL,L1,L2,XFO,LET)
      DO 2 I=1,NNXN
      DX(I)=XN(I)
      DY(I)=YN(I)
2     CONTINUE
      NNNN=NNXN
      CALL STRIP(DX,DY,NNNN)
      DO 1 I=1,NNNN
      XP(I)=DX(I)
1     YP(I)=DY(I)
      K=NNNN
      RETURN
      END
```

Figure I.21 MAXMIN and POINTS

```
C***** OUTSID
C
C       THE PURPOSE OF THIS SUBROUTINE IS TO FIND IN WHICH
C       TRIANGLE THE MIDPOINT OF TWO INTERSECTION POINTS IS
C       LOCATED. IN OTHER WORDS WHICH TRIANGLE LIES BETWEEN
C       A PAIR OF INTERSECTION POINTS.
C
C       PARAMETERS:
C
C       X       X COORDINATES OF THE DATA SET
C       Y       Y COORDINATES OF THE DATA SET
C       XP      X COORDINATES OF THE INTERSECTION POINTS
C       YP      Y COORDINATES OF THE INTERSECTION POINTS
C       NGRLIN THE NUMBER OF THE TRIANGLE BETWEEN THE INTER-
C              SECTION POINTS AN NTRI LONG VECTOR
C       K       THE NUMBER OF INTERSECTION POINTS
C       N       THE NUMBER OF REAL AND DUMMY POINTS IN THE DATA SET
C       NV1
C       NV2     INTEGER ARRAYS CONTAINING THE VERTEX NUMBERS OF
C       NV3     EVERY TRIANGLE IN THE DATA SET
C       NTRI    THE NUMBER OF TRIANGLES IN THE DATA SET
C
C
      SUBROUTINE OUTSID (X,Y,XP,YP,NGRLIN,K,N,NV1,NV2,NV3,NTRI)
      DIMENSION X(N),Y(N),XP(N),YP(N),VX(3),VY(3),NGRLIN(NTRI),
     1NV1(NTRI),NV2(NTRI),NV3(NTRI)
      NN=1
      KM1=K-1
      DO 2 J=1,KM1
      AX=XP(J)+(XP(J+1)-XP(J))/2.
      AY=YP(J)
      DO 1 I=1,NTRI
      KL=0
      ITEMP=ISON(I,NN,KL)
      IF(ITEMP.EQ.1) GO TO 1
      VX(1)=X(NV1(I))
      VX(2)=X(NV2(I))
      VX(3)=X(NV3(I))
      VY(1)=Y(NV1(I))
      VY(2)=Y(NV2(I))
      VY(3)=Y(NV3(I))
      NP=3
      CALL PNPOLY(AX,AY,VX,EY,NP,INOUT)
      IF(INOUT.EQ.-1) GO TO 1
      NGRLIN(J)=I
      GO TO 2
1     CONTINUE
2     CONTINUE
      RETURN
      ENTRY SETIS(NTRI)
      KL=-1
      NN=1
      ITEMP=ISON(NTRI,NN,KL)
      RETURN
      ENTRY RESET(JJ)
      NN=1
      KL=1
      ITEMP=ISON(JJ,NN,KL)
      RETURN
      END
```

Figure I.22 OUTSID

```
C***** WORK1
C
C     THE PURPOSE OF THIS SUBROUTINE IS TO PLOT CONTOUR
C     VALUES GENERATED BY THE PROGRAMS KONTUR OR ISOKONT.
C
C    PARAMETERS:
C               N     IS THE NUMBER OF X,Y POINTS ALONG THE CONTOUR
C               LSIG  IS A DUMMY SIGNAL WHICH CAN CARRY A VALUE TO
C                     ALLOW FOR DOTTED LINES IF SO DESIRED
C               CONT  IS THE VALUE OF THE CONTOUR BEING PLOTTED
C               INT   IS ANOTHER DUMMY WHICH CAN BE USED FOR ANY
C                     PURPOSE THE USER DESIRES
C               XH    IS THE HEIGHT OF THE SYMBOLS TO BE PLOTTED
C               THETA IS THE ANGLE AT WHICH THE SYMBOLS SHALL PLOTTED
C               CNT   IS THE STARTING POSITION OF THE SYMBOL TO BE
C                     PLOTTED IN RELATION TO ITS X,Y POSITION
C               X     THE X COORDINATES OF THE POINTS TO BE PLOTTED
C               Y     THE Y COORDINATES OF THE POINTS TO BE PLOTTED
C
C
      SUBROUTINE WORK1(X,Y,N,LSIG)
      COMMON/SYM/CONT,XH,THETA,CNT
      DIMENSION X(N),Y(N)
      IPEN=3
      CALL PLOT(X(j),Y(j),IPEN)
      IPEN=2
      DO 7 I=1,N
      CALL PLOT(X(I),Y(I),IPEN)
7     CONTINUE
      IF(LSIG.EQ.0) RETURN
      DTH=THETA*57.308
      IF(DTH.LT.0.0) DTH=DTH+360.
      PH=XH/2.
      IF(DTH.GT.90..AND.DTH.LT.270.) GO TO 8
      X(N)=X(N)+PH*SIN(THETA)
      Y(N)=Y(N)-PH*COS(THETA)
      IP=-1
      CALL NUMBER(X(N),Y(N),XH,CONT,DTH,IP)
      RETURN
8     IF(DTH.GT.180.) GO TO 9
      DTH=DTH+180.
      XN=X(N)+CNT*COS(THETA)-PH*SIN(THETA)
      YN=Y(N)+CNT*SIN(THETA)+PH*COS(THETA)
      IP=-1
      CALL NUMBER(XN,YN,XH,CONT,DTH,IP)
      RETURN
9     DTH=DTH-180.
      XN=X(N)+CNT*COS(THETA)-PH*SIN(THETA)
      YN=Y(N)+CNT*SIN(THETA)+PH*COS(THETA)
      IP=-1
      CALL NUMBER(XN,YN,XH,CONT,DTH,IP)
      RETURN
      END
```

```
C***** BORDER
C
C      THE PURPOSE OF THIS SUBROUTINE IS FOR PLOTTING BORDERS
C      AROUND CONTOUR MAPS PRODUCED BY KONTUR OR ISOKONT
C      THE PARAMETERS ARE SELF EXPLANATORY EXCEPT FOR XFACT WHICH IS
C      THE FACTOR BY WHICH ALL POINTS TO BE PLOTTED ARE MULTIPLIED.
C
C      THIS SUBROUTINE EXPECTS ONE DATA CARD ON UNIT 5
C      THIS DATA CARD SHALL CONTAIN THE TITLE THAT THE
C      USER DESIRES ON THE OUTPUT MAP, IT CAN OF COURSE BE
C      JUST A BLANK CARD, BUT THERE MUST BE A CARD.
C
C
      SUBROUTINE BORDER(XMAX,XMIN,YMAX,YMIN,XFACT)
      DIMENSION TITLE(20)
      CALL PLOT(XMAX,YMIN,3)
      CALL PLOT(XMAX,YMAX,2)
      CALL PLOT(XMIN,YMAX,2)
      CALL PLOT(XMIN,YMIN,2)
      CALL PLOT(XMAX,YMIN,2)
      READ(5,100) (TITLE(I),I=1,20)
100   FORMAT(20A4)
      X=XMAX+1./XFACT
      Y=YMIN+1./XFACT
      XH=.175/XFACT
      CALL SYMBOL(X,Y,XH,TITLE,90.0,80)
      XOUT=XMAX+5./XFACT
      CALL PLOT(XOUT,0.0,3)
      RETURN
      END
```

Figure I.23 WORK1 and BORDER

APPENDIX II: A SAMPLE DATA STRUCTURE

A sample data structure was prepared for the topographic surface of the Ptarmigan Creek area of Canoe River valley located in central British Columbia (Figure 4.1). This is a good surface for tests of this data structure since it consists of regular surface functions interspersed with irregular functions. The river bottom is a well defined gently sloping plane with the sides of the valley breaking sharply from the bottom and rising to irregular peaks. Surface specific points were chosen from the surface as points representative of changes in neighbourhoods. A triangulation of these points was carried out with the criteria for triangle formation being to minimize the distance between the triangular plane and the real surface (Figure 3.3). The vertices of the triangles were numbered from one through eighty one as shown in Figure II.1. Following the rules outlined in CHAPTER III the data were recorded in the format

(I 5 , 3 F (10.0), 7 I 5).

The finished data structure, ready for use with the display routines, is shown in Figure II.2. This structure consists of eighty one real data points and eight dummy data points (i.e. eight real points have more than seven pointers). This data structure was used to produce all the example graphics of the display system outlined in this thesis.

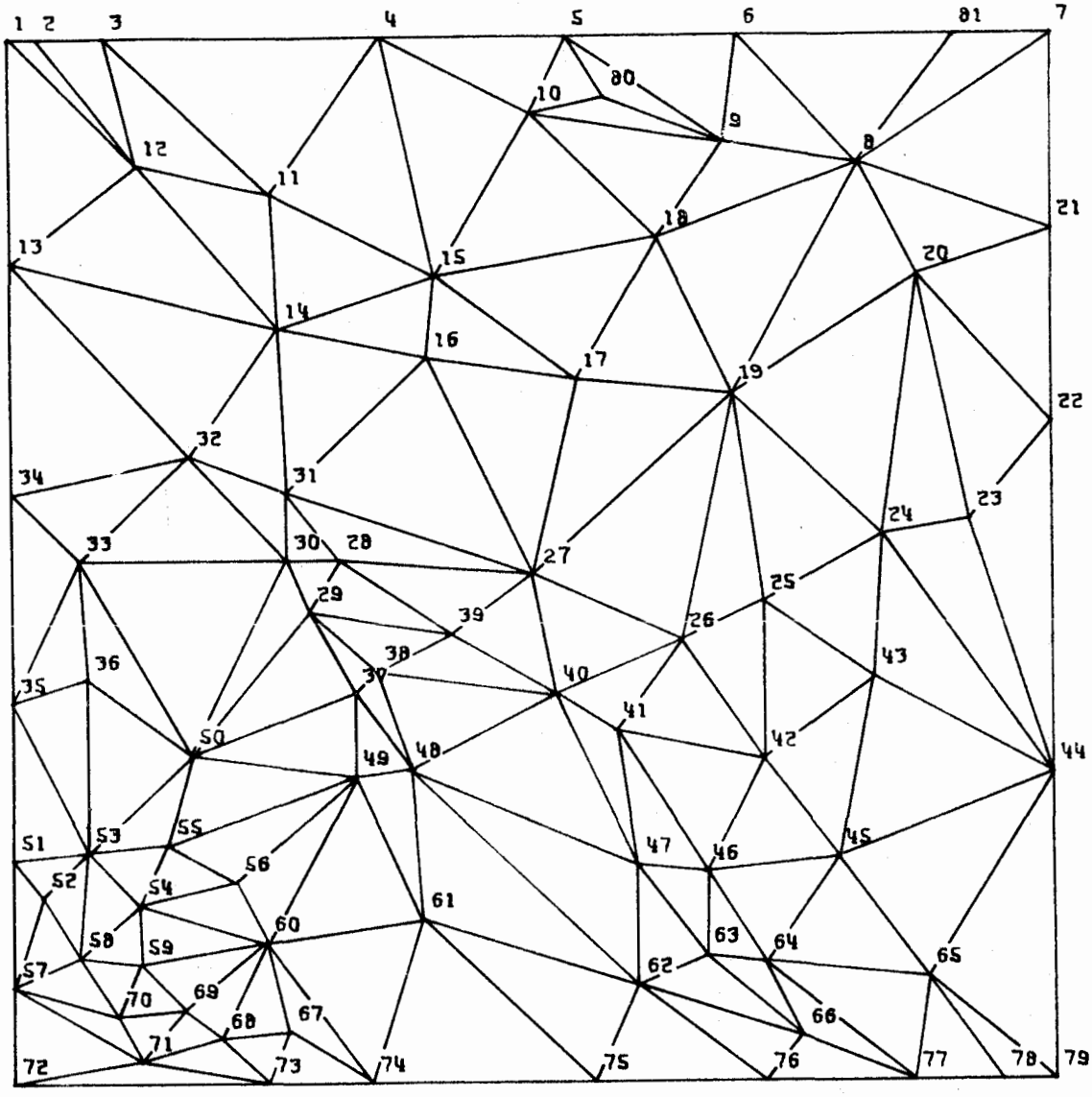


Figure II.1 Irregular Data Structure. The triangulated surface specific points ready for recording the data.

1	0.	7000.	732.	32000	2	12	1332000				
2	200.	7000.	762.	32000	3	12	132000				
3	650.	7000.	1006.	32000	4	11	12	232000			
4	2500.	7000.	1402.	32000	5	10	15	11	332000		
5	3750.	7000.	1600.	32000	6	9	80	10	432000		
6	4900.	7000.	1402.	32000	81	8	9	532000			
7	7000.	7000.	2149.	32000	21	8	8132000				
8	5700.	6150.	1692.	81	7	21	20	19	18	-82	
82	5700.	6150.	1692.	9	6	-8					
9	4800.	6300.	1570.	6	8	18	10	80	5		
10	3500.	6500.	1585.	5	80	9	18	15	4		
11	1750.	5950.	1067.	4	15	14	12	3			
12	850.	6150.	732.	11	14	13	1	2	3		
13	0.	5500.	732.	32000	1	12	14	32	3432000		
14	1800.	5050.	732.	15	16	31	32	13	12	11	
15	2850.	5400.	1097.	10	18	17	16	14	11	4	
16	2800.	4850.	762.	15	17	27	31	14			
17	3300.	4700.	756.	18	19	27	16	15			
18	4350.	5050.	1311.	9	8	19	17	15	10		
19	4850.	4600.	762.	8	20	24	25	26	27	-83	
83	4850.	4600.	762.	17	18	-19					
20	6100.	5400.	1341.	21	22	23	24	19	8		
21	7000.	5700.	1615.	732000		22	20	8			
22	7000.	4400.	1067.	2132000		44	23	20			
23	6450.	3750.	930.	22	44	24	20				
24	5850.	3650.	762.	20	23	44	43	25	19		
25	5050.	3200.	701.	24	43	42	26	19			
26	4500.	2950.	732.	19	25	42	41	40	27		
27	3500.	3400.	732.	17	19	26	40	39	28	-84	
84	3500.	3400.	732.	31	16	-27					
28	2200.	3500.	930.	27	39	29	30	31			
29	2000.	3150.	945.	28	39	38	37	50	30		
30	1850.	3500.	853.	28	29	50	33	32	31		
31	1850.	3950.	749.	16	27	28	30	32	14		
32	1200.	4200.	732.	14	31	30	33	34	13		
33	450.	3500.	1036.	32	30	50	36	35	34		
34	0.	3950.	1219.	32000	13	32	33	3532000			
35	0.	2550.	1478.	32000	34	33	36	53	5132000		
36	500.	2700.	1524.	50	53	35	33				
37	2300.	2600.	1027.	38	48	49	50	29			
38	2450.	2750.	1103.	39	40	48	37	29			
39	2950.	3000.	1158.	27	40	38	29	28			
40	3650.	2600.	1280.	26	41	47	48	38	39	27	
41	4050.	2350.	1250.	26	42	46	47	40			
42	5050.	2150.	701.	43	45	46	41	26	25		
43	5800.	2700.	701.	24	44	45	42	25			
44	7000.	2050.	762.	2232000		79	65	45	43	-85	
85	7000.	2050.	762.	24	23	-44					
45	5550.	1500.	701.	43	44	65	64	46	42		
46	4650.	1400.	1189.	42	45	64	63	47	41		
47	4200.	1450.	1259.	46	63	62	48	40	41		
48	2650.	2100.	1091.	40	47	62	61	49	37	38	
49	2300.	2050.	1143.	37	48	61	60	56	55	50	
50	1200.	2200.	1436.	30	29	37	49	55	53	-86	
86	1200.	2200.	1436.	36	33	-50					
51	0.	1500.	2042.	32000	35	53	52	5732000			
52	200.	1250.	2240.	53	58	57	51				
53	500.	1550.	2164.	36	50	55	54	58	52	-87	
87	500.	1550.	2164.	51	35	-53					
54	850.	1200.	1859.	55	56	60	59	58	53		
55	1050.	1600.	1829.	50	49	56	54	53			
56	1500.	1350.	1524.	49	60	54	55				
57	0.	650.	2164.	32000	51	52	58	70	71	-89	
89	0.	650.	2164.	7232000		-57					
58	450.	850.	1951.	53	54	59	70	57	52		
59	850.	800.	1951.	54	60	69	70	58			
60	1700.	950.	1859.	49	61	74	67	68	69	-88	
88	1700.	950.	1859.	59	54	56	-50				
61	2750.	1100.	1311.	62	75	74	60	49	48		
62	4200.	650.	1167.	63	66	76	75	61	48	47	
63	4650.	850.	1256.	64	66	62	47	46			
64	5050.	800.	1219.	45	65	77	66	63	46		
65	6150.	700.	732.	44	79	78	77	64	45		
66	5300.	300.	1213.	77	76	62	63	64			
67	1850.	350.	2103.	74	73	68	60				
68	1400.	300.	2404.	60	67	73	71	69			
69	1150.	500.	2134.	60	68	71	70	59			
70	700.	450.	1951.	59	69	71	57	58			
71	850.	150.	2134.	69	68	73	72	57	70		
72	0.	0.	1783.	32000	57	71	7332000				
73	1700.	0.	2134.	67	7432000		72	71	68		
74	2400.	0.	2103.	61	7532000		73	67	60		
75	3900.	0.	1372.	62	7632000		74	61			
76	5050.	0.	1192.	66	7732000		75	62			
77	6050.	0.	1225.	65	7832000		76	66	64		
78	6650.	0.	747.	7932000		77	65				
79	7000.	0.	695.	4432000		78	65				
80	4000.	6600.	1631.	9	10	5					
81	6350.	7000.	1859.	32000	7	8	632000				

Figure II.2 Records of the Data Structure

SELECTED BIBLIOGRAPHY

1. Abramson, N. Information Theory and Coding, New York: McGraw-Hill Book Co., Inc., 1963, pp. 1-39

2. Agterberg, F.P. "Computer Techniques in Geology," Earth Science Review, Vol.3, 1967, pp. 47-77

3. Armand, A.D. "The Application of Information Theory to Geographical Division into Areas," in Mathematical Methods and Modern Geography, U.S.S.R.: Kazan University, 1972 pp. 114-125

4. Berylant, A.M. "Maps of Interrelations and Their Application in Geographic Research," Soviet Geography, Review and Translation, Vol.13, 1972, pp. 472-481

5. Board, C. 1967, "Maps as Models," in Chorley, R.J., and P.Haggett, Models in Geography, London: Methuen & Co., Ltd., 1970, pp. 671-725

6. Board, C. "Cartographic Communication and Standardization," International Cartographic Association Proceedings,1972

7. Codd, E.F. "A Relational Model of Data for Large Shared Data Banks," Communications of the ACM, Vol.13, No.10, June 1970, pp. 377-387

8. Connelly, D.S. "The Coding and Storage of Terrain Height Data: An Introduction to Numerical Cartography," M.Sc.Thesis, Cornell University, September, 1968

9. Dayhoff, M.O. "A Contour-Map Program for X-Ray Crystallography," Communications of the ACM, Vol.6, No.10, October 1963, pp. 620-622

10. Dodd, George, C. "Elements of Data Management Systems," Computing Surveys, Vol.1, No.2, June 1969, pp. 117-133

11. Douglas, David H. and T.K.Peucker "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature," The Canadian Cartographer, Vol.10, No.2, December 1973, pp. 112-122

12. Gokhman, V.M. and M.M.Mekler "The Information Theory and Thematic Cartography," in Mathematical Methods and Modern Geography, U.S.S.R.: Kazan University, 1972, pp.99-113

13. Hamilton, J.A. A Survey of Data Structures for Interactive Graphics, Rand Corp. Memorandum, RM - 6145 - ARPA, April 1970

14. Johnson, R.E. and F.L.Kiokemeister Calculus with Analytic Geometry, 3rd edition. Boston: Allyn and Bacon, Inc., 1964

15. Kadar, I. and F.Karsay The More Profitable Storing and Transferring Possibilities of the Information Content of the Map, Budapest: Institute of Surveying, 1969

16. Knuth, Donald, E. The Art of Computer Programming, Volume 1/ Fundamental Algorithms. Reading, Massachusetts: Addison-Wesley Publishing Co., 1968

17. Kubert, B., J. Szabo, and S. Giulieri "The Perspective Representation of Functions of Two Variables," Journal of the Association for Computing Machinery, Vol.15, No.2, April 1968, pp. 193-204

18. Lawson, H.W. "PL/1 List Processing," Communications of the ACM, Vol.10, No.6, June 1967, pp. 358-367

19. Marchand, Bernard "Information Theory and Geography," Geographical Analysis, Vol.IV, No.3, July 1972, pp. 234-257

20. McIntyre, Donald, B., David D. Pollard, and Roger Smith Computer Programs for Automatic Contouring, Computer Contribution 23, Editor Daniel F. Merriam, State Geological Survey, The University of Kansas, 1968

21. Peucker, Thomas, K. Computer Cartography, Commission on College Geography Resource Paper No.17, Association of American Geographers, Washington, D.C.: 1972

22. Peucker, Thomas, K. Computer Cartography, A Working Bibliography, Discussion Paper No.12, Department of Geography, University of Toronto, August 1972

23. Prenter, P.M. Splines and Variational Methods, Manuscript, to be published, 1975

24. Ratajski, L. "The Research Structure of Theoretical Cartography," Paper, 6th Annual International Cartographic Association Conference, Montreal - Ottawa, 1972

25. Rayner, J.N. An Introduction to Spectral Analysis, No.2, Monographs in Spatial and Environmental Systems Analysis, London: Pion, 1971

26. Sukhov, V.I. "Application of Information Theory in Generalization of Map Contents," International Yearbook of Cartography, Vol.10, 1970, pp. 40-47

27. Thrower, Norman, J.W. Maps and Man, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1972

28. Tobler, Waldo, R. "Geographical Filters and their Inverses," Geographical Analysis, Vol.1, 1969, pp. 234-253

29. Warntz, W. "The Topology of a Socio-Economic Terrain and Spatial Flows," Papers, Regional Science Association, Vol.17, 1966, pp. 47-61

30. Williams, R. "A Survey of Data Structures for Computer Graphics Systems," Computing Surveys, Vol.3, No.1, March 1971, pp. 1-21

31. Yoeli, P. "Analytical Hill Shading," Surveying and Mapping, December 1965, pp. 573-579

32. Yoeli, P. "Analytical Hill Shading and Density," Surveying and Mapping, June 1966, 573-579

33. Yoeli, P. "The Mechanization of Analytical Hill Shading," The Cartographic Journal, Vol.4, 1967, pp. 82-88