DORSET INSTITUTE OF HIGHER EDUCATION

Automated Interpretation of Digital Images of Hydrographic Charts

By

Kelvin J. Goodson

Thesis submitted in candidature for
the degree of Doctor of Philosophy
at the Dorset Institute of
Higher Education
May 1987

In collaboration with Marinex Marine Holdings Ltd.
Creekmoor, Poole, Dorset

Automated Interpretation of Digital Images of Hydrographic Charts

Kelvin J. Goodson

ABSTRACT

Details of research into the automated generation of a digital database of hydrographic charts is presented. Low level processing of digital images of hydrographic charts provides image line feature segments which serve as input to a semi-automated feature extraction system, *(SAFE)*. This system is able to perform a great deal of the building of chart features from the image segments simply on the basis of proximity of the segments. The system solicits user interaction when ambiguities arise.

The creation of an intelligent knowledge based system *(IKBS)* implemented in the form of a backward chained production rule based system, which co-operates with the SAFE system, is described. The IKBS attempts to resolve ambiguities using domain knowledge coded in the form of production rules. The two systems communicate by the passing of goals from SAFE to the IKBS and the return of a certainty factor by the IKBS for each goal submitted. The SAFE system can make additional feature building decisions on the basis of collected sets of certainty factors, thus reducing the need for user interaction. This thesis establishes that the cooperating IKBS approach to image interpretation offers an effective route to automated image understanding.

Acknowledgements

My most sincere thanks go to Dr. P.H. Lewis, whose encouragement and help in all aspects of the research and production of this thesis have been unbounded. I am grateful to the DORSET INSTITUTE OF HIGHER EDUCATION and in particular to Mr. A. Potton for making this research project possible. I would like to thank Dr. A.A. Hashim and Dr. W. Randolph for their direction and many helpful suggestions. I wish to express my thanks to the Computer Science department of Southampton University for the use of their text processing facilities in the preparation of this document and in particular to Sebastian Rahtz for his help with the LaTeX text processing package. I am in great debt to Sally Talbot for her help and encouragement at the times when motivation was lacking. And lastly my greatest debt of thanks is, of course, to my parents.

Authors note: All the work in this thesis is the sole and original work of the author except where stated otherwise by acknowledgement or reference.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Aims of this Research

This research has been undertaken with the aim of developing a system for automated extraction of features from digital images of hydrographic charts. A motivation for pursuing this line of research is that a local company requires a digital database of hydrographic chart features for use in an automated marine navigation system for small boats which is being developed by the company. Figure 1.1 gives an outline of the format of the desired navigational system. It should take input data from a host of navigational aids such as satellite global positioning, radar, radio beacon and depth soundings. This data should be processed by a small boat behaviour model and correlated with stored hydrographic data. With a suitable man-machine interface, desired courses can be plotted and the boat's controls adjusted automatically in order to follow this course.

The aspect of the project with which this research is concerned is the *generation* of the hydrographic data. The utility of hydrographic information to

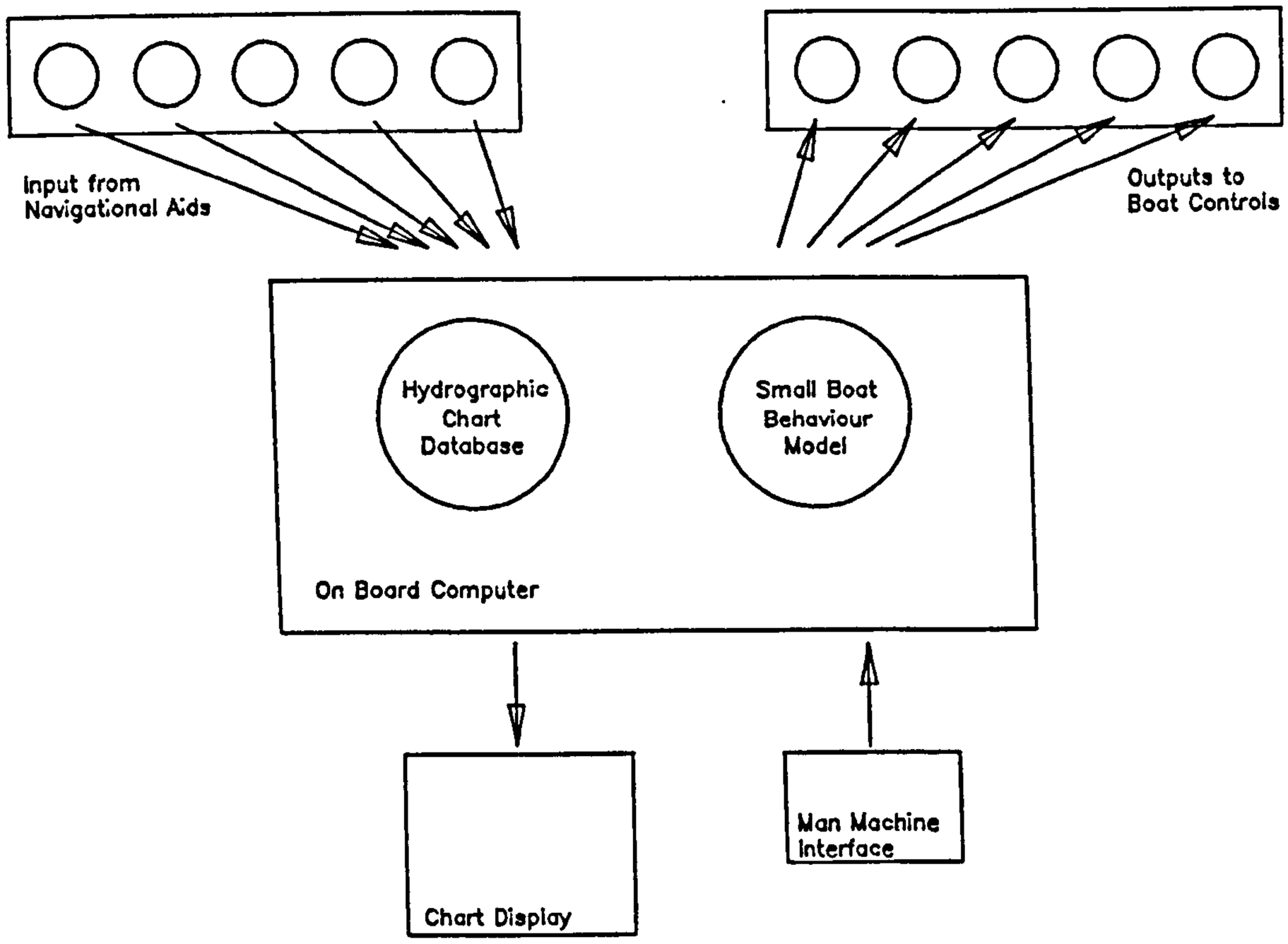


Figure 1.1: Small Boat Navigation System

the navigational system is twofold, firstly the data will be used as an information source for the navigational algorithms incorporated in the small boat behaviour model, and secondly, it will be used to provide an on board visual display to the mariner operating the system. Preliminary results of this research were presented in a paper at the *4th International Conference on Pattern Recognition* (Goodson and Lewis [1988]).

Attempts have been made in the past to automatically extract the information in complex line drawings such as terrain maps. Fully automatic systems have still not been realised and semi-automatic systems, based for example on the Laser Scan (Fulford [1981]) approach are not only highly expensive, but also require a substantial amount of user interaction. This is underlined by the fact that the Ordnance Survey has abandoned its approaches to automatic digitising of maps and has resorted to a totally manual system of digitisation.

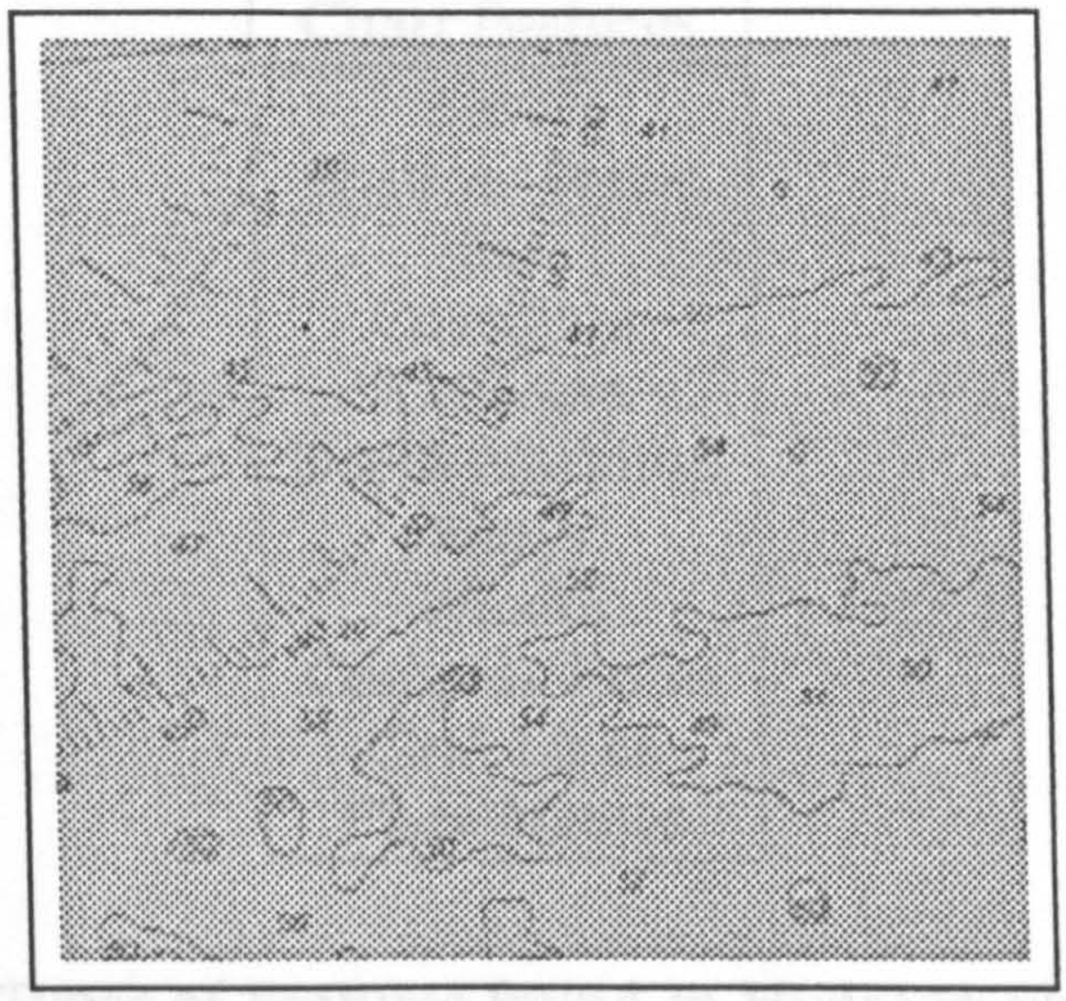


Figure 1.2: Image of a typical portion of a Hydrographic chart

The technique is, however, still undergoing continuous commercial development for increasing the level of automation of processing complex drawings and charts.

The information contained within hydrographic charts is much less varied or dense than in most terrain maps. Figure 1.2 shows a monochrome image of a typical portion of a hydrographic chart and table 1.1 lists the contents, in terms of the types of features found in charts.

It was decided to investigate the possibilities of extracting features from sea charts using image processing techniques. The objective of this research is therefore to develop a relatively cheap line drawing interpreter initially aimed at recognising features such as contours and coastlines in hydrographic charts. The system should create a digital database of features suitable for the dual purpose of rapid display and interrogation by navigational algorithms in an automated marine navigation system.

The aim of the initial stage of the research project was to develop suitable

Chart Features
Land
Shallow Water
Sea
Coastlines
Contours
Depth Markings
Buoys
Wrecks
Compass Roses
Grid Lines
Place Names

Table 1.1: Types of Features found in Hydrographic Charts

image processing algorithms to perform some form of primary feature extraction which might provide geometric descriptions of the *image* features of sufficient quality to build a database of the *chart* features. The quality of the representation of chart features was seen from the start as an important aspect of the project.

After some initial research it soon became clear that significant problems were likely to be encountered in obtaining coherent representations of chart features automatically. The types of problems to be expected were in developing suitably robust feature extraction algorithms which could reliably detect and produce good geometric representations of true image features whilst not producing spurious, fragmented or merged features.

A further aim in the later stages of the project would therefore be to investigate the possible routes for overcoming the deficiencies observed in the techniques developed. The possibilities for this extension to the processing at the time were seen to be :-

- Increasing the sophistication of a low level line tracker by incorporating

domain knowledge into the decision making processes of the algorithm

- Developing a procedural method of reorganising and modifying the feature data

- Developing some form of knowledge based processing approach to producing or modifying the feature data

1.2 Overview of this thesis

The structure of this thesis is as follows. The remainder of this introductory chapter (section 1.3) provides an overview of the theory directly related to the work in hand. The reader who is unfamiliar with the essentials of image processing is directed to appendix A and to the fundamental references cited in the first paragraph of section 1.3.

Chapter 2 presents a literature review of line tracking and related topics. A literature review of topics peripheral to the main research work is presented in appendix B. The relevant work carried out at the outset of the project is documented in chapter 3. It presents the research and development plan; a description of the facilities available to the project and the initial exploratory preprocessing of chart images. It then goes on to describe the implementation of a line tracker and the limitations of data driven line tracking. Chapter 4 describes how these limitations have, to a certain extent, been overcome by the development of a semi-automatic feature extraction system (S.A.F.E.) in which the information obtained from line tracking is assembled into coherent labelled features. The usage of this system is documented in the format of a sample terminal session. The extension of this system to include a cooperating knowledge based processing system to increase the level of automation is presented in chapter 5. Chapter 6 presents the conclusions which can be

drawn from this work and pointers to the possibilities for further research.

1.3 Related Theory

1.3.1 Digital Image Processing

Obtaining the Fundamentals

Many standard texts on image processing cover the processes of image formation, enhancement, segmentation, scene analysis etc.. The quantity of information necessary to describe these processes is too vast and will not be detailed in this thesis, although the basic principles of edge enhancement and detection are presented in appendix A. The reader who is unfamiliar with the background of image processing is directed to this appendix and the following texts. Gonzalez and Wintz [1977], Pratt [1982] and Castleman [1979] all give full, clear accounts of the fundamentals of the subject. The revised two volume text by Rosenfeld and Kak [1982] is an authoritative work containing details of many useful algorithms. Faugeras [1983] has edited a volume in which many well revered workers in the subject have presented chapters on selected topics which together form a useful insight into the fundamentals of computer vision. Ballard and Brown [1982] have developed a classic work on computer vision, concentrating on the higher level task of vision rather than what can be achieved using the various low level image processing operations. A useful route into the journal literature is the annual surveys produced by Rosenfeld, e.g. Rosenfeld [1987], which typically contain well over a thousand references relating to all aspects of image processing and computer vision.

A Brief Overview of a Typical Processing Sequence

Digital image processing is, when scaled against most other scientific disciplines, a young subject. The possibilities in terms of applications of image processing techniques have given researchers endless avenues to pursue over the past 30 years. Some have focussed upon long term goals, such as emulating the human eye / brain system, whilst others have adopted utilitarian approaches to provide functional image interpretation systems to work within constrained domains. This large research interest has resulted in many disparate approaches to similar tasks, each tailored to the meet the needs of the particular domain in question. Hence the nature of the science of image processing is very disjointed. There is no coherent binding theory that can be presented as one logical flow of information. A trend can however be observed in the overall approach to many image manipulation systems. The following list of process classifications describes a typical sequence of operations and a brief paragraph conveying the essence of each operation follows.

- Image acquisition
- Enhancement / noise reduction / restoration
- Feature enhancement
- Segmentation
- Object description
- Classification of scene elements
- Scene interpretation

Image Acquisition The most common method of obtaining a digital image is to sample the continuous image produced by some sensor such as a video

camera. In general a digital image is represented as a rectangular matrix of brightnesses. This matrix is obtained by sampling the continuous image function produced by the sensor at equally spaced intervals in two perpendicular directions. The value of a point in the matrix is a quantised measure of the integrated luminous intensity of a small patch surrounding a corresponding point in the continuous image.

Enhancement / Noise reduction / Restoration These three terms refer to operations which, whilst performed for different reasons are often carried out by similar operations. Here the term enhancement refers to altering the image to make visual interpretation easier. Noise reduction techniques generally employ methods of suppressing high spatial frequencies using techniques such as Fourier transform filtering, neighbourhood averaging or median filtering. Restoration implies that the nature of degradation of an image during acquisition is known, and a mathematically inverse process is applied to achieve an approximation to the undegraded image.

Feature Enhancement Feature enhancement uses low level image properties to transform an image matrix to a *characteristic feature space* which is often in a similar matrix form to that of the original image. The values of the matrix serve as indicators of the likelihood of the presence of the features in question, e.g. edges.

Segmentation Segmentation is the term used for processes to which digital images may be subjected in order to divide the image into regions with similar properties. The intention of this subdivision process is to use low level image properties to obtain a first approximation to a description of the entities present in the domain from which the image was obtained. The most

important of these low level image properties are the homogeneity and discontinuity of the grey level amplitude. These properties are detected by the processes of region and edge detection respectively. By performing either of these operations a route to obtaining a description of the richest information carrier in images, i.e. *object boundaries,* is provided. Other image properties which offer routes to segmentation are texture and shading.

Object Description A segmented image's subsets contain implicit shape information which may be made explicit by, for example, employing some process to obtain linkage information from one boundary point to the next. This linked boundary information may offer some clue to the geometry of the object which gave rise to the boundary in the image.

Classification From the derived object descriptions, it may be possible to make decisions about the contents of the scene. The possible results of this sequence of processes are many and varied. For example control decisions may be possible, such as guiding a robot arm, or presenting valuable information to a human interpreter as in the automated diagnosis of medical X-ray images.

Routes to Obtaining Classifications

For a typical application which involves image processing the point at which the image processing stage is complete and the application routine takes over can generally be considered to be when classification and scene analysis has been performed. However, at this stage it is likely that a considerable amount of application domain knowledge has been used in analysing the scene.

The hierarchy of symbolic representations which may be derived from image features is extensive and ranges from, at the lowest level, the brightness

information of the image matrix itself, through primitive shape representations up to purely abstract names for real world entities. The decision as to where in this hierarchy a matching process for performing classification might take place will depend upon a number of factors, for example the level to which the contents of the image are known to be constrained, or the availability of a set of processes which can reliably produce descriptions from the image data of the level desired for matching. In general, the less constrained the domain of interest, the higher the level of representation used for matching.

As a result of this spectrum of representations there are two significantly different modes of processing used to achieve classifications. The first is to repetitively refine representations derived from the image of interest until a suitable representation for matching is achieved. The second is to hypothesise upon the presence of particular elements within a scene and to decompose a stored high level model of that element to attempt to perform matching with a relatively low level representation. The former of these modes of processing is termed *bottom up* or *data driven* processing and the latter is termed *top down* or *goal directed* processing.

In the limit one of these modes of processing would be used alone so that matching would be performed either at the pixel matrix level or at a highly symbolic level, however it is generally most efficient to employ a mixture of the two modes so that matching is performed at an intermediate level.

1.3.2 Knowledge Based Processing

A convenient method for implementing the top down approach to classification is the use of knowledge based processing techniques. These techniques employ processes for making inferences by evaluating the truth value

of propositions. One of the mechanisms of inference processes was documented by Aristotle more that 2000 years ago as the rule of *modus ponens*. The mechanism allows simulation of rational thought processes.

Propositions are used as hypotheses by an inference process. The inference process evaluates the true or false state of a proposition by evaluating conditional elements within that proposition. A knowledge based processing system therefore consists of a program, known as an inference engine, which performs the inference process, and propositions, or in the more general case, predicates, supplied as data expressed in a suitable form for manipulation by the inference engine. The most reliable and widely used method for the expression of propositions is in the form of production rules. These rules contain sets of conditions (antecedents) and actions (consequents, conclusions). Conditions are either true or false with respect to a given world state. If all conditions within a given rule are true then by definition the conclusions are true (i.e. the inference process may perform the appropriate actions).

One particular grammar for expressing rules, here written in Backus-Naur form, is :-

Rule ::= IF condition THEN action.

condition ::= clause | clause AND condition

action ::= clause | clause AND action

clause ::= attribute, predicate, value

attribute ::= text

predicate ::= text

value ::= text

where

::= means *is defined to be*

| means *or*

The attribute is the property of the entity about which information is available or desired. The value is the description of the characteristic pertaining to the particular attribute and the predicate is the way in which the value relates to the attribute.

An example of a rule in this form is :-

IF region_left_of_line is land AND
region_right_of_line is sea THEN
line is coastline.

An extension to this grammar might allow null conditions or action sets. A rule with no conditions would express facts and one with no actions could express illegal situations. It is only necessary to provide the AND conjunction between clauses. Situations in which a logical OR of condition clauses might be desired are best dealt with by creating a number of propositions representing each of the possible world states leading to the single set of conclusions.

Goal Directed Inference

Goal directed or *backward chained* inference is analogous to the top down approach to classification described in section 1.3.1. The approach involves the proposal of a main hypothesis or *main goal*. The inference process proceeds by searching a data base of production rules for a rule which has the main goal as its conclusion part. If the condition part of this rule can be proved true then the main goal is also proved true. Each of the conditions therefore become sub-goals which must be satisfied in a manner exactly analogous to that of satisfying the main goal.

When a sub-goal is proved false the condition set from which it arises may be considered false and a process of *backtracking* occurs, whereby an alternative condition set from another appropriate rule replaces the false condition set.

Data Driven Inference

Data driven or *forward chained* inference is analogous to the bottom up approach to classification described in section 1.3.1. In this mode of inference rules are sought for which the condition sets may be proved true directly from a database of facts. The conclusion part of these rules is used to enhance the database of facts. The inference process continues, utilising the increasing number of facts known, until no other rules can be proved true.

Chapter 2

Literature Review

2.1 Line tracking

The main emphasis of this literature review has been to identify existing techniques for feature extraction and in particular for extracting lines from digital images. Edge detection and image segmentation are clearly related to the problem and a review of these topics is presented in an extended review of the literature in appendix B.

Line images generally result from two types of scenes, those obtained from uncontrolled natural scenes such as satellite imagery of road, rail and river networks, and those from documents such as engineering drawings, printed or written matter and maps. These two categories will usually differ in terms of the amount of noise present, the contrast between lines and background and the nature of the background. Processes which extract linework from natural images, although being more generally applicable must therefore attempt to deal with the problems of removal of noise and the detection of low contrast grey level discontinuities. There is thus a significant division in the methods of line extraction described in the literature.

A further important division of approaches is between those processes which assume a prior knowledge of the characteristics of the lines to be extracted and those which do not. The division is not clear cut, as all line feature extraction procedures must incorporate some form of knowledge of line structure. However, for the purposes of this discussion we assume that use of prior knowledge of a lines structure implies that the line extraction process uses more than just a knowledge of the local topography of the brightness matrix in the region of a line, for example, searching for lines with geometry corresponding to a given parametric equation.

Whatever the source of the line image, routines to extract lines which exhibit loosely constrained meandering must take into account the facts that lines within a single image may be of different contrast and polarity, they may have inter and intra line width and grey level variations, they may contain discontinuities, intersect, be looped structures, leave the bounds of the image at one or more places or be wholly contained within the image.

2.1.1 Document Processing Surveys

Nadler [1984] has produced a survey of literature relating to document segmentation and coding which covers many aspects of the problem, concentrating on listing the relevant work and providing general comment on the state of the subject at the time. The section which covers work on the *raster to vector conversion* problem states that the results are *rather meagre*, referencing only two articles dealing with the problem. Much of the rest of this chapter is devoted to discussing a number of additional papers in which the problems of line extraction have been tackled, but in spite of various approaches there is still a clear need for a general and robust line feature extraction procedure.

With emphasis on achieving fast, cost effective optical character recognition

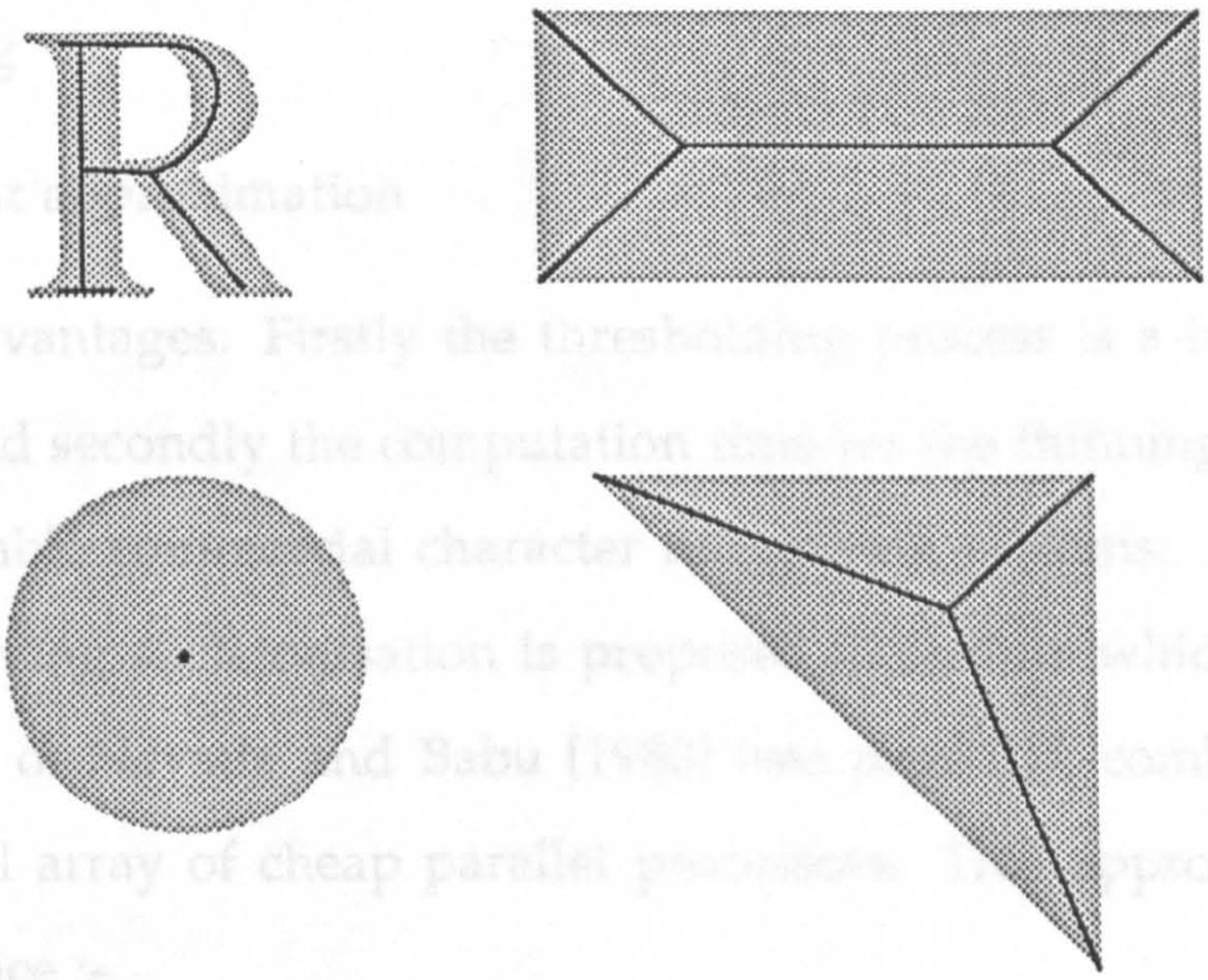


Figure 2.1: Image Segment Skeletons

Smith [1987] also surveys the processing of line images. The paper discusses means of achieving a skeleton like shape representation of characters (see fig. 2.1) and line segment approximation of line features. A number of processes for skeletonisation are documented which adopt the policy of thinning a binary line image by stripping edge pixels until a thin line representation is achieved. The iterative nature of these algorithms make them very slow. Faster skeletonisation can be achieved using edge information. In general these approaches form the skeleton of a shape from the central points of lines which perpendicularly intersect the two parallel lines created by edge detection of line features. With this type of approach the accurate detection of junctions becomes a problem.

In conclusion Smith states that for the most common methods of document processing, those which follow the sequence :-

- Thresholding
- Thinning

- Line tracking
- Line segment approximation

have severe disadvantages. Firstly the thresholding process is a highly non-trivial problem and secondly the computation time for the thinning operation is too high for viable commercial character recognition systems. A different approach to achieving skeletonisation is proposed in outline which seems to be similar to that of Nevatia and Babu [1980] (see page 21), combined with the use of a small array of cheap parallel processors. This approach would follow the sequence :-

- Edge detection
- Line tracking of the lines produced by edge detection
- Line segment approximation of these edge lines
- Symmetric axis projection, which uses the boundary information to produce a list of points which, when joined, form the skeleton of the object.

2.1.2 Tracking Without A Priori Knowledge of Line Characteristics

The references in this section all describe methods of performing line following using techniques which may incorporate constraints on the nature of the lines that may be extracted by the operation, but do not use knowledge of the characteristics of the lines to be found to guide the line finding mechanism in the same manner as the references in section 2.1.3.

Capponetti et al. [1982] describe a method for extracting linework from a binary image of a contour map. The routine scans the image in either vertical or horizontal raster mode until a pixel which is part of a line is detected. It tracks the line by scanning in the same horizontal or vertical mode, saving the

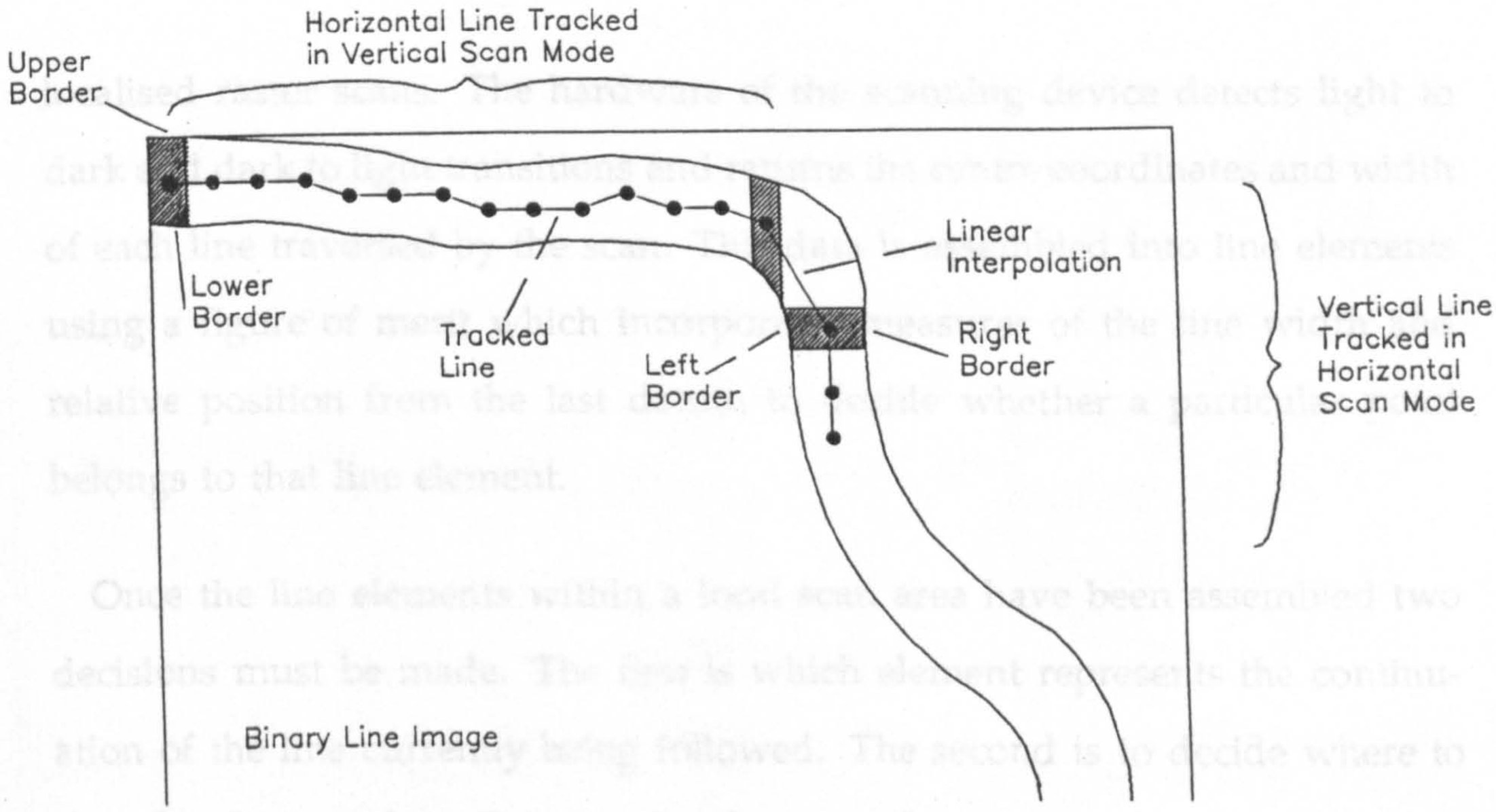


Figure 2.2: Line Tracker due to Capponetti et al.

centre coordinates of the line segment, until the line width exceeds a certain threshold. When this situation is encountered the mode of scanning swaps and the procedure continues until the line ends, leaves the bounds of the image or rejoins itself. The input image can be only of a restricted nature in that lines must not exceed a certain threshold width, and this threshold cannot exceed the average width of a line by too much. The lines can only intersect in certain allowed manners, also regions cannot be present. The authors suggest that a global thresholding technique should be employed at the acquisition stage in order to segment the image to give the desired line data. No attempt is made to account for degradation of the image during acquisition and preprocessing, therefore broken lines are not reconnected. The process is therefore one which is of use only for very constrained scenes and good quality imaging.

Fulford [1981] describes the hardware and software principles of the FAS-TRAK interactive line following system applied to the capture of cartographic data. Data acquisition is performed by a laser scanning device which makes

localised raster scans. The hardware of the scanning device detects light to dark and dark to light transitions and returns the centre coordinates and width of each line traversed by the scan. This data is assembled into line elements using a figure of merit which incorporates measures of the line width and relative position from the last datum to decide whether a particular point belongs to that line element.

Once the line elements within a local scan area have been assembled two decisions must be made. The first is which element represents the continuation of the line currently being followed. The second is to decide where to scan for the next line element. In choosing the continuation element, each candidate is ranked according to a figure of merit which incorporates measures of line width and direction and the gap between the line element and the end of the current line. If only one good candidate is found then the element is appended to the tracked vector, otherwise the operator is called upon to resolve the problem.

The choice of direction to scan is initially in the current direction of the track, then to left and to right and finally backtracking along the direction of the line to detect U turns. When scanning fails to produce acceptable line elements the operator is called upon once more.

The system finds no difficulty in thresholding to produce binary scan information due to the high brightness and resolution of the laser scanning device and the good contrast of the subject matter. A high degree of user interaction is required to drive the system. It is perhaps worth noting that for bulk conversion of maps to digitised form the Ordnance Survey attempted to use such Laser Scan techniques and, because of the substantial amount of user interaction, they abandoned it and resorted to wholly manual digitisation methods.

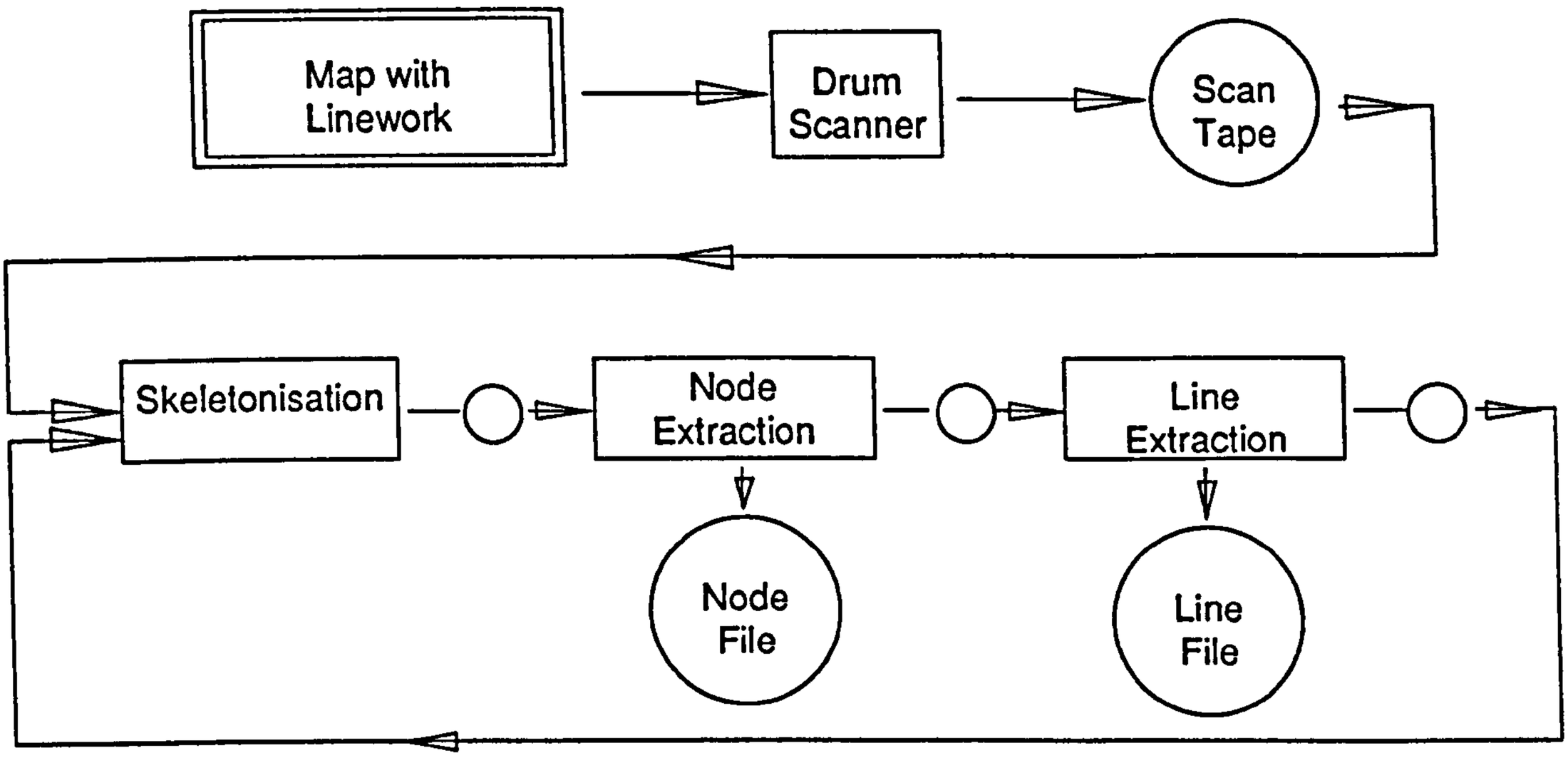


Figure 2.3: Line extraction due to Woetzel

The technique is, however, still undergoing continuous commercial development for increasing the level of automation of processing complex drawings and charts.

Ramachandran [1980] uses a technique for production of vectors from an engineering drawing which run length encodes a clean binary image of the drawing, preserving line width information. This code is then submitted to a vectorising process which produces piecewise linear segments of an arbitrary number of pixels in length. During the process, contiguous linear segments are cascaded if the combined vector differs from the individual parts by less than a specified tolerance.

Woetzel [1978] describes a system for deriving vector codes from maps whilst the original is being scanned on a drum scanner (see figure 2.3). Only $K+3$ scan lines need to be held in primary storage at a time where K is the maximum thickness of lines. The image is thresholded to give a binary representation and skeletonised. Each pixel in the skeletonised image is encoded

according to its environment; 0 = background; 1 = end point of a line; 2 = inner line point and 3 = node point. A node extraction process records the corresponding end point pairs of the line structure. The scan lines are then passed to a line extraction process to generate vector chains. The process requires expensive drum scanning equipment and relies on being able to extract a good quality binary image before skeletonisation.

Nevatia and Babu [1980] use six 5×5 directional edge detection masks to give an edge map consisting of the maximum output from the set of operators and the corresponding direction. This edge map is then thinned in a manner which produces a predecessor and successor matrix representing linked edge elements. From this linked edge map, boundary traces may be extracted and broken lines may be linked. Lines in the linked edge map are characterised by having long parallel edges of opposite polarity, which the authors termed *apars*, and which can therefore be extracted from the image. This particular approach has been adopted in a general vision system in use for vision research. This process is described further on page 56.

A novel approach to extraction of straight line edges is described by Burns et al. [1986]. The technique could be applied to line extraction in a similar fashion to that of Nevatia and Babu. They contest that edge *orientation* is a much more important source of information than it had been previously seen to be. In their process a gradient image is produced by applying two 2×2 edge operators to give edge magnitude and orientation information. The edge orientations are then grouped to form *line support regions*. Each pixels orientation is coded according to the angular range to which it belongs, e.g. for a scheme in which 8 orientations are allowed, the pixels in the range -22.5^0 to $+22.5^0$ would be labelled 1 pixels. Two such encodings are performed. For the above scheme the angular range for 1 pixels in the first grouping would

be centred on 0^0 and for the second grouping would be centred on 22.5^0. For each grouping connected sets of pixels of the same value are uniquely labelled. A single merged grouping is then obtained by the following processes :-

1. Line lengths are determined for each labelled region.

2. Since each pixel belongs to two regions, one for each grouping, each pixel votes for, and is associated with, the longest of the two regions.

3. Each region receives a vote count and is elected to join the merged grouping if more than 50% of its members vote for it.

This process is necessary to avoid the fragmentation and merging of edge segments characteristic of a grouping scheme based on a single arbitrary orientation labelling scheme. The underlying image intensity surface associated with each line support region is assumed to be a noisy representation of an ideal ramp. This intensity surface is approximated by a planar surface computed using a weighted least squares fit. The orientation of the edge segment may be assumed to be perpendicular to the line of maximum gradient of the fitted plane and its location may be calculated by, for example, constructing a horizontal plane at a height representative of the average grey level of the intensity surface weighted by the local gradient magnitude. The lines location is defined by the intersection of the two planes.

Having obtained the line support region, other characteristics of the line may be obtained to serve as input to an image understanding system. True straight image lines, rather than lines produced by edge detection, could be extracted using this scheme by searching for the intersections of the planar intensity surfaces of adjacent edge segments with opposite edge orientations.

Wójcik [1984] adapts procedures used for checking path widths in printed

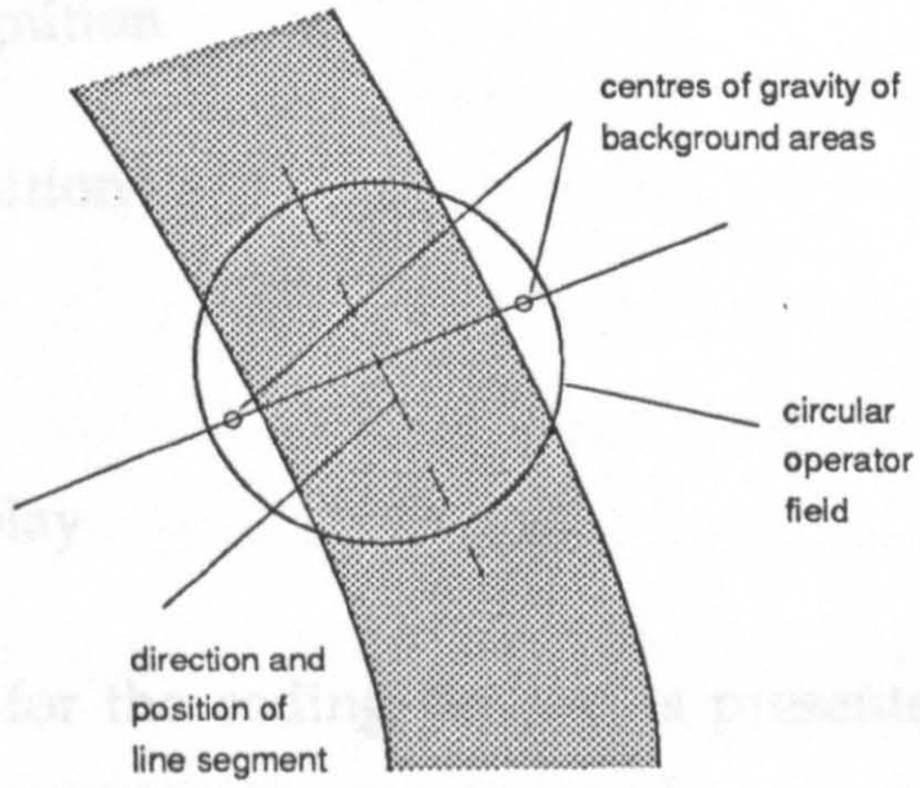


Figure 2.4: Line detection using a Circular Operator

circuit boards. Lines are extracted from a binary image using a circular operator of width slightly larger than the width of lines in the image. When two disconnected sets of background pixels are found within the operator field the position and direction of a line may be calculated using the centre of gravity of the two background areas (see figure 2.4). The position and direction of the line is given by the perpendicular bisector of the line between the two centres of gravity.

Pferd and Ramachandran [1978] outline the hardware and software needed for a variety of computer aided automatic digitising schemes (CAAD) for engineering drawings. The resolution requirements are defined for conforming with American and international standards. The overall strategy is :-

- Scanning
- Thresholding

- Encoding
- Character recognition
- Element recognition
- Editing
- Storage or display

A new algorithm for the coding process is presented which preserves all information in the thresholded image.

Bertolazzi and Pirozzi [1984] show how dynamic programming methods, which are computationally expensive when implemented on a single processor computer, can become useful when adapted to run on a parallel processor machine. They develop the concept by considering a machine with an unlimited number of processors and then show how their methods can be implemented on a machine with a limited number of processors. This allows a good deal of flexibility for their algorithm as it can be implemented on any of the current parallel machines with a reduction in computation time approximately proportional to the number of processors available. The sequential algorithm which they adapt is one which optimally detects a curved line in a noisy background using a figure of merit to obtain the optimal assessed line. Their figure of merit is the grey level at each point along the line minus a measure of the curvature at each point. Hence this technique is constrained to dealing with the extraction of lines which have predominantly smooth curvature.

Groch [1982] shows how an operator which classifies the profiles of grey level image functions into: valley, side, bottom and peak can be used to semi-automatically extract line shaped objects from images. This concept of profile

classification is extended in the paper by Haralick et al. [1983] in which a *topographic primal sketch* is defined. In this system the classifications are: peak, pit, ridge, ravine, saddle, flat and hillside, with hillside having non-invariant subcategories of slope, inflection, saddle hillside, convex hillside and concave hillside. The paper is aimed at defining a primal sketch for use in achieving computer vision. However, the detection of ridges by the methods described were put to use by Watson et al. [1984] in proposing a general method of extraction of lines and regions from grey tone line drawing images.

The process described in the paper by Watson et al. [1984] begins by smoothing the image with a gaussian filter. This has the effect of giving lines within the image a ridge like structure. The resultant image is subjected to a process which recognises a convex profile. Each pixel on such a profile retains its original grey level, otherwise pixels which exceed a threshold are set negative and all other pixels are set to zero. The resulting image is subjected to a routine which fills small gaps in regions and removes solitary region pixels. A line tracker is then used which searches the image in raster mode until a test is passed which requires three connected pixels to satisfy conditions describing a line beginning. When the beginning of a line is found the routine switches to line tracking mode. The line tracker works on the principle of walking on the ridge top in order to use the maximum amount of information contained in the grey levels of the image to achieve the best possible representation of the original line. This approach is unique in its incorporation of knowledge of the structure of lines in general rather than characterising lines by their edge properties or assuming that lines may be segmented by histogram analysis techniques. A modified version of this algorithm is described in detail in section 3.24.

A formal derivation of an optimal edge operator is described by Canny

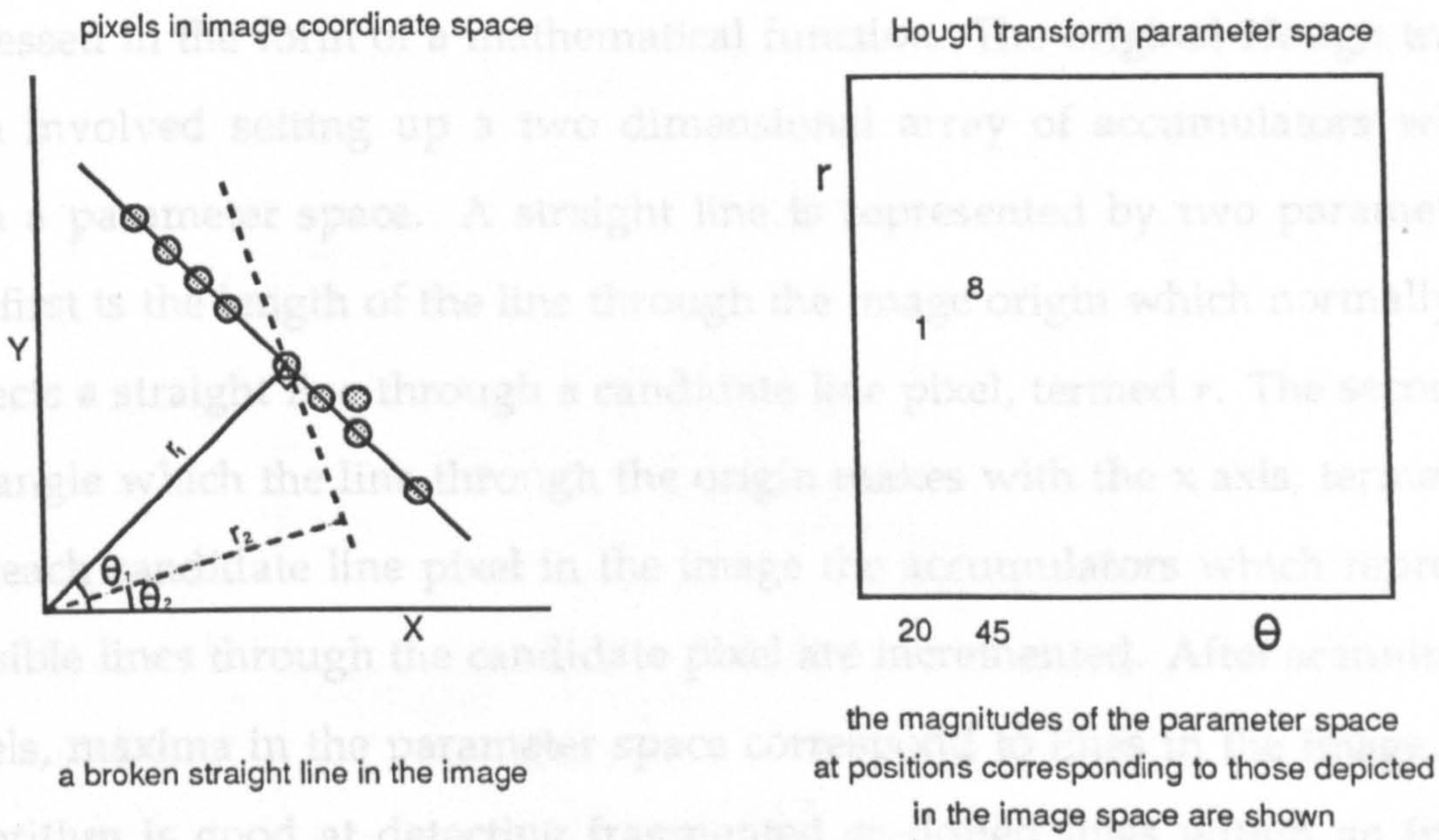


Figure 2.5: The Hough Transform

[1983] based on the premises that an operator should give :-

- good localisation of edges
- a high probability of detecting edges
- only one response to an edge

A more thorough review of this paper is given in the extended literature review on page IX of appendix B of this thesis. The importance of this work in terms of line detection is that Canny points out that a similar derivation may be performed for deriving the optimal ridge profile detection mask.

2.1.3 Tracking Using A Priori Knowledge of Line Characteristics

A well known example of a line extraction process which assumes prior knowledge of the line characteristics is the Hough transform (see figure 2.5). Duda and Hart [1972] describe the concepts of the Hough transform which detects straight lines and then extend it to detect any type of line which can be

expressed in the form of a mathematical function. The original Hough transform involved setting up a two dimensional array of accumulators which form a parameter space. A straight line is represented by two parameters. The first is the length of the line through the image origin which normally intersects a straight line through a candidate line pixel, termed r. The second is the angle which the line through the origin makes with the x axis, termed θ. For each candidate line pixel in the image the accumulators which represent possible lines through the candidate pixel are incremented. After scanning all pixels, maxima in the parameter space correspond to lines in the image. The algorithm is good at detecting fragmented or dotted lines within an image, but does not give spatial localisation of the extremes of a line. The computation time and memory requirements are high. Duda and Hart's extension involves increasing the number of dimensions in the parameter space in order that lines represented by functions of higher order, e.g. circles or ellipses, may be detected.

Whilst investigating the application of techniques originally developed for analysis of bubble chamber film analysis to the analysis of pen-recorder charts, Black et al. [1981] found that for each new line following application, new software had to be developed. A general purpose line follower is described for which the overall strategy is defined after compilation by a set of control parameters loaded from a disk file. Image data acquisition is performed by a flying spot scanner (PEPR) which creates a binary image representation in hardware prior to software analysis.

A four stage model of general line following techniques was developed. The model assumes that some data has already been collected about the start of a line. The model cyclicly repeats the following sequence:-

- Predict a new Trace Point and Direction

Prediction	linear circle user supplied
Scan	slice scan area scan skeletonising scan user supplied
Analysis	histogramming continuity linking minimum spanning tree user supplied
Addition	simple convert to regular x spacing user supplied

Table 2.1: Algorithms for each Stage of the Line Follower due to Black et al.

- Acquire image data at this point
- Analyse scan data
- Add new trace data to Existing Trace Data

This sequence is ideally repeated until the end of the line is encountered. A number of algorithms are identified for each stage of the model (see table 2.1). Selection of the most appropriate algorithm to perform each stage is made by defining the *home state* of the system prior to execution by defining the system parameters in the control table. This will generally be the least computationally expensive set of algorithms which can reliably perform the line following for the type of line under consideration. Apart from the home state an ordered set of *alternative states* may be defined which will be invoked when the home state or the previous alternative state fails to perform adequately.

Table 2.1 gives examples of the algorithms employed by the system for each stage of the model. The most appropriate parameter set for a number

of classes of line images has been tabulated in order that a first choice may be made by the user of the system by selecting the parameters of the test image best resembling the new type of image. In this fashion the user of the system is providing a priori knowledge of the line characteristics to the system. However, as documented within this paper, each of the algorithms proposed for performing the various stages of the model contain little or no knowledge of the characteristics of the particular line to be found, but instead rely upon more computationally expensive procedures which can be brought to bear when particularly adverse situations are encountered.

Yachida et al. [1979] describe a system for finding lines in noisy image regions using a priori knowledge (figure 2.6). The suggested scheme for using their line finder is that an alternative extraction process extracts the lines that are easy to find. A high level operation fits models to the linework. When a fit has been hypothesised, lines which the model suggests ought to be present but are missing from the image may be searched for by the line finder described. Knowledge of the approximate starting point, search region and line characteristics such as edge, grey level and curvature properties are supplied to the line finder in the form of data. The characteristics of the line are described by selecting terms from a merit function menu. Each term describes a particular characteristic and may be weighted according to its importance. The merit function calculates a global measure of how a given line conforms to the characteristics described in the merit function by summing the contributions made by each individual part of the line.

The scheme used here is that individual parts of the line are line segments rather than pixels. This has the effect of reducing the systems susceptibility to noise as well as reducing the computation time and storage requirements. The line finder searches the starting region for promising start points. The

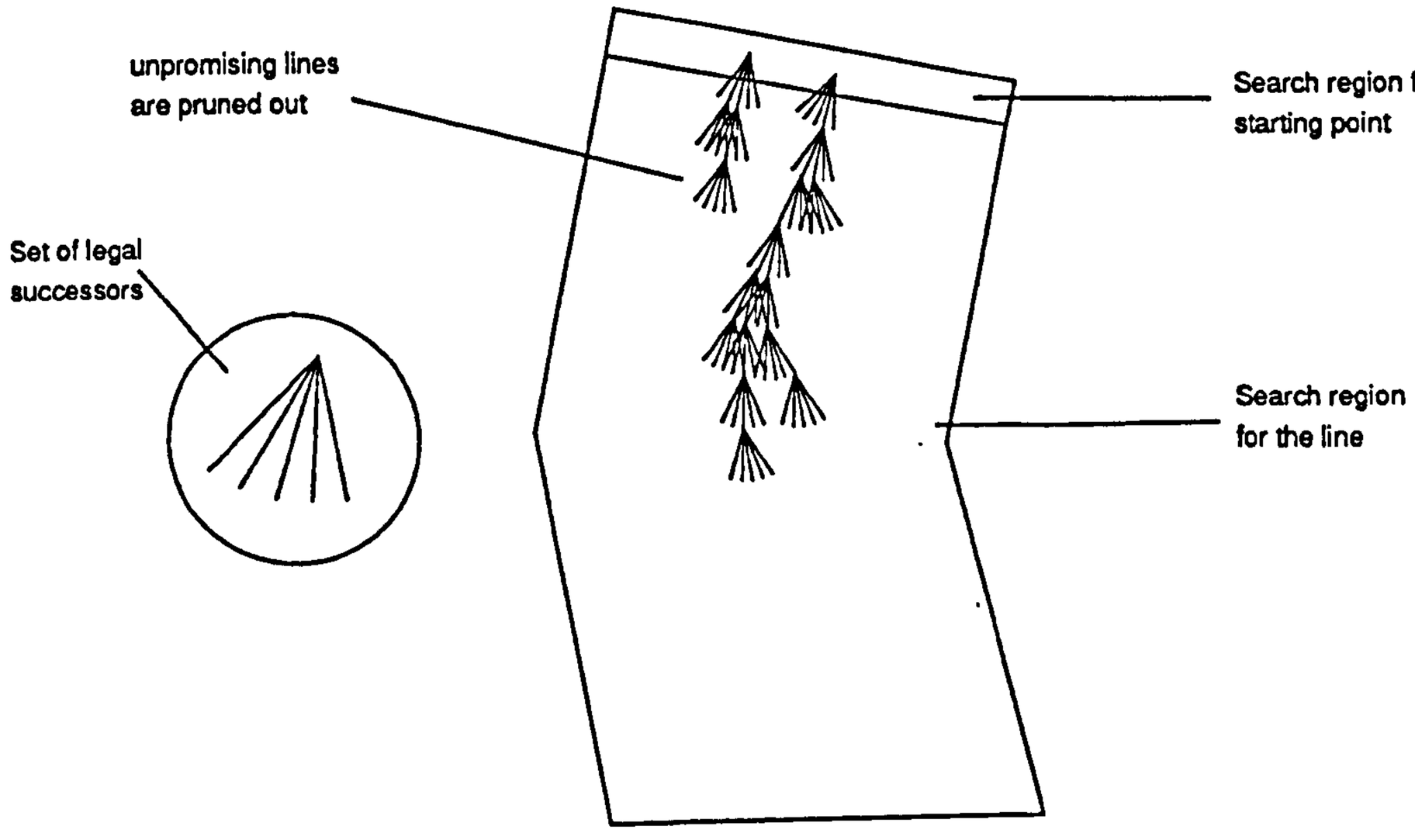


Figure 2.6: Line extraction due to Yachida et al.

direction of continuation of lines is limited to a predefined angular region within which a limited number of directions are tested. Unpromising lines are pruned out as the search progresses. This also reduces the computation and storage requirements whilst still finding a near optimal line. The scheme is well suited to finding lines in noisy environments but the computational expense is high.

2.2 Map Interpretation

Some initial work towards an intelligent map interpretation system is described by Amin and Kastura [1987]. A simple line drawing map image of 512 x 512 pixels resolution is segmented by global grey level thresholding and thinning. The thinning process labels pixels according to their connectivity i.e. background, free end, line link and junction. The line segments

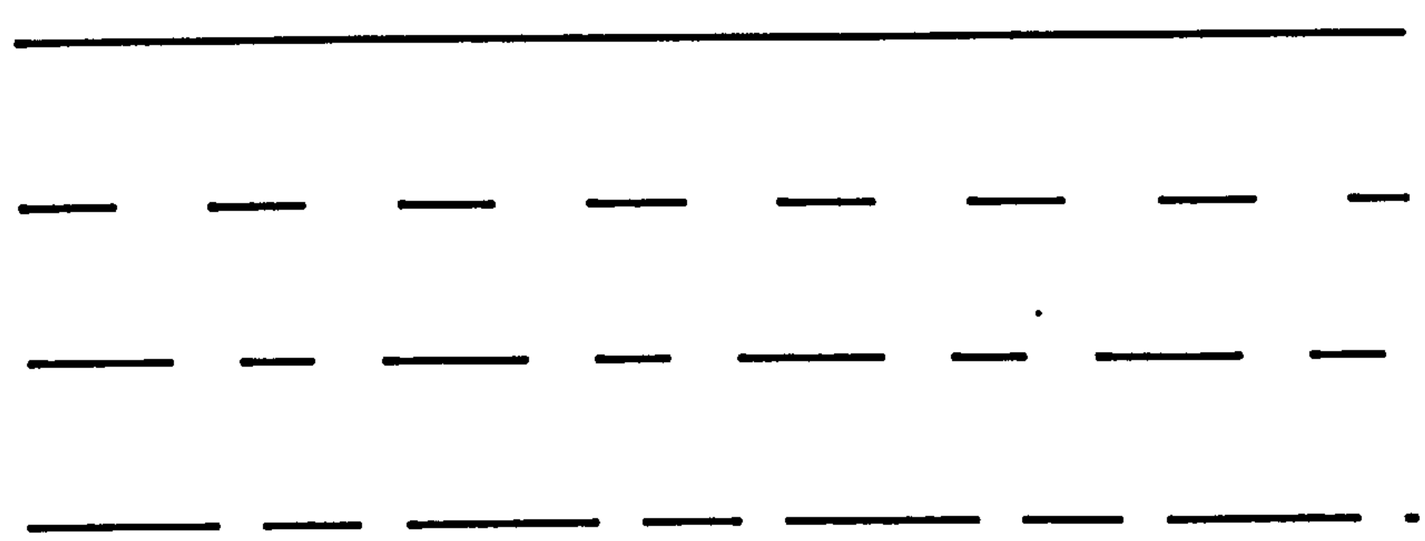


Figure 2.7: Line types recognised by Amin and Kastura's system

are subjected to post processing in order to classify lines and symbols. Four types of lines are recognised (see figure 2.7), one continuous and three types of dashed line. Continuous lines are identified by using a length threshold. The segments forming dashed lines are grouped into individual lines according to the *inter-link gap* and segment slope. Lines which begin and end in *junction* pixels are tested to see if they represent symbols. Circular symbols are recognised by having an area to square perimeter ratio greater than 0.7. Other symbols which can be identified are rectangles, ellipses and triangles. These are classified on the basis of the number of significant slope changes in the perimeter of the symbol. For example a triangle may be classified by having three major slope changes. The thresholding and thinning of the image makes many of the symbols appear ragged and distorts the representation at junctions of lines with symbols.

A scheme for extracting and classifying features in national large scale maps of Japan is described by Suzuki et al. [1987]. The national standards for production of maps define that the maps are produced in black and white and that the line widths for each feature type is fixed. For example, buildings

are drawn according to the following specification. If part of a building casts a shadow caused by solar radiation from an angle of 135^0 then that part of the buildings outline is drawn with a width of 0.3 mm, otherwise it is drawn with width 0.1 mm.

A labelling algorithm takes a high resolution binary image matrix (16 pixels / mm) and labels each line pixel according to the width of the line within which it resides. Four line widths are allowed. Building outlines are extracted using the knowledge about the drawing of buildings previously described. The image is then thinned to give a medial line image and the outlines of buildings are deleted to avoid erroneous extraction of features. The system then classifies railways and contour lines primarily on the basis of their line widths. The system can only work on map images which are constrained by such regulations. A good quality binary digital image is assumed to be available in order to achieve accurate line width calculation.

2.3 Representations for Map Data

Freeman [1970],[1974] and [1977] produced a number of early papers on the subject of line drawings as a method of communication. He reviews the ways in which line drawings can be represented and discusses the Freeman chain code in depth, comparing it to other methods of encoding line structures. Finally he goes on to describe how coded lines can be expanded, rotated and smoothed. The Freeman chain code is based on the concept of connected neighbours. The code can consider just four pixels as being adjacent to a central pixel, or more commonly eight, or it can be extended to cover larger neighbourhoods. A boundary or line is described by a sequence of numbers corresponding to the direction from one pixel on the line to the next. Extensive facilities are described for including extra information within the code. The

codes of lines or boundaries may be used for correlation procedures in pattern recognition and classification operations. Saghri and Freeman [1981] perform an analysis of generalised chain codes i.e. codes derived from considering arbitrary sized neighbourhoods. They show that the average quantisation error, which is a measure of the codes precision, is directly proportional to the grid size of the matrix imposed upon the natural image. This implies that the precision of representation of a line drawing may be selected solely by the selection of the grid size, allowing the selection of the type of code to be based on other criteria such as compactness or ease of encoding and processing.

Merrill (1973) describes a data structure for storing the boundaries of regions in a form suitable for efficient computer search. The paper defines a *tightly closed boundary* in which the x coordinates which lie on the border of a closed boundary are grouped into sets of points with the same y coordinate, known as a *y partition*. The search strategy is then based upon the principal that for a horizontal line which starts outside the boundary and goes to a point (x,y), the point (x,y) is contained within the region if the border is crossed an odd number of times. A point on the boundary at a local maximum or minimum must be repeated to preserve the above property, as is the case when an inflection contains an odd number of points.

These concepts are extended by Miller and Iyengar [1983] in order to reduce the computational expense of computing the intersections of regions from a number of data sets of the same area. Using Merrill's scheme the intersection of regions is computed by exhaustive odd/even calculations. Miller modifies the structure by sorting the y partitions in a given image row into order according to the last x value in the partition and including the *ending x* notation which allows a simpler and less computationally expensive process for compiling region intersections. This form of storage would not allow individual

features within an image to be stored as a single entity.

Middleton and Taylor [1985] take map data stored in the form of chains of cartesian coordinates which form line segments and subjects them to a three stage compression. The first is a requantisation stage. This is followed by using a model to sequentially process the chain of data and, from a few connected points, form a prediction of the next point's coordinates. The *error* in the models prediction is stored. The line can be reconstructed using the model and the stored error data. These errors, if the model is good, will have strongly varying probabilities of occurrence, with small errors being common and large errors rare. This type of data is well suited to encoding using a variable length code for which the elements of the source alphabet with the greatest probability of occurrence have short code lengths and those elements with low probability of occurrence have long code words. The most optimal form of this type of coding is Huffman coding, however Middleton uses the slightly less optimal Shannon code. The stored data is highly compressed and is suitable for processing on a small word length computer. This highly compressed format is required for this work because it is aimed at packing as much geographical information as is possible into a hand held battery powered computer. The computational expense of the decoding process may not be justified when hardware constraints are not quite so rigorous.

Samet and Webber [1983] and Samet et al. [1983] discuss a number of requirements of the quadtree structure to facilitate efficient representation of polygonal maps. In the latter paper they consider various refinements in terms of point in polygon search tasks to demonstrate the data structure's attributes. The quadtree has been shown to offer useful data compression ratios for images in which there are large uniform regions. It has also been shown to offer an alternative data structure for pattern matching processes.

The method of addressing video output terminals by defining rectangular areas with given brightnesses makes the quadtree a good data format for fast display of images. Unless point, line and region quadtrees are used feature data is not explicitly available, and even with these structures features are not stored as single entities. Geometric calculations on quadtree structures require relatively complex tree traversal operations.

2.4 Rule Based Systems

The field of artificial intelligence has provided image processing and vision researchers with a framework for attempting to perform high level visual perception tasks. The use of inferential processes as an aid to understanding reasoning has been practised since Aristotle (see e.g. Sell [1985]). Aristotle proposed the inference rule of *modus ponens*.

If P implies Q and P is true, then Q is true.

Sell describes the operation of forward and backward chaining inference mechanisms and points out that the main use of a forward chaining system is in an *embedded* rather than a *conversational* system. An embedded system is one which is part of a larger conventional system and the suitability of forward chaining here is that all the information that is needed or ever likely to be given is available at the start of the process as one set.

Ballard and Brown [1982] give an account of predicate logic and point out that rigorous application of classic inference mechanisms have limitations which can, to a certain extent, be overcome by employing processes of inference which do not embody the full essence of the logic discipline. The general difference in these extended inference systems is that an interpreter that controls the inference process in special ways is usually an integral part

of the system. Relational structures which may be derived from input data or stored as templates for matching are also discussed along with a description of matching processes. Semantic nets provide a useful method of describing interelationships between and characteristics of scene elements. They are suitable for manipulation by inference processes but the matching task throws up a few problems in terms of the computational expense and the control of non-rigorous matching using suitable metrics to reflect the quality of a particular match.

The text also covers planning schemes using inference, such as are needed when attempting to implement robot vision, when actions can change the world state and the consequences of many possible actions must be considered before selecting the appropriate one in order to achieve a given goal state. Davis [1980] describes how the inference process (or indeed knowledge embedded in procedural code, theorems or any other form of knowledge representation) can be controlled by using knowledge about knowledge. Within an inference process the expression of this *meta-knowledge* may be in rule form within the rule data base. These are then termed meta-rules. Such rules allow dynamic priority ranking of the rules which embody *object level knowledge.* The problem considered in this work involves the effective application of knowledge when the knowledge base is very large and chunks of this knowledge are likely to be of little importance under certain situations or when many knowledge sets could be applied to achieve a given goal. The meta-rules define strategies of processing. By including the meta knowledge in the form of rules the same inference engine that deals with the object level knowledge can perform the problem solving task of refining the order of the various knowledge sets within the knowledge base.

Quite justifiably, many workers have limited their research interests to problem areas which are intended to contribute to an individual processing stage in an image interpretation system, assuming that the preprocessing required to form the input data will be developed. However, significant problems have been encountered in the development of robust preprocessing algorithms to provide reliable sets of image primitives. An attempt to consolidate approaches to initial image segmentation is described by Nazif and Levine [1984]. Their approach is to use an expert system for low level domain independent image segmentation. It uses knowledge about images in general, e.g. segment grouping criteria. The system uses the original image, its gradient and initial line and region maps as the data for interpretation. This data set is enhanced by a set of low level descriptions of attributes of the image segments, e.g., geometry, statistical properties and relative positions of the image segments. Rules are divided into sets which deal with :-

- refining the segmentation
- controlling the focus of attention (in terms of image area and segments) and ordering knowledge rules
- strategy rules which establish the flow of control by specifying the next process to be activated (These essentially reorder the control rules)

The system offers a well developed common sense approach to resolving the problems and deficiencies encountered in low level image processing.

McKeown et al. [1985] describe a rule based approach to interpretation of aerial imagery of airport scenes. The airport interpretation task is knowledge rich because the functional relationships between structures such as runways and hangars etc. preclude arbitrary spatial arrangement. Classification rules use the segmented regions of an aerial image of an airport to create *fragments*

for which the classes are - small blob, large blob, linear and compact. Each of these classes contain subclasses relating to domain objects, e.g. access road, taxiway and runway are the subclasses of *linear*. A single region might give rise to many fragments with different classifications each with an initial confidence value for the classification. Further fragments may be generated by recognising aligned linear segments and submitting these new regions to the classification rules.

Next, fragments with subclass interpretations are checked for consistency, i.e. that their spatial arrangements comply with their function. Each test alters the confidence value for the classification. Certain subclasses are *seed fragments* for *functional areas*, e.g. terminals, access roads, runways and hangars all provide evidence for the utility of the areas in which they occur. When at least one functional area of each type has been created then an *airport modelling* process begins. When a large number of functional areas are created, multiple alternative models are generated. The modelling phase analyses inconsistencies and overlaps in functional areas and can invoke image analysis routines to look for hypothesised regions.

A data driven rule based approach to segmentation of coronary vessels from digital subtracted angiograms is described by Stansfield [1986]. An edge segmentation, provided by a simplified version of the Canny operator, is used to refine a region segmentation. The resultant segmentation is processed to provide trapezoidal region representations which are the primary region representations used by the rule based system. The rule based system operates in two phases. The first deals with domain independent knowledge, establishing relationships between segments such as *adjacent* and *parallel* as well as performing linkage operations on segments. Processing constraints are achieved by maintaining *active goals*. For example, a rules condition set would contain

a clause such as :-

If there is an active goal to run trapezoids....

The second stage of processing by the rule based system uses domain dependent knowledge to label segments as either vessel or noise. This labelling is simply based upon assuming that the largest trapezoid is the aorta. The system uses the adjacency of segments to navigate its way round the image until labelling is complete. Stansfield makes the point that whilst separation of the domain independent and domain dependent knowledge is in principle is a good idea, it can lead to problems. Also the use of data driven production rule based systems does not provide an adequate control structure and a more complex control structure must be developed.

An image understanding system for the knowledge based identification of artery branches of cine-angiograms is described by Tsuji and Nakano [1981]. The system interprets line segments, assuming that some preprocessing stage has been able to supply *all* line segments corresponding to arteries. The knowledge base is segmented into control rules, identification rules and judgement rules. The control rules define which artery is being considered at a given time. The identification rules are segmented according to which artery they attempt to identify. Multiple hypotheses for identification of each artery are built up as an hypothesis tree. Each hypothesis has an associated certainty factor. Hypotheses for several images from different views are created and integrated. The judgement rules are used to decide which set of hypotheses are most likely to be true. The description of the system concentrates on the medical domain knowledge. Some user interaction is required with the system, e.g., to point out where the catheter that injects x ray opaque dye into the blood system is located.

Chapter 3

Preliminary Chart Processing and Implementation of a Basic Line Tracker

3.1 Research and Development Plan

The idea of generating a digital data base of sea charts was formulated during conversations with a local company involved in marine navigation instrumentation. The company expressed their need for a computer data base which could provide a pleasing visual display of the information in sea charts and the necessary data for input to automatic navigational algorithms. The Hydrographic Office's records of depth data consist of paper records of spatially irregular sets of depth soundings, produced over a number of surveys.

The possible non-automated routes to producing an initial data base from existing information are to manually transfer the depth data and other associated information onto computer storage or to hand digitise the information in paper sea charts using a digitising table. This latter approach has been widely

adopted for digitising maps and other line drawings. For display purposes alone, some form of storage of raw images of sea charts themselves, captured e.g. by a television camera, might provide a suitable data base. The basic nature of the process of rapid information retrieval for display purposes and the information processing tasks required for navigational operations impose certain conflicting requirements in terms of the format of data storage, for example some formats of data storage are suitable for fast retrieval and display but contain no explicit shape information.

An initial plan of research was developed for which the goal was to produce a data base in which coastlines and depth contours are stored as vector coded information. The individual map features should be coherent labelled entities. The first step would be to develop image processing techniques to provide an initial conversion of coastlines and contours in digital images of sea charts to some form of vector coding. It could not be expected that a basic domain independent line tracking operation could reliably segment chart features. For example it could not be expected to account for actual breaks in the graphical information where a depth contour is broken in order to mark it with a depth indicator.

The human viewer of sea charts subconsciously performs complex interpretations. Apart from low level visual interpretations such as accounting for illumination gradients across the charts, the viewer uses a large amount of knowledge of line drawings and sea charts. The viewer perceives the image as an organised configuration rather than as a collection of independent parts, this is the *gestalt* phenomenon. It was apparent that this knowledge would have to be identified and incorporated into some stage of an automated interpretation process. It seemed that the nature of the knowledge involved in assembling the individual parts of an image into organised configurations

could not be incorporated into the essentially local decision making processes involved in line tracking. The plan of research was therefore to achieve as much interpretation as is possible with an automatic low level tracking process and then to attempt to collect the fragmented data so provided into features in a later process involving human interaction wherever necessary.

The next stage in the plan would be to identify the knowledge used by the human interpreter. The subconscious nature of human interpretation makes eliciting this knowledge a problem. A convenient formalism for expressing this knowledge is in the form of rules of thumb, this being one of the easiest ways for the human to express the processes of reasoning and allows direct translation into production rules for implementing knowledge based processing techniques.

With the advantages of knowledge based processing techniques in mind it was considered that one way of making useful knowledge explicitly available would be to develop a semi-automated procedural interpreter in which the user's knowledge could be utilised to guide the interpretation process. This might then lead to gradually increasing the level of automation of the system by providing an intelligent knowledge based system (*IKBS*) to work interactively with the procedural system to assist in the decision making processes. Requests for user guidance would be formulated as goals for the IKBS and only if the goals could not be satisfied would the user interaction be solicited.

3.2 Equipment Available to the Project

3.2.1 Original Facilities

The facilities available at the outset of the project to carry out this research plan were :-

- a D.E.C. Micro PDP 11/ 23+ host computer, with Micro RSX 11M+ operating system, advanced programmers utilities, 10 MB hard disk, dual 400 KB floppy disks
- a Gresham Lion Supervisor S214 off line image capture device and frame store,
- a Fortran 77 compiler
- a library of some basic image processing algorithms written at Leicester Polytechnic,
- a Link 109 B video camera,
- a D.E.C. VT100 monitor,
- a D.E.C. LA 50 printer,
- and a Digivision video monitor.

The Gresham Lion image capture and frame store device provides the image acquisition facilities. In its original configuration it was controlled from the host computer over a serial line by sending mnemonics and associated parameters which caused the device's processor to execute imaging and graphics routines. It is capable of storing an image of 1024 × 512 pixels. The device accepts an analogue signal from the video camera and a fast analogue to digital converter can convert an image into digital matrix form of up to 512 × 512 pixels with 64 discrete grey levels in 0.04 seconds. The image memory can be configured to a number of predefined image resolutions, for example eight images of 256 × 256 pixels can be stored. Images of up to 512 × 512 pixels may be captured from the Link video camera. Image data could be transferred to the host, row by row, over a serial line running at 9600 baud. The live video

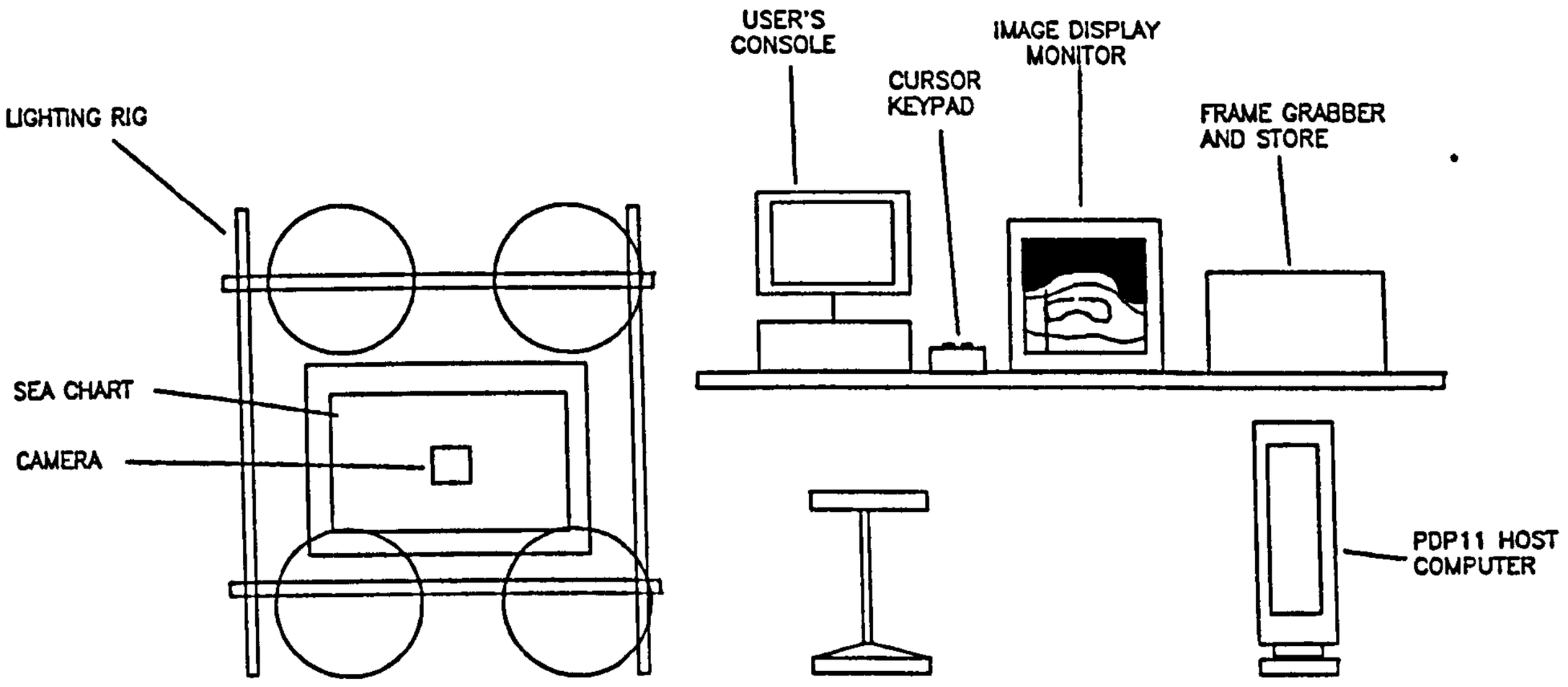


Figure 3.1: Block Diagram of the Equipment for Image Processing

image or captured image is displayed on the Digivision monitor. The VT100 monitor and LA50 printer are used for programming and for interaction with the various software tools.

3.2.2 Problems with the Development Environment

The reasoning behind acquiring this particular configuration of equipment was that a similar development environment had been established at Leicester Polytechnic for some time and it was from this establishment that some initial image processing software was obtained. This software consisted of a menu driven tool box of image processing routines and image handling utilities written in Fortran. The Leicester Polytechnic Image Processing group had their imaging equipment attached directly on line to a PDP/11 45. Certain problems have been encountered during this project which have affected the research methodology employed, and for this reason are documented here. Because of the limited processing power available it has been necessary to exercise discretion when considering the implementation of procedures, peripheral to the main avenue of research, which might otherwise have been investigated in order to demonstrate their suitability or unsuitability for the task in hand. Wherever such decisions have been made the reasoning is documented within this thesis.

The Micro RSX 11M operating system is a multi-user, multi-tasking operating system. The maximum addressing capability for a user's task is 64 Kilobytes. With hindsight it is possible to suggest that this host / operating system combination was perhaps a non-ideal choice for the task in hand. For the duration of this research project the system has been entirely dedicated to image processing with only one user. The core memory capacity is such that the execution of a task of significant size excludes further tasks from running

simultaneously. Much of the 64 KB task space is taken up by input / output subsystem requirements and efforts to reduce this overhead have been unsuccessful. Much effort has therefore been expended in tightly and efficiently packing programmes into the available memory. Memory overlaying of tasks, a laborious and time consuming affair, has been used extensively for all routines written for the project. The large quantity of data involved in image handling has meant that significant efforts have been expended in buffering the image data using the Gresham Lion frame store and the host's secondary storage resources. The declaration of virtual arrays is theoretically possible with the system. However, the maximum size of such arrays is less than the size of a working image and the declaration of virtual arrays approaching this maximum size brings the system grinding to a halt.

The *off line* configuration of the image frame store was also a handicap in terms of program execution times. Each 256 pixel image row transfer operation from frame store to host took 0.3 seconds, hence if each row of a 256×256 pixel matrix was read and written only once in a processing operation then the data transfer process alone took over 2 minutes to perform.

One year into the project the system was upgraded in the following manner :-

- The host computer's central processing unit was changed to the Micro PDP 11/ 73.
- The imaging device was put directly on line to the host computer.
- The storage capacity of the imaging device was increased to allow 256 grey levels per pixel.
- A pascal compiler was obtained.

- The bulk storage capacity of the host was increased with the addition of a 31 MB hard disk.

- A lighting rig was set up to allow even illumination of the sea charts.

- The kermit file transfer package was installed to facilitate communication with the Institutes general purpose computing facilities and with other machines via the Packet switching and Janet networks.

As a result of this upgrade the host processing and image data transfer speeds were increased considerably. However the necessity for buffering of image data, whether it be resident in the frame store or the disc drive, remained. To give an example of the handicap which this slowness has proved to be, one particular preprocessing operation (Nevatia and Babu [1980]) which involves 6 5 × 5 convolutions of an image takes over 2 hours. The same routine which has since been implemented by a colleague at Southampton University required considerably less time dealing with machine considerations and takes less than 2 minutes to complete on a Sun 3/50.

3.3 Exploratory Processing of Charts

3.3.1 Capturing and Cleaning Images

In order to ensure that the best possible images were obtained as a starting point for processing, a lighting rig was designed and built to allow even illumination of the chart. The lighting rig is illustrated in figure 3.2. The four large photoflood lamps can be moved independently laterally and as pairs vertically. Individual dimmer controls for each lamp allow fine control of the lighting balance which is checked with a reflection photometer with 1^0 angle of view.

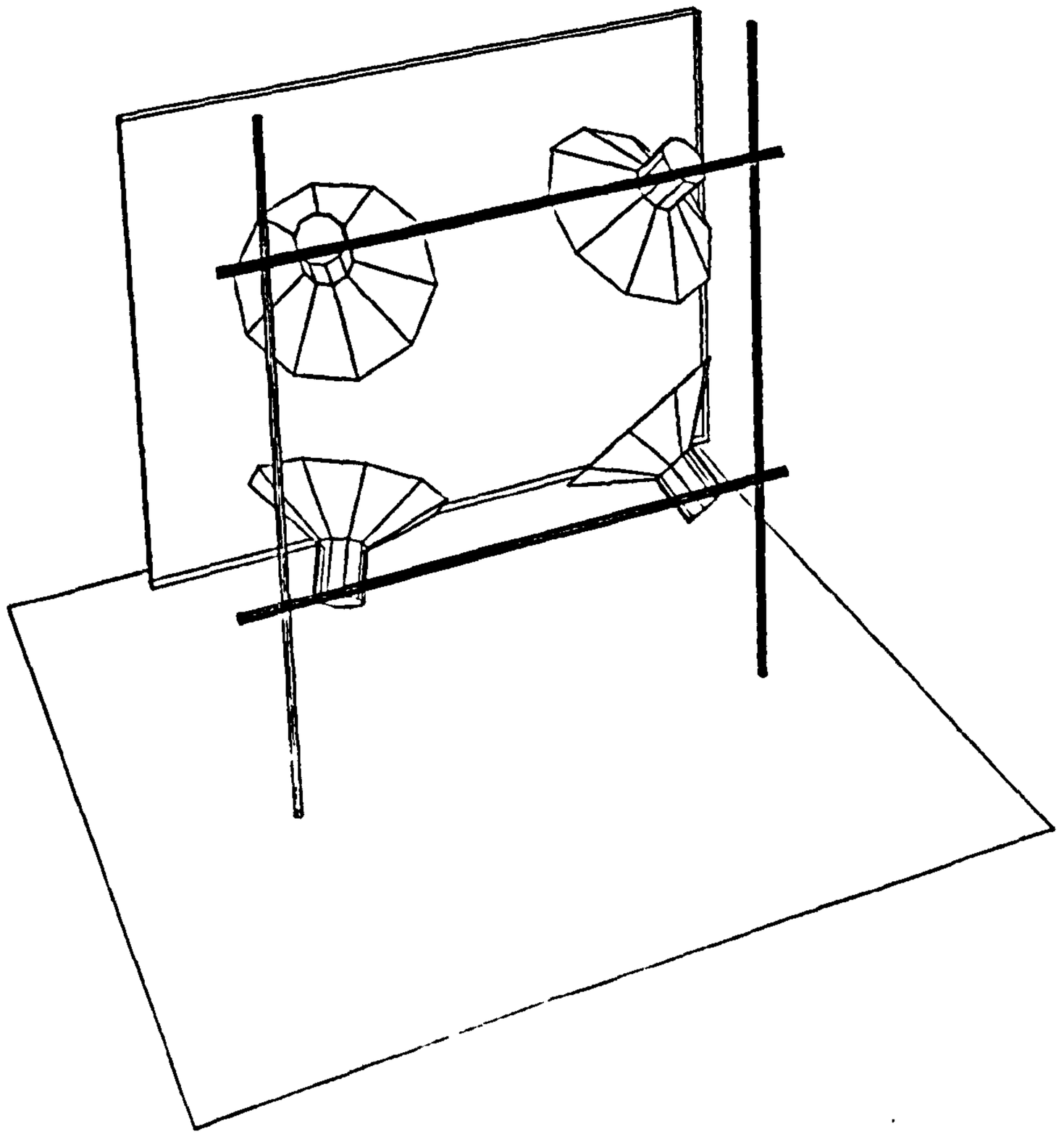


Figure 3.2: Lighting the Chart

In spite of the careful approach taken in order to ensure capture of a clean image the images obtained inevitably suffered from some noise and slight variations in illumination. A raw captured image is shown in figure 3.3. It shows a 256 × 256 pixel image with 64 levels of grey tone of part of a hydrographic chart. The chart was of scale 1:150,000 and this image corresponds to a chart area of 4″ × 4″ . It was captured using a monochrome video camera. This provided a rapid way of capturing a chart image for research purposes. A more advanced automatic digitising technique could be used to digitise whole charts for analysis when the appropriate techniques have been developed.

Basic techniques for cleaning the image were then explored. Noise reduction can be achieved by applying a median filter before applying any feature detection process. The results of applying a median filter to the image in figure 3.3 are shown in figure 3.4 and 3.5. The first image shows the result of selecting the true median from 3 × 3 regions within the image. The small scale of the line widths within the image effectively makes this a thinning operation on the lines. Selecting an element from the 3 × 3 region of lower value than the median counteracts this to some extent, and indeed can broaden the line widths. However the noise reduction properties of the median filter are reduced by this adjustment. The median filter was tested by using it before performing the line tracking process detailed below. The result was that the broadening of lines made the line profile unsuitable for the subsequent processes.

3.3.2 Image Histogram Properties

Some histogram analysis of the sea charts was carried out to see if global grey level thresholding techniques could be used as a basis for segmenting

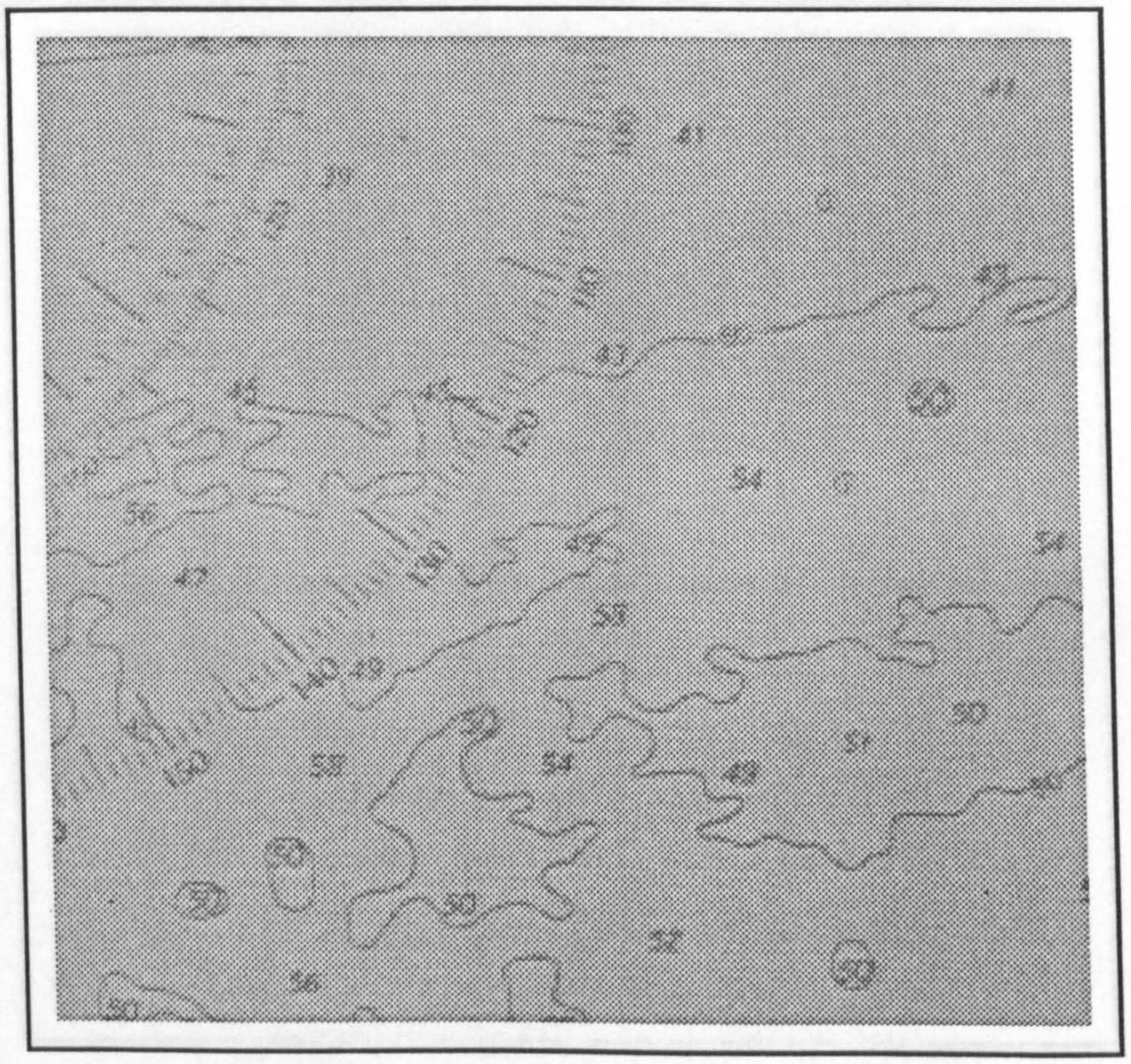


Figure 3.3: A raw captured image

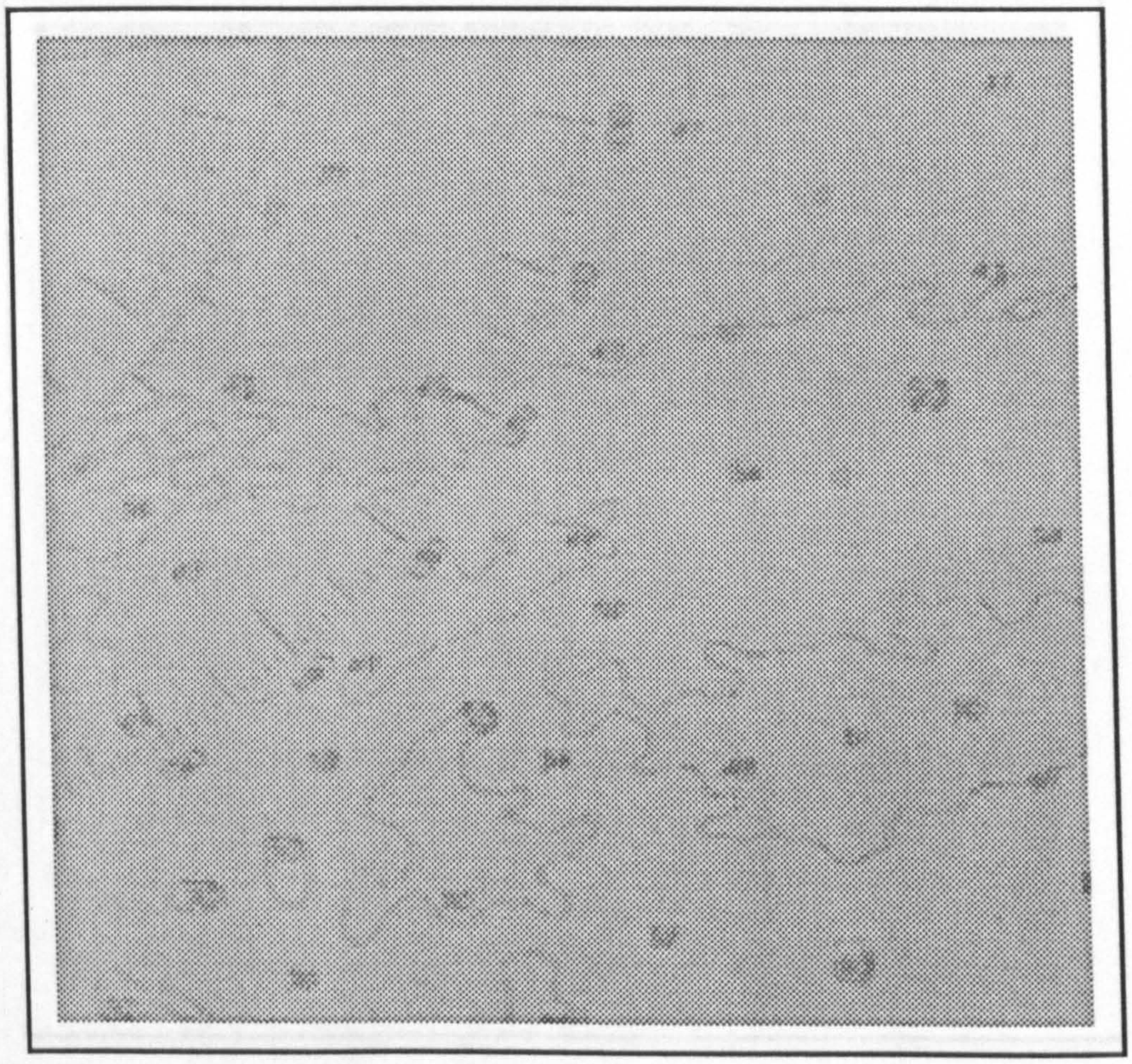


Figure 3.4: The result of applying a 3×3 median filter on the image in figure 3.3

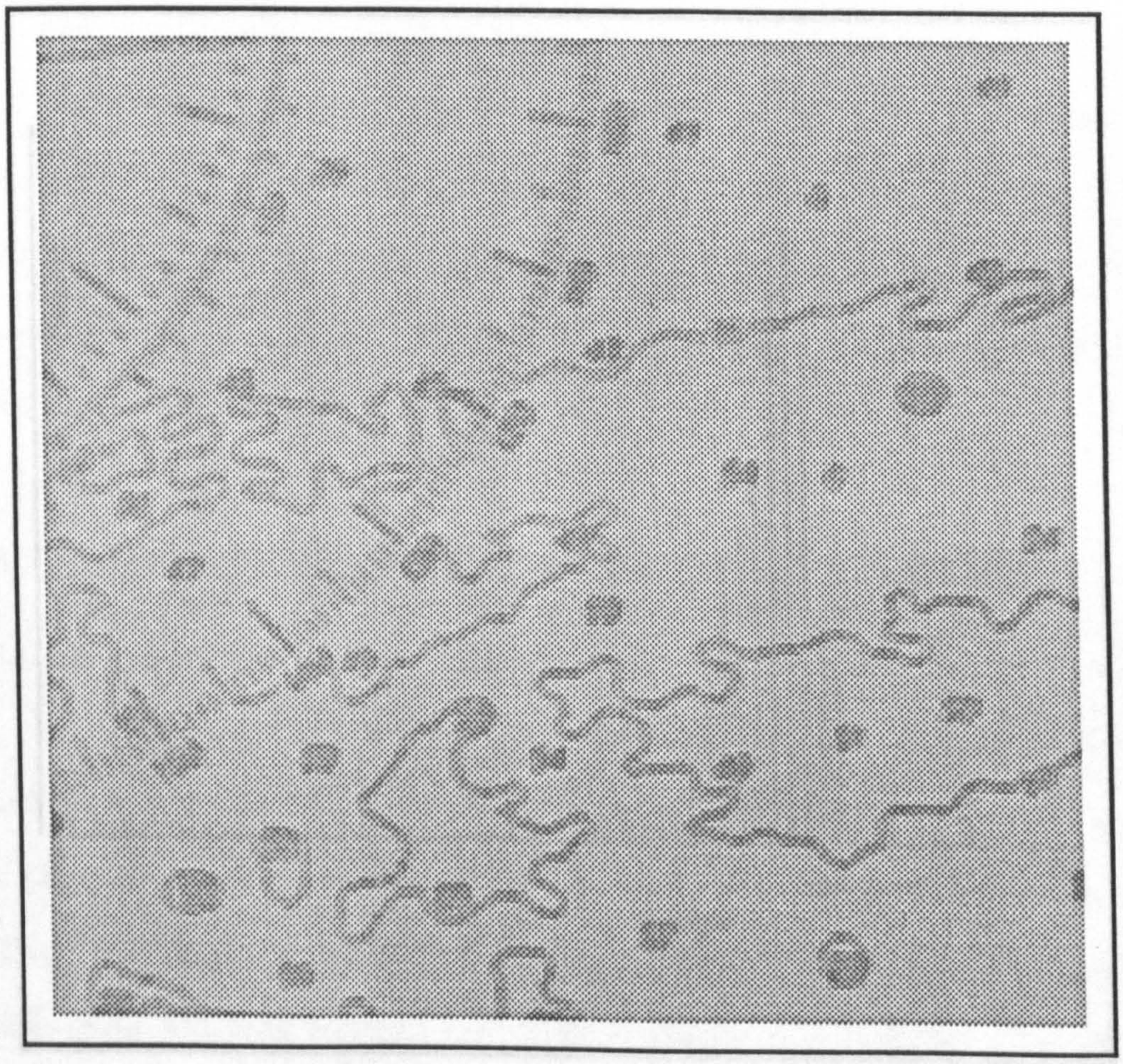


Figure 3.5: The second lowest element has been selected in the median filter process from 3×3 regions of the image in figure 3.3

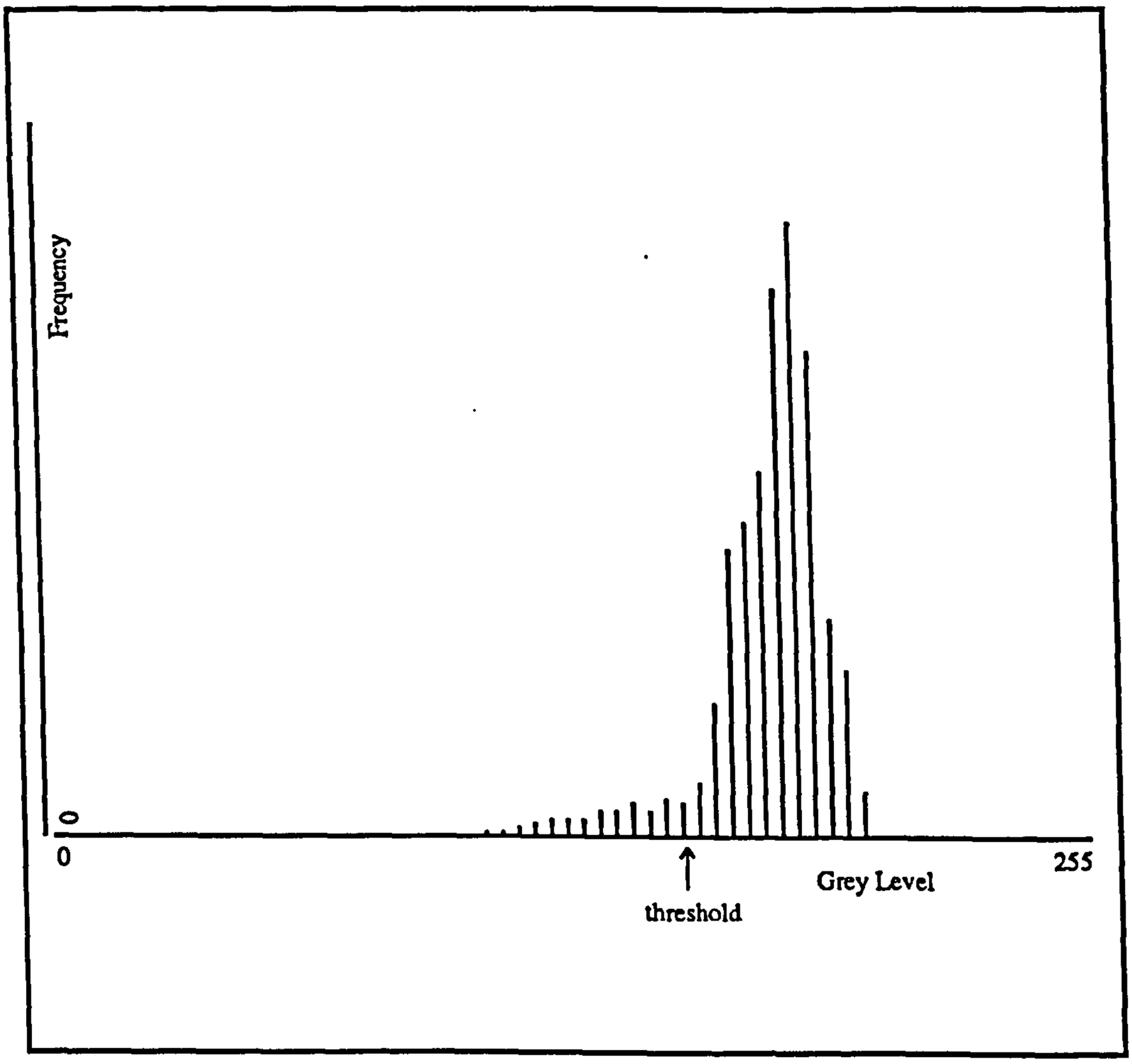


Figure 3.6: The grey level histogram of the image in figure 3.3.

the image. The histogram of an image of a chart is shown in figure 3.6. The image capture system theoretically has a 6 bit grey level range. The proportion of this grey level range which is populated implies that the system as a whole has an effective range of 5 bits; a typical figure for video image acquisition systems. Figure 3.7 shows a binary image produced by thresholding at the grey level shown in figure 3.6. The overlap in background and line intensities results in an unclean separation of those two classes. The low proportion of pixels which belong to lines results in a small mode in the histogram for which automatic detection would be unreliable. In addition, processes which begin by binarisation cannot utilise the valuable information contained within the grey levels of the original image and often result in aesthetically poor representations of the image features.

3.3.3 A Compressed Image Representation for Fast Display

The need for a data base of sea charts which could generate a rapid visual display of a portion of a sea chart and be used for navigational processes suggested that two forms of chart representation might be necessary, one for each task. The quadtree structure of image representation (Samet [1984]) might offer a suitable format of storage for fast display purposes. A quadtree generation program was written which generated a compressed form of representation of the quadtree structure known as the linear quadtree (Tamminen [1984]). The hierarchical nature of the quadtree structure might then offer an easy method of performing zoom operations on the display of a chart. Performing this process on a binary image of similar information content to the image in figure 3.7 resulted in a memory saving of 90%. The image coding aspects of the work are now being explored in a separate project and were not pursued here.

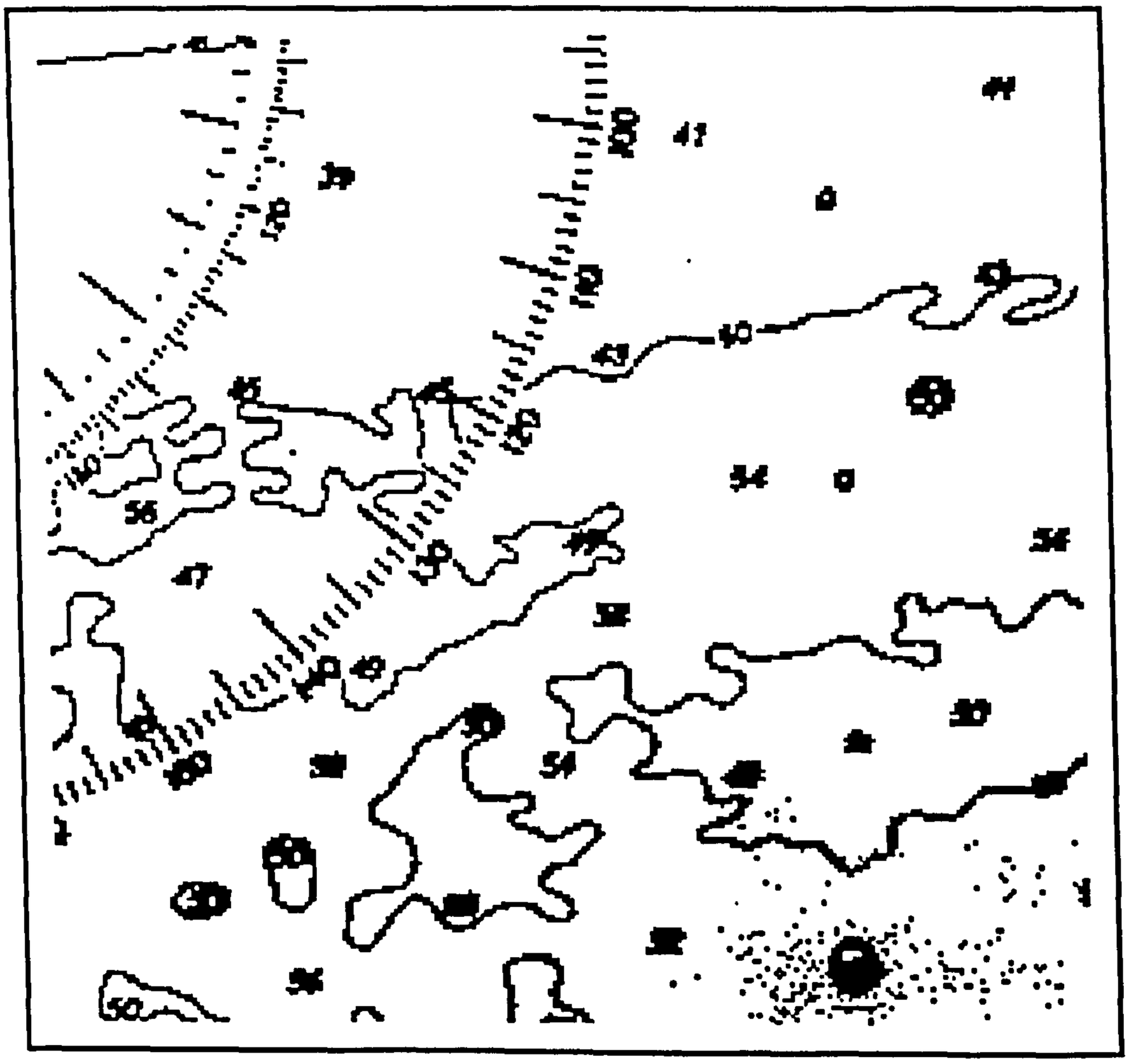


Figure 3.7: Result of Global Thresholding of the image in figure 3.3

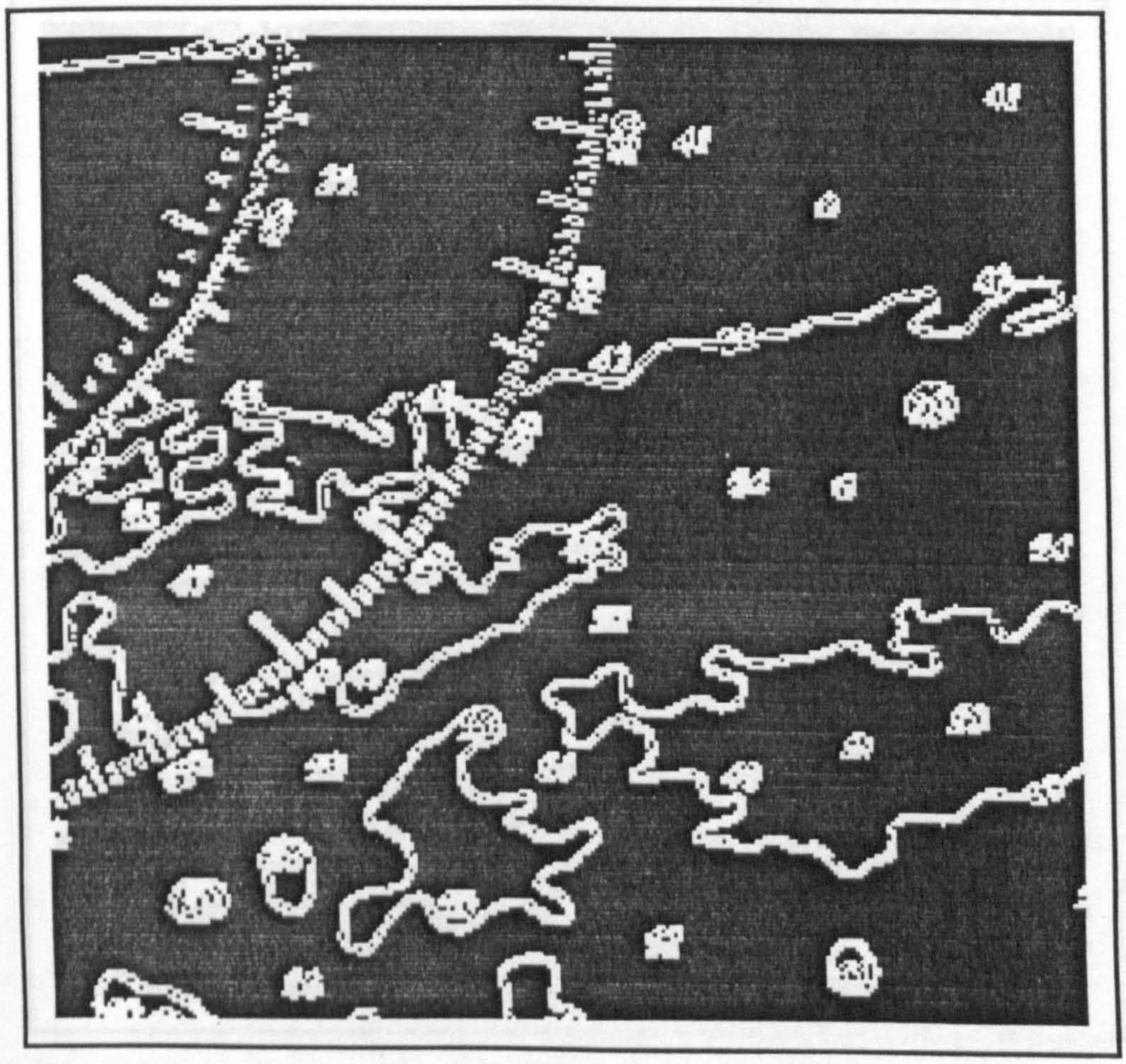
3.3.4 Line Finding from Edges

First attempts at extracting features from charts involved applying sequences of standard noise removal and edge detection techniques.

Line finding is often considered as a problem in finding the edges of lines and extracting the line work from the resultant edge map. A general convolution routine was written and the results of applying a number of edge operators were observed. Figure 3.3 shows the original image from which many of the following images were derived.

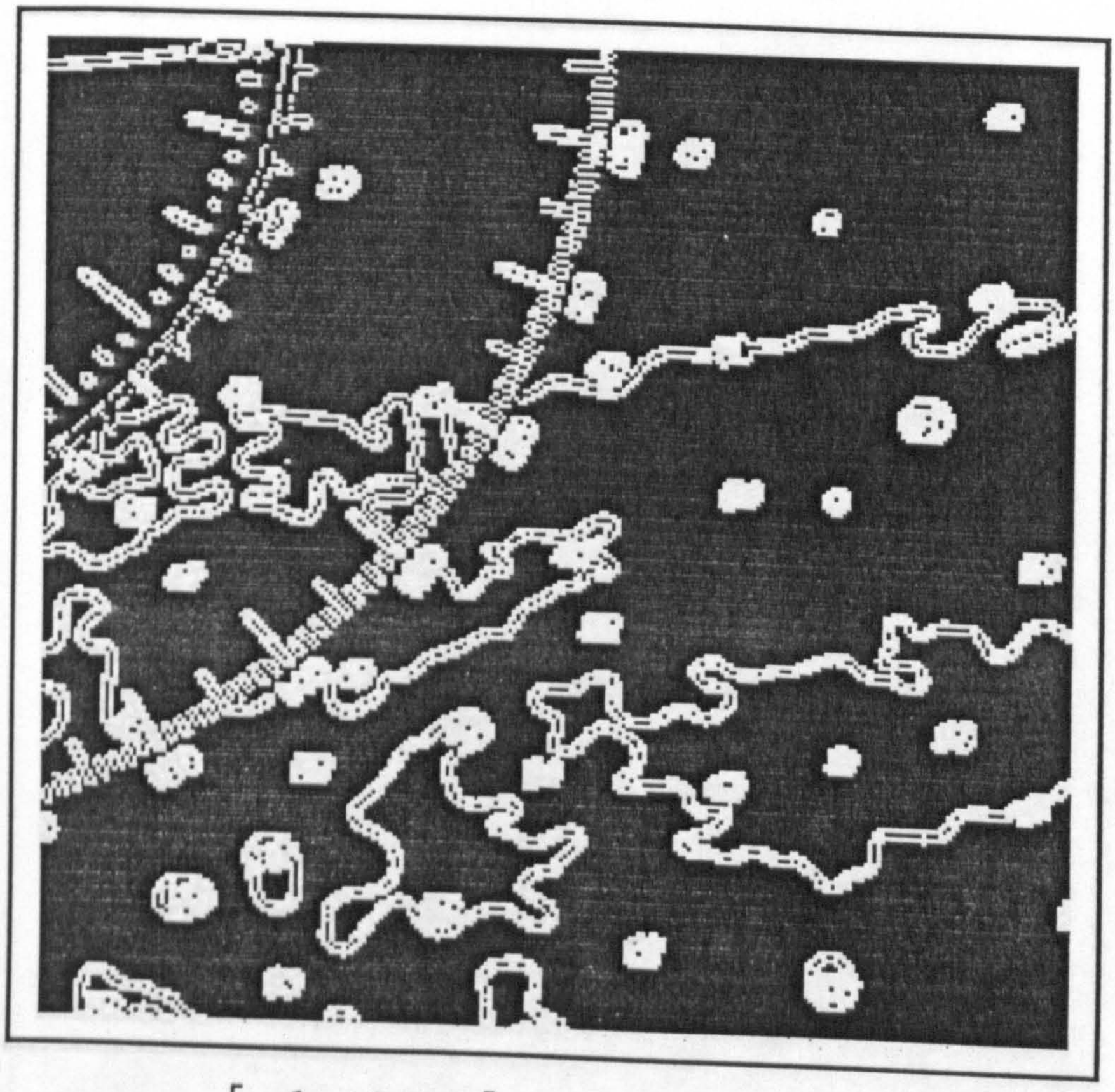
Figures 3.8 to 3.10 show the results of applying various edge operators to the image shown in figure 3.3. These images show the result of applying standard edge operators on images of sea charts. The line information in the images is of the scale of the operators themselves. The edges therefore occur at scales smaller than the operators. This results in spatially coincident responses to both edges of a line. The smallest operator, that of figure 3.8 exhibits this merging of edge information. Establishing the true path of the line with the information provided by these operations would require tracking or thinning operations which could make use of the small amount of local edge direction information to attempt to interpolate the true position of the line. A search was therefore made for an algorithm which could recognise the structure of a line rather than its edges. A suitable process is described in section 3.4. Figure 3.11 shows the result of applying the EHNUM operator. This operator substitutes the central pixel in a 3×3 neighbourhood for the highest or lowest pixel value in the neighbourhood, depending on whether the central pixel's value is closer to the high or low value. This has the effect of sharpening edges in the image, along with emphasising the noise level.

The first stages of the set of processes proposed by Nevatia and Babu [1980]



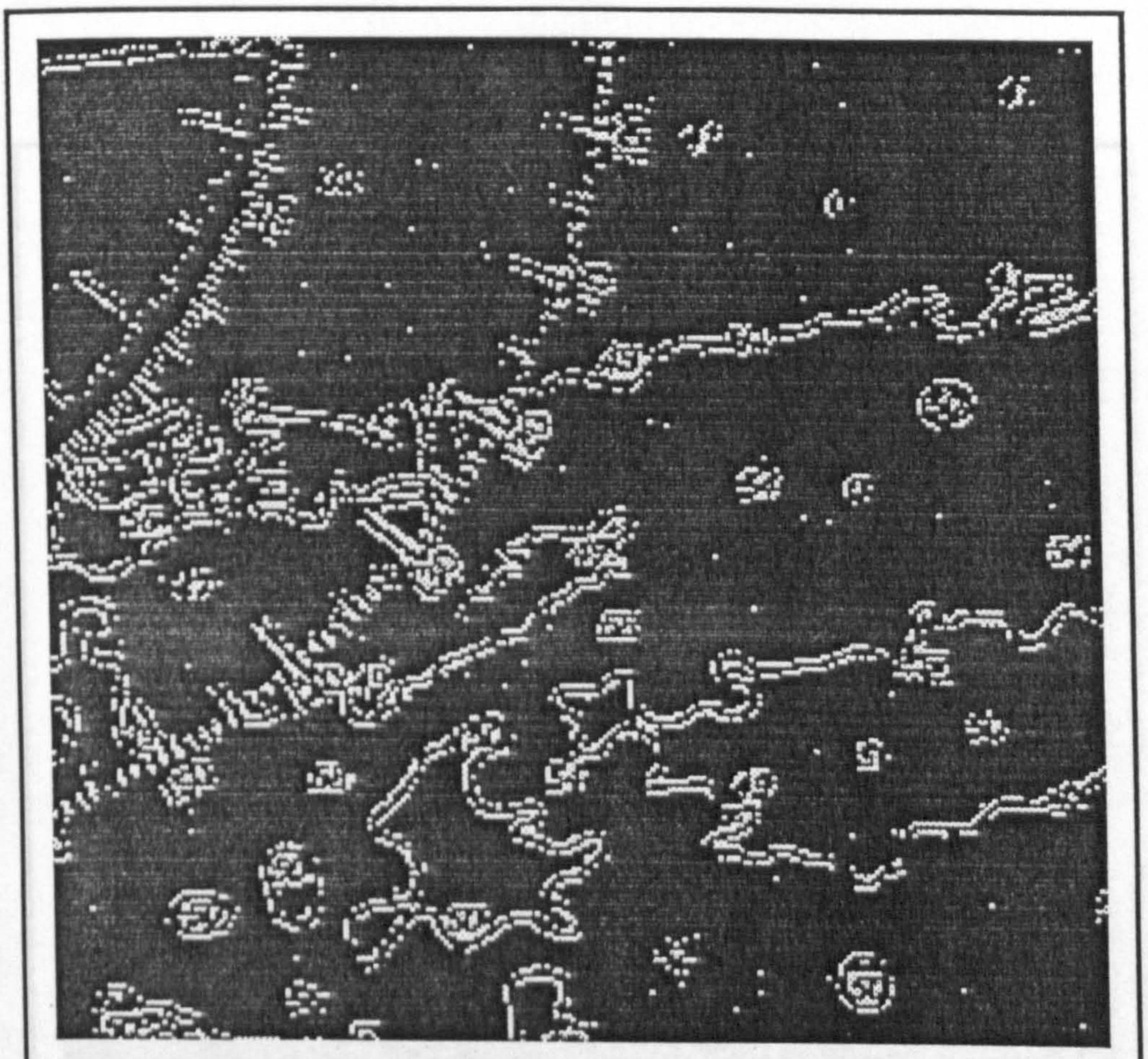
$$\begin{bmatrix} -1 & 1 \\ 0 & 0 \end{bmatrix} and \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix}$$

Figure 3.8: Edge Detection Using Small Horizontal and Vertical Edge Masks



$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} and \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Figure 3.9: Edge Detection using the Sobel operators



$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Figure 3.10: Edge Detection using the Laplacian Operator

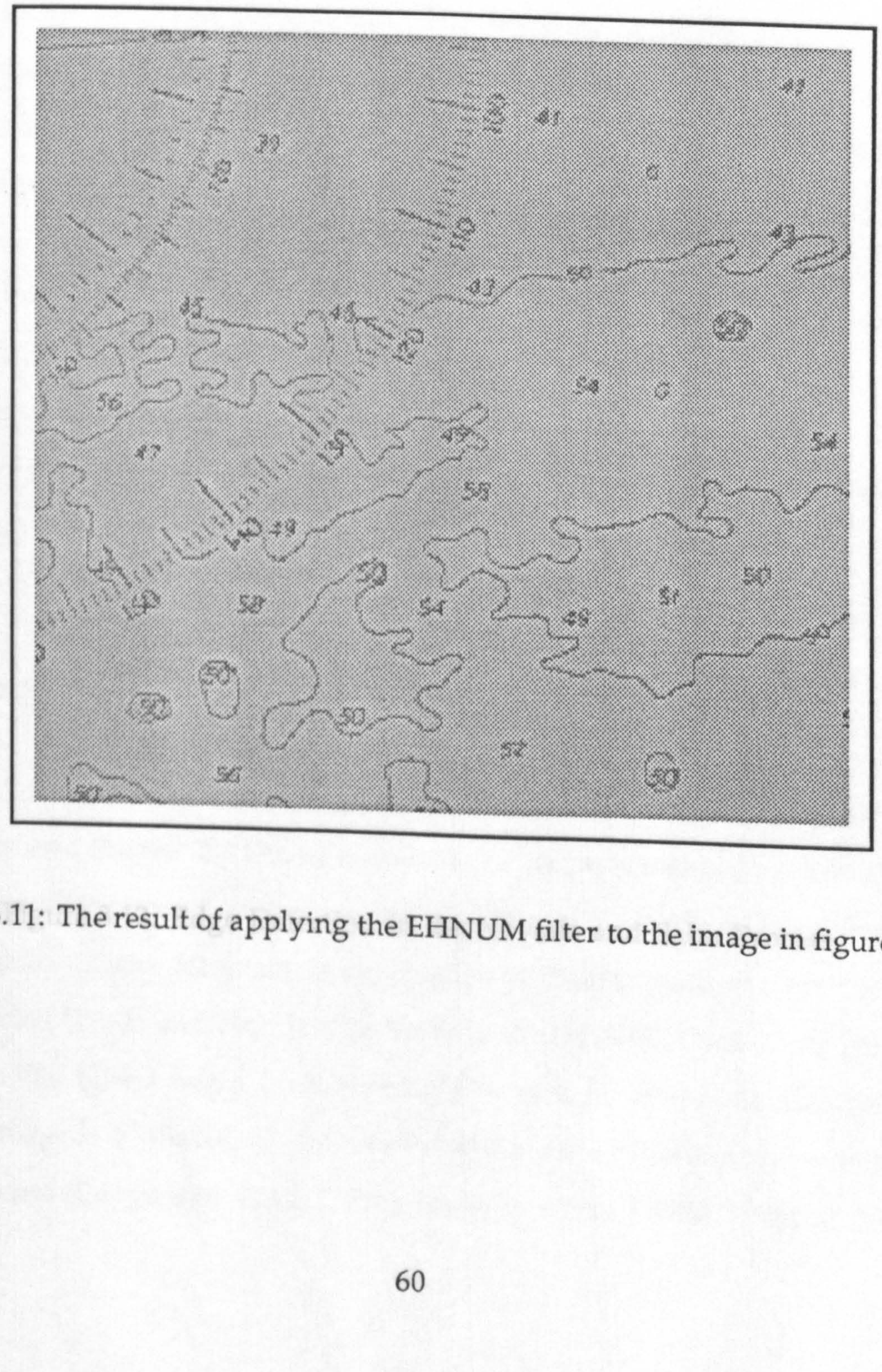


Figure 3.11: The result of applying the EHNUM filter to the image in figure 3.3

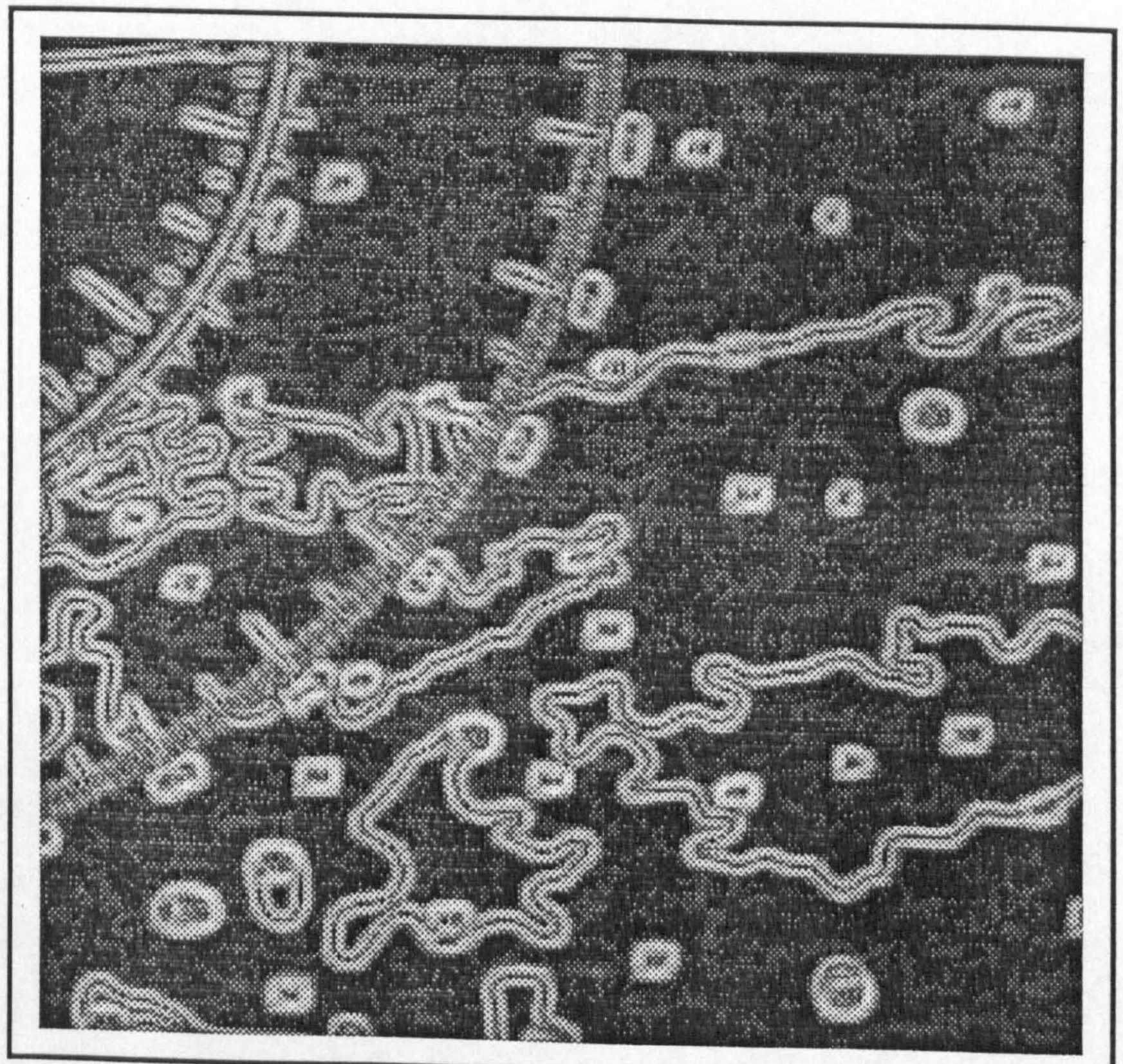


Figure 3.12: Edge Detection in the Nevatia and Babu Process

were implemented and tested on a chart image. This involved performing convolution on the image with a set of six 5×5 operator masks shown in figure 3.13. The masks model step edges in gaussian noise in 6 directions. The output image of the convolution step is comprised of the maximum output from the set of masks at each image point accompanied by a marker indicating the mask which gave this maximum output. The image in figure 3.12 shows the result of this convolution. The sequence of operations following the convolution step involves thinning of the edge point map and linking of the remaining edge points. Following this, the edge boundaries are linearly approximated and close parallel edge segments, termed *apars*, of opposite edge directions are found, which indicate the presence of a line. Having performed the convolution it was possible to foresee that a linear approximation stage, that could give sufficiently long edge segments to allow recognition of parallel segments, would lose the smoothly meandering properties of the depth contours. This was seen as undesirable and an alternative line feature extraction technique was sought.

3.4 Implementation of Robust Line Tracker

Various schemes for extracting lines from images have been described in the literature (see chapter 2). Out of the dozen or so papers found on line tracking many were rejected without practical trial on the basis of their obvious unsuitability. Many for example assumed a particular geometry for lines to be extracted (Duda and Hart [1972], Yachida et al. [1979], Black et al. [1981]). Many of the others begin by binarising a gray tone image, or assuming a binary image is available, to provide the input for a thinning or skeletonisation process (Capponetti et al. [1982], Ramachandran [1980], Woetzel [1979],

-100	-100	0	100	100
-100	-100	0	100	100
-100	-100	0	100	100
-100	-100	0	100	100
-100	-100	0	100	100

-100	32	100	100	100
-100	-78	92	100	100
-100	-100	0	100	100
-100	-100	-92	78	100
-100	-100	-100	-32	100

100	100	100	100	100
-32	78	100	100	100
-100	-92	0	92	100
-100	-100	-100	-78	32
-100	-100	-100	-100	-100

100	100	100	100	100
100	100	100	100	100
0	0	0	0	0
-100	-100	-100	-100	-100
-100	-100	-100	-100	-100

100	100	100	100	100
100	100	100	78	-32
100	92	0	-92	-100
32	-78	-100	-100	-100
-100	-100	-100	-100	-100

100	100	100	-32	-100
100	100	92	-78	-100
100	100	0	-100	-100
100	78	-92	-100	-100
100	32	-100	-100	-100

Figure 3.13: Convolution masks for the Nevatia and Babu process

Wojcik [1984]). Some of the work found in the literature depends upon using specialised hardware (Fulford [1981], Black et al. [1981]). The work of Nevatia and Babu [1980] was investigated but rejected on the basis that it gives only a polygonal approximation of the line features.

One particular approach seemed an ideal starting point for the problem of extracting the randomly meandering smooth lines found in sea charts (Watson et al. [1984]). The process is able to recognise the topographic intensity profile of lines in images which have been smoothed by a gaussian convolution operator (see figure 3.14). The convex topographic profile of the smoothed lines has the property that a point on such a profile will exceed the average of its eight connected neighbours by a non-negligible amount (see figure 3.15). The sequence involves :-

- Gaussian smoothing
- Double adaptive thresholding
- Region tidying
- Ridge walker line tracking

Gaussian Smoothing Gaussian smoothing is the process of convolving the image with a mask which in discrete form approximates equation 3.1.

$$h(x,y) = A_0 e^{-\frac{x^2+y^2}{\sigma^2}} \tag{3.1}$$

The gaussian smoothing process has the dual purpose of performing noise reduction on the raw image and converting the topographic structure of line features to a convex intensity profile (see figure 3.14). This intensity profile will only be achieved with lines up to a certain width for a given σ value for the gaussian filter. Lines whose width are appreciably greater than the width

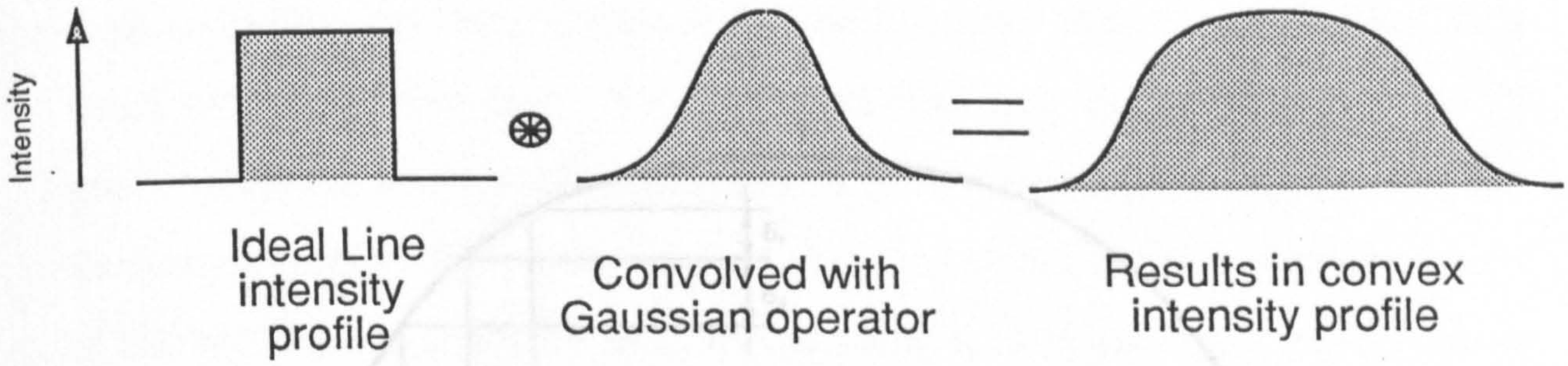


Figure 3.14: A Lines Intensity Profile after Gaussian Smoothing

of the gaussian will retain a plateau in the cross section of their intensity profiles and will thus not be suited to the ensuing feature extraction process.

When developing the Gaussian smoothing process it seemed opportune to implement a more general algorithm providing, in addition, the popular difference of gaussians edge enhancement.

Double Adaptive Thresholding Prior to the process of Double Adaptive thresholding it is suggested that the smoothed image is resampled at each alternate pixel. This has the effect of steepening and narrowing the ridge profile of lines. Its relevance clearly depends on issues of scale in the original image. Double adaptive thresholding was tried on both the resampled and the original smoothed image and seemed to perform better on the original image than on the resampled image. Use of a resampled image caused a noticeable fragmentation of certain line features when the line width was relatively thin. This is probably because the resolution of the images used here (in terms of the number of pixels in a lines cross section) was less than that used by Watson. The resolution of images used for this work was limited by the equipment.

The line feature extraction properties of double adaptive thresholding are

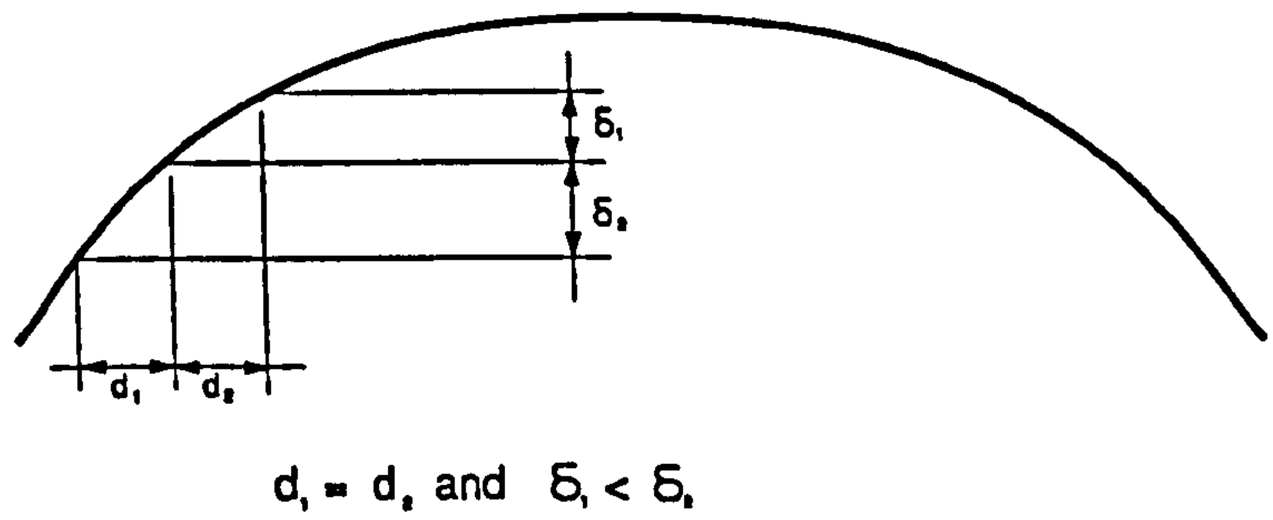


Figure 3.15: The Topographic Properties of a Ridge Structure

```
Factor := user defined factor slightly greater than unity;
Threshold := user defined region threshold;

FOR x := 2 TO x_image_size-1 DO
   BEGIN
      FOR y := 2 TO y_image_size-1 DO
         BEGIN
            Average:=average grey level of eight connected neighbours of pixel(x,y);
            IF grey level of pixel at (x,y) > average × factor
               THEN mark pixel on a convex topographic profile
               ELSE IF grey level of pixel at (x,y) > Threshold
                  THEN Mark pixel as region pixel
                  ELSE set pixel to zero;
         END;
END;
```

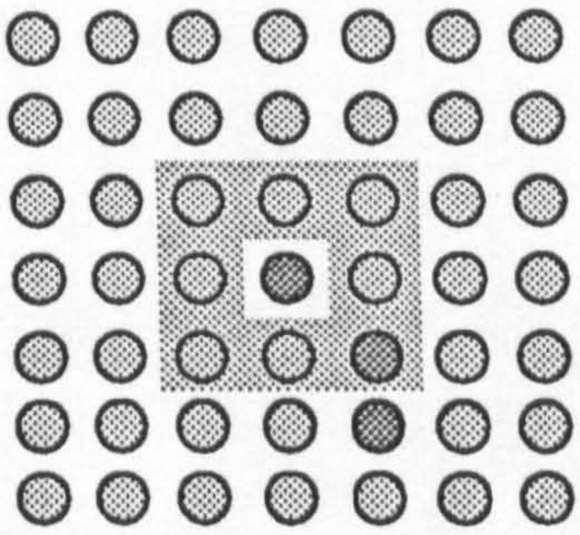
Figure 3.16: Double Adaptive Thresholding.

based upon a knowledge of the mathematical characteristics of ridge profiles. A ridge profile can be recognised by the property that a central pixel in a small neighbourhood (e.g. 3×3) has an intensity value which is slightly greater than the average of its neighbours within that region (see figure 3.15). Excluding pixels which lie below a certain threshold set just above the level of the background intensity from contributing to this average serves to make this operation more robust. Pixels which exceed the average multiplied by some factor slightly greater than unity are considered to be on a ridge profile and thus retain their original grey level. Those which fail this test but exceed a threshold set to extract regions are set negative. All other pixels are set to zero. This process is illustrated in pseudocode in figure 3.16

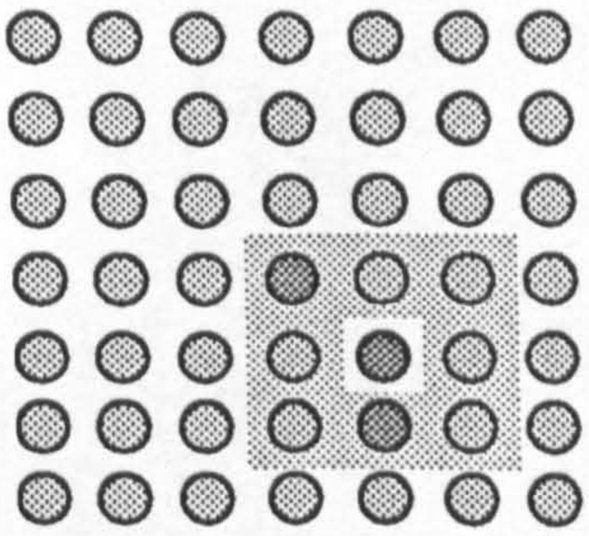
Line Tracking The line tracking algorithm takes as input the image produced by double adaptive thresholding and scans in raster mode until a pixel exceeds a threshold computed by equation 3.2.

$$\text{Threshold} = \max \begin{cases} \text{Background threshold} \\ 0.7\times \text{ non zero eight connected neighbours} \\ 0.9\times \text{ unmarked eight connected neighbours} \end{cases} \tag{3.2}$$

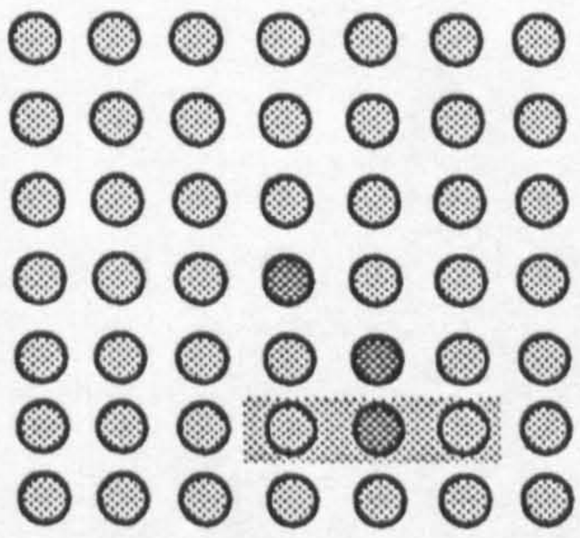
When such a pixel is found then a number of tests are carried out to see if it represents the beginning of a line (see figure 3.17). This testing is described in pseudo-code in figure 3.19. For reference this pixel is termed *current_pixel*. Each of the unmarked eight connected neighbours of the pixel are tested in decreasing order of grey level magnitude. If one of these neighbours also exceeds the threshold computed by equation 3.2 then, using the label *temporary_pixel* to reference this second pixel the direction from *current pixel* to *temporary_pixel* (see figure 3.17) is established. The three pixels which lie in



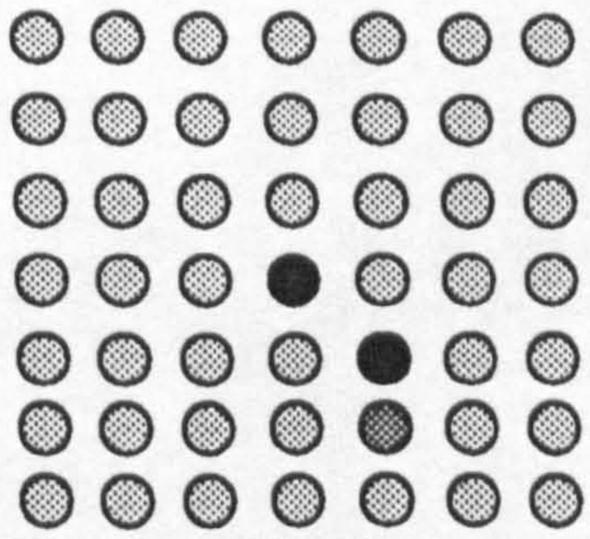
a) the cut off condition is tested using the pixels in the shaded area and if satisfied then the maximum valued pixel is tested



b) the highest value pixel is tested against the cut off condition



c) if this pixel satisfies the cut off condition then the three neighbours shown are tested to see if they are already marked



d) if all these conditions are met then the two pixels shown are marked and tracking of the line begins

Figure 3.17: Testing for the Beginning of a Line

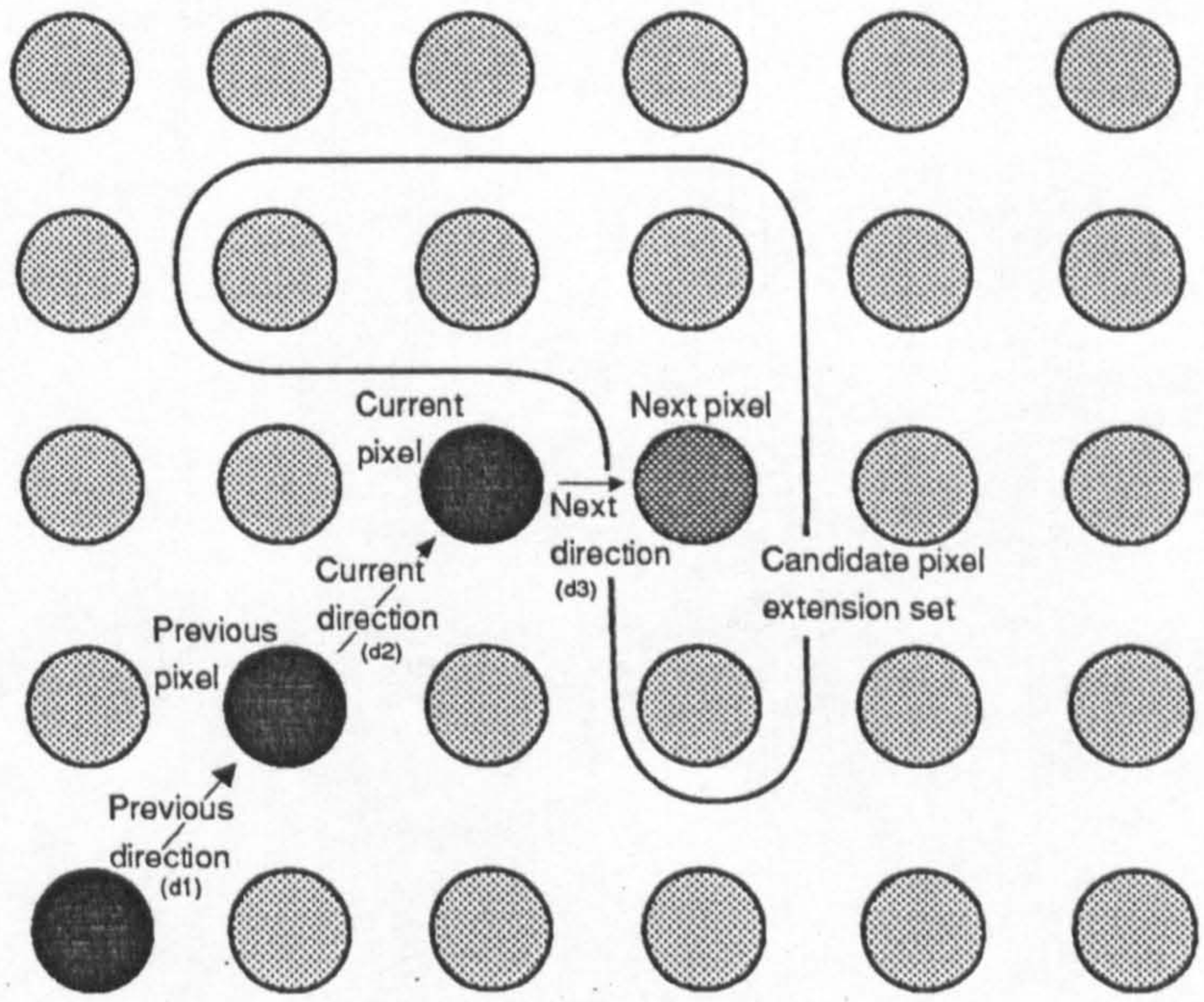


Figure 3.18: Testing for Line Continuation

this direction ± 1 are tested in descending order of magnitude to see if all of their neighbours in the direction from *temporary_pixel* to the pixel in question ± 1 are not marked. If this is so then *current_pixel* and *temporary_pixel* are considered to be the start point of a line and are marked as such. The line is then tracked as described below until the end of the line is reached. At this point the remainder of the unmarked untested neighbours of *current_pixel* are tested to see if the line continues in another direction. When all candidate neighbours of *current pixel* have been tested the raster scan search for further line beginnings is recommenced.

Tracking a line is a process of examining a local neighbourhood surrounding the current last pixel in the line. The pixels involved in testing for a continuation are shown in figure 3.18. The tracking process is described in pseudo-code in figure 3.20. The candidate pixel is selected as the pixel with maximum grey level from the set shown in figure 3.18, i.e., the five pixels which lie in the direction range $d2 \pm 2$ from *current pixel*. The pixel is accepted

if it satisfies the following conditions :-

1. Its grey level must be higher than a threshold slightly greater than the average background grey level.

2. It must have no marked neighbours in the directions $d3-1 \pmod 8$ to $d3+1 \pmod 8$ from *current pixel.*

3. The difference in the directions d1 and d3 must be less than or equal to 2 (mod 8).

If the pixel is accepted then it becomes the new *current pixel* and the search for more extensions to the line continues. Otherwise the system returns to searching for the beginning of another line.

To divorce the code representing the algorithm from that of data handling a versatile buffer routine was written in Fortran which allowed any pixel to be accessed by a function call which returns a 2 byte variable representing the grey level of the pixel. This hid from the main algorithm the fact that the whole image matrix could not be held in memory at one time. All routines written since the line tracker just described reference the image data through this buffer.

3.4.1 Data Structure for Line Representation

The data structure chosen for line representation is that of the Freeman chain code (Freeman [1974]). This coding scheme offers explicit pixel by pixel encoding of lines in a relatively compact fashion and is therefore suited to the storage of the output from the tracker. An individual line is stored by specifying the cartesian coordinates of the start point of the line followed by a

```
FOR x := 3 TO X_image_size-2 DO
   BEGIN
      FOR y := 3 TO y_image_size-2 DO
   BEGIN
      curpix := coordinates of current pixel;
      curval := grey level of current pixel;
      IF curval > cutoff(curpix) AND curpix is unmarked
         THEN BEGIN
            WHILE NOT all unmarked 8 connected neighbours
                                          of curpix tested DO
               BEGIN
                  tempix := coordinates of maximum unmarked
                                          untested neighbour of curpix;
                  temval := grey level of tempix;
                  dir := direction from curpix to tempix;
                  IF temval > cutoff(tempix)
                     THEN
                        REPEAT
                     pix := coords of maximum neighbour of
                              tempix in direction dir ± 1;
                     IF neighbours of tempix in direction from
                              tempix to pix ± 1 not marked
                        THEN BEGIN
                           mark tempix; mark pix;
                           track line;
                        END;
                  UNTIL line tracked OR neighbours of tempix
                              in direction dir ± 1 tested;
               END;
            END;
         END;
   END;
```

Figure 3.19: Pseudo-code for Searching for the Beginning of Lines

```
Cur_Dir := Current direction;
Prev_Dir := Previous direction;
Threshold := Grey level just above background level;
REPEAT
   next_pix := unmarked neighbour in direction d from cur_pix with
                  maximum grey level (where abs (d – Cur_Dir(mod 8)) ≤2
   IF next_pix exists
      THEN
         BEGIN
            next_dir := direction from Cur_pix to next_pix;
            next_val := Grey level of next_pix;
         END
      ELSE
         next_val := 0;
   IF next_val > threshold AND
         abs (next_dir – prev_dir(mod 8)) ≤2 AND
         neighbours of next_pix in directions
         next_dir-1, next_dir + 1 from cur_pix
         are not marked
      THEN
         BEGIN
            mark next_pix;
            cur_pix := next_pix;
            prev_dir := cur_dir;
            cur_dir := next_dir;
         END
      ELSE
         flag tracking complete;
UNTIL tracking is complete;
```

Figure 3.20: Pseudo-code for Tracking a line

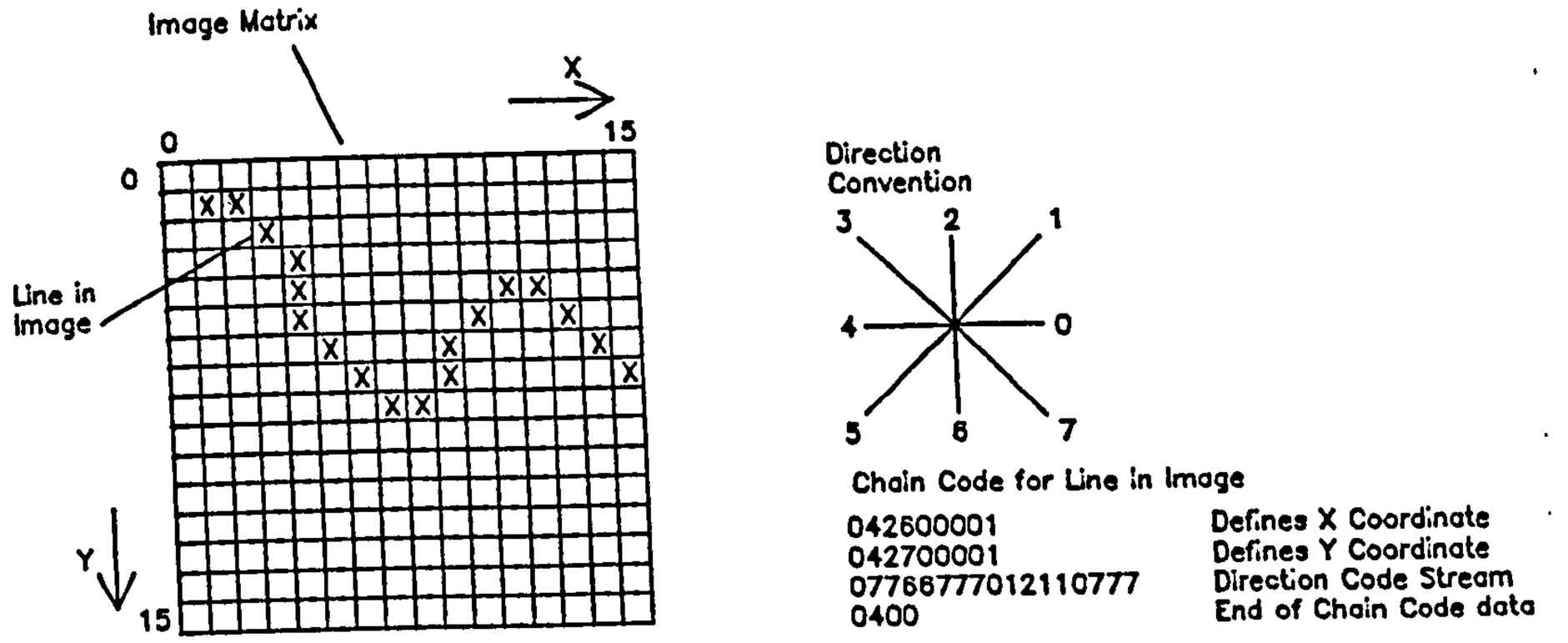


Figure 3.21: The Freeman Chain Code

sequence of direction codes in the interval 0 to 7. These direction codes define the track of a line from one eight connected neighbour to another (See figure 3.21).

The lines are stored as one long stream in a data base and are delimited by an introducer and one of a set of function codes. The introducer of a function code is the numerical sequence 04. This is chosen as it is unlikely that a line will require this code as it implies a backstepping in the path of the line. Following the 04 sequence a 2 digit function code specifies the nature of the information which the function conveys. Table 3.1 shows some of the function codes which can be included in a chain code sequence. This form of coding is well suited for access by routines which require sequential passes through the data, but if random access is required or reverse sequential passes then the Freeman chain code is not well suited.

0400	End of chain Code
0404	Valid 04 sequence
0405XYZ	Marker number XYZ. XYZ is a three digit octal number used for marking points in a chain
0423WXYZ	Contour elevation indicator. Chain that follows is a contour of elevation WXYZ
0426VWXYZ	Resets the absolute x coordinate to VWXYZ
0427VWXYZ	Resets the absolute y coordinate to VWXYZ

Table 3.1: Some Function codes from the Freeman Chain Code Line Representation Scheme

3.4.2 Experience with the Line Tracker

Examples of images generated in the line tracking process are shown in figures 3.22 to 3.24. The first of these has first been negated and then blurred by convolving it with a gaussian filter of $\sigma = 1.2$. This gives a typical cross sectional width of a line as being approximately 4 pixels.

The image in figure 3.23 shows the result of performing double adaptive thresholding on the image of figure 3.22 with a *factor* of 1.063, a *low threshold* of 40 and a *region threshold* of 255. The process has performed well in most areas. All but the smallest lines of the compass rose have been detected. A small amount of spurious line pixels resulting from ridge type profiles occurring in noisy areas of the background where the background intensity is relatively high. Increasing the low threshold for the operation causes true line pixels to be missed. The few spurious line pixels in the image of figure 3.23 do not prove to be a problem as they do not generally form connected sets that satisfy the criteria for beginning a line during line tracking.

Line tracking is performed on images such as that in figure 3.23 and an

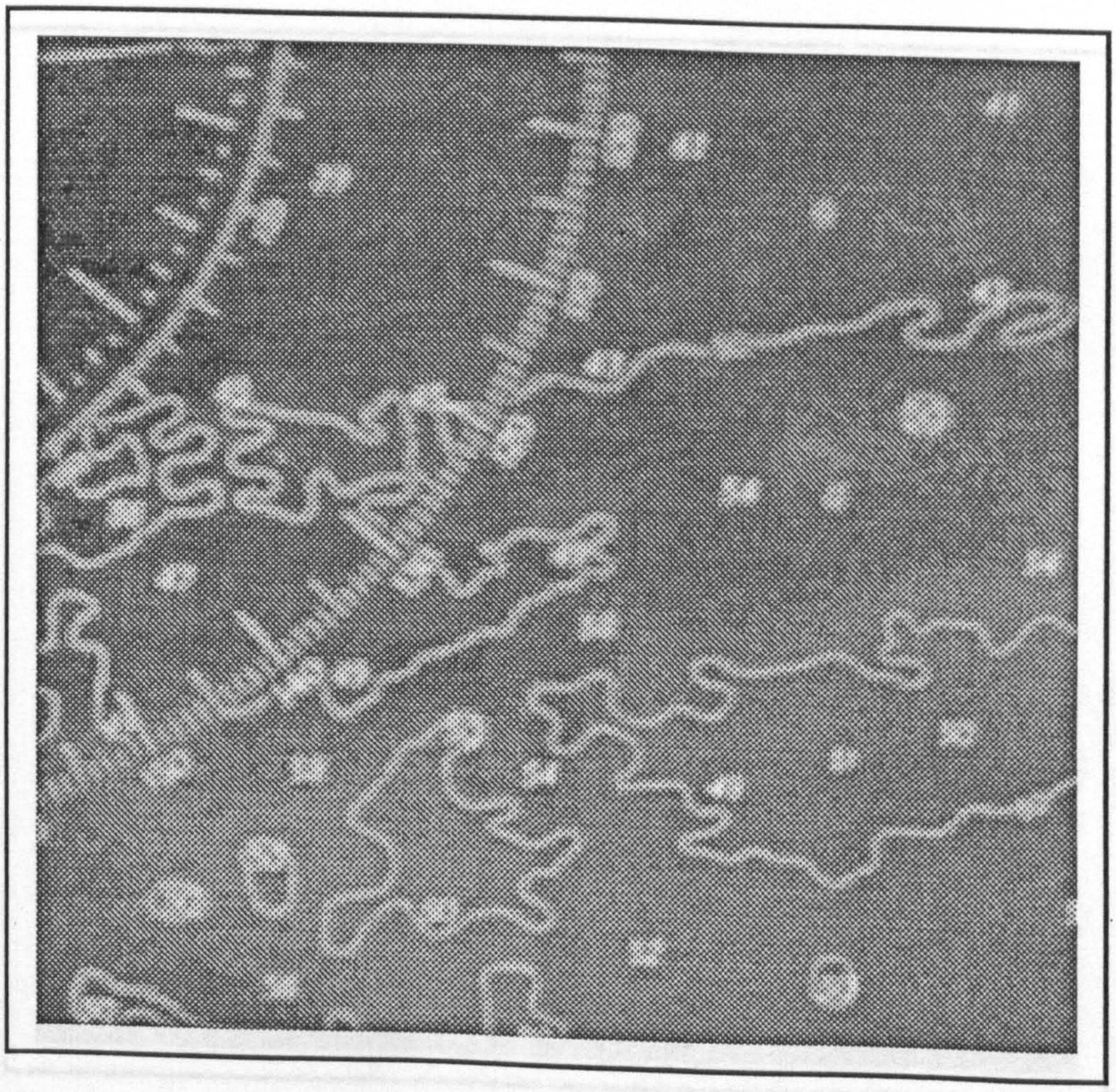


Figure 3.22: The image in figure 3.3 after grey level inversion and smoothing with a Gaussian Filter

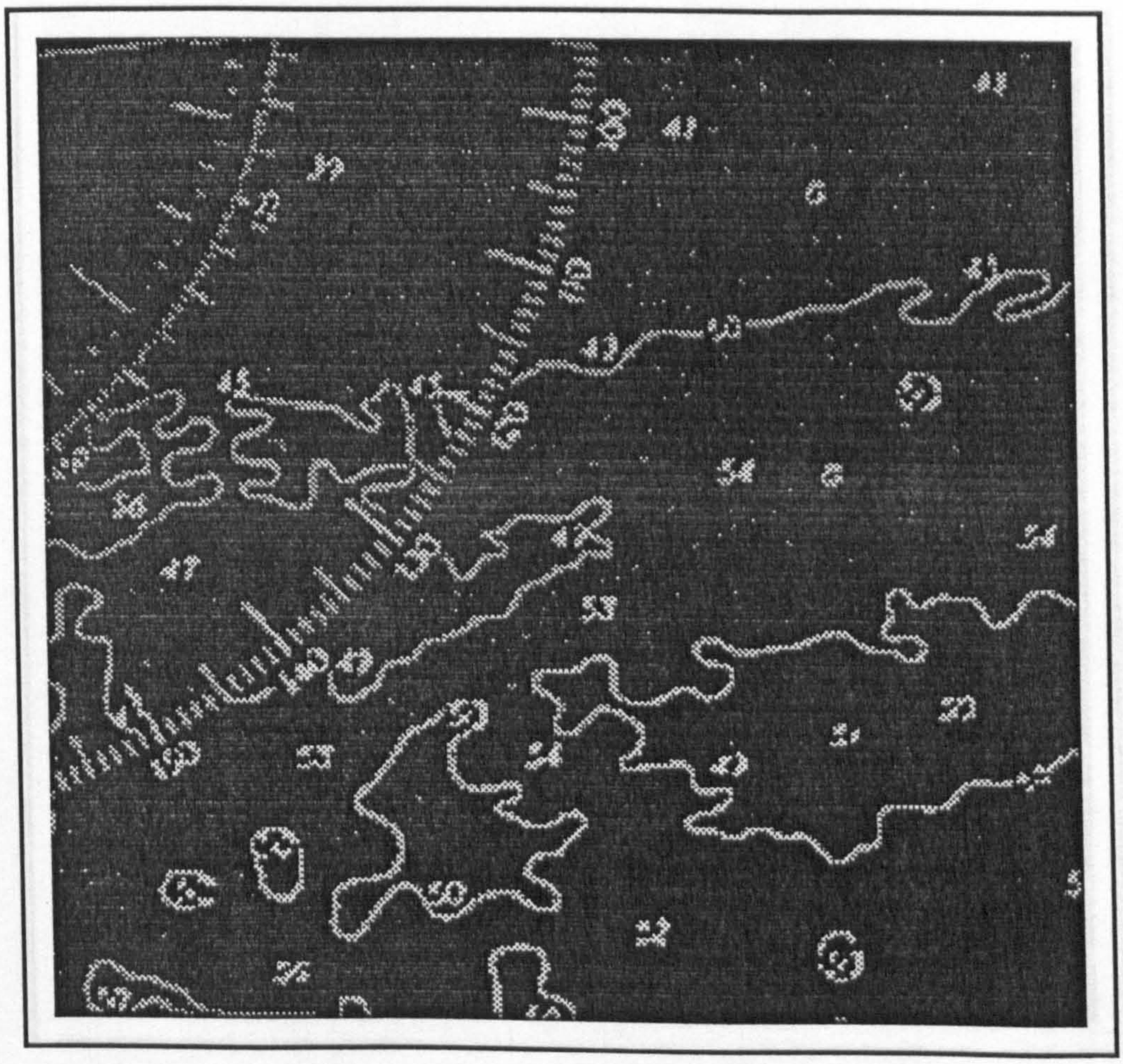


Figure 3.23: The image in figure 3.22 after Double Adaptive Thresholding

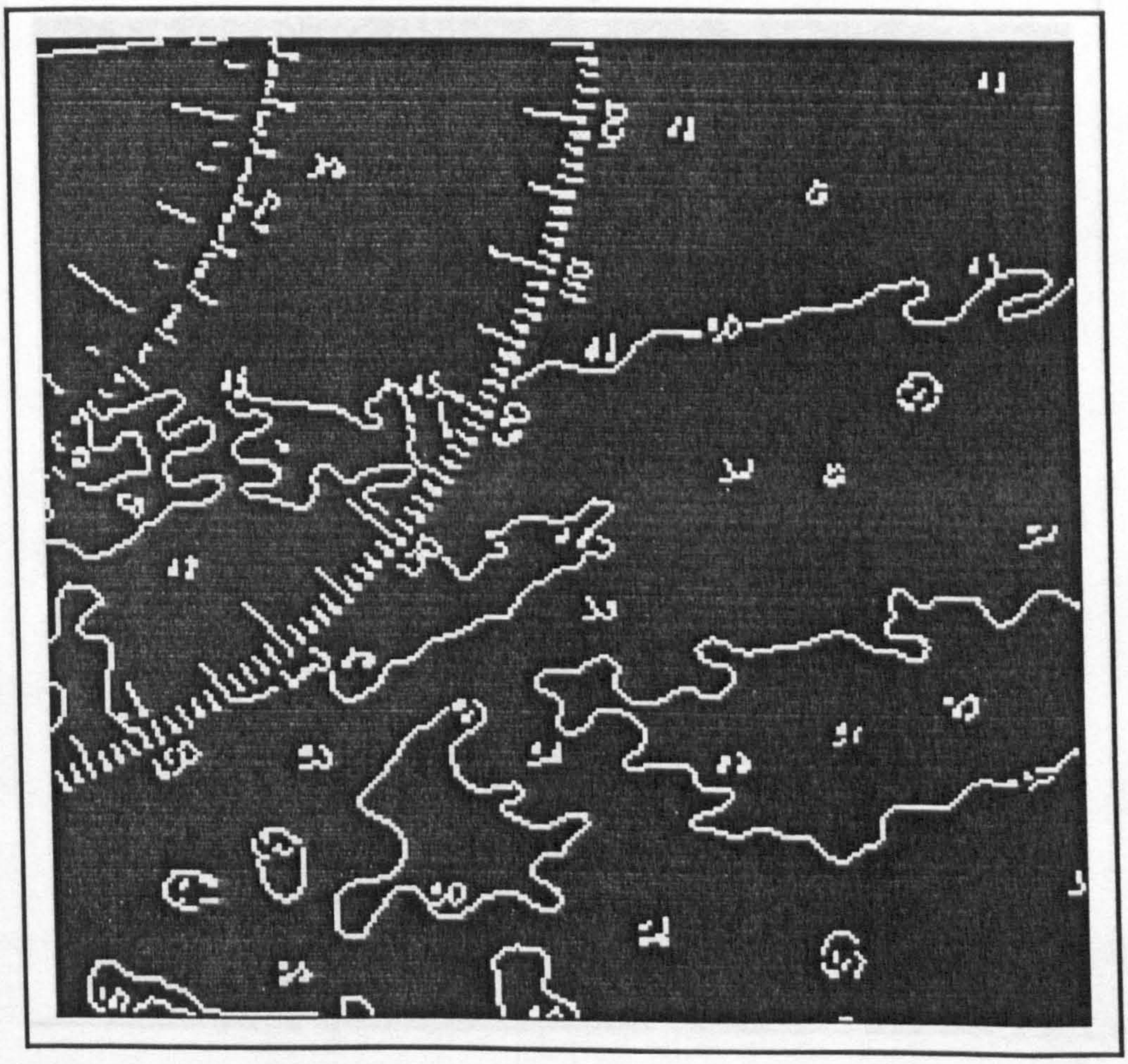


Figure 3.24: The image in figure 3.23 after Line Tracking

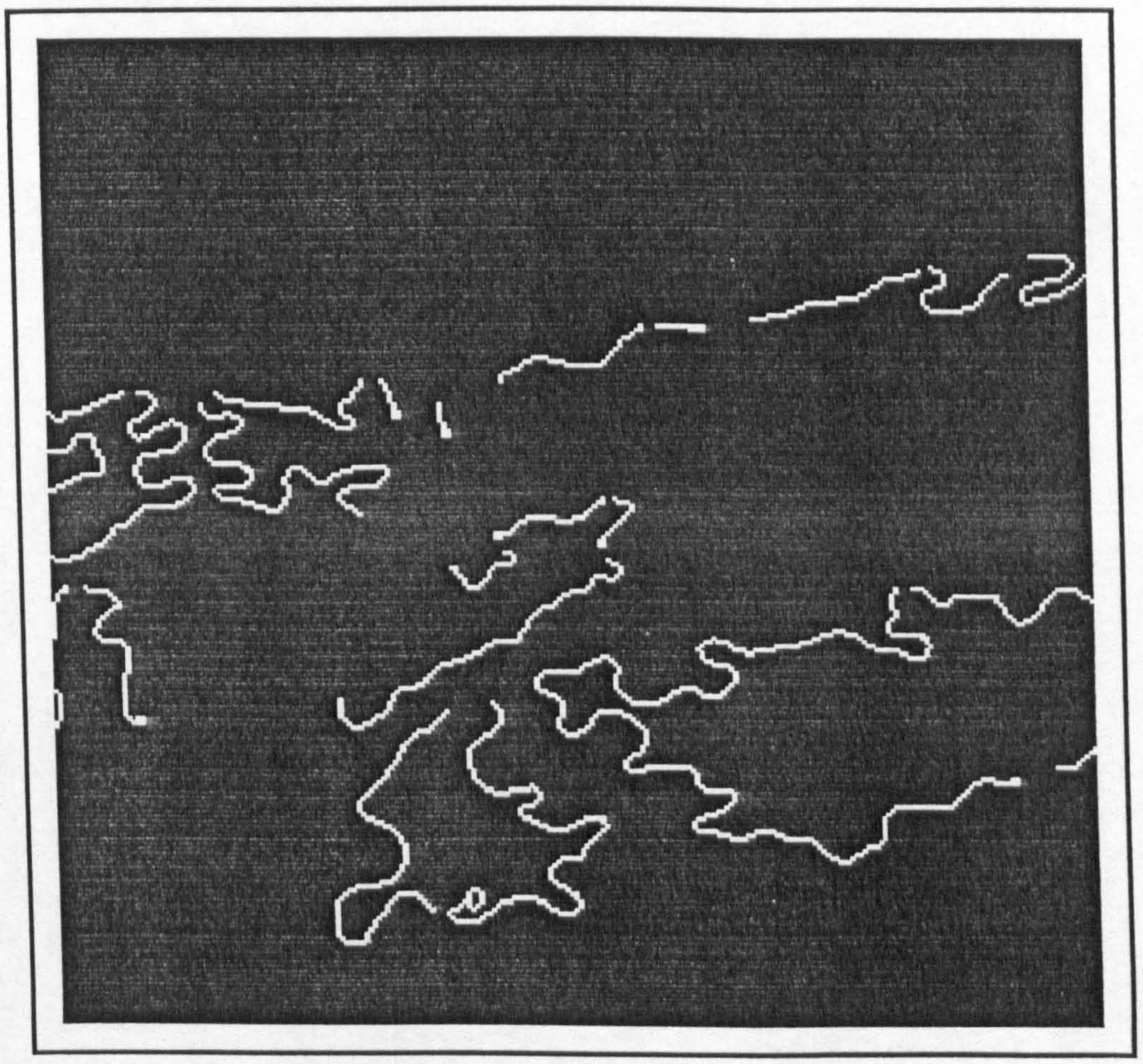


Figure 3.25: Features from the image in figure 3.24 separated into their component segments to illustrate fragmentation of features

example of an image generated by the line tracking process is shown in figure 3.24. The depth contours have given rise to well formed line representations of single pixel width. Only a small number of the characters are still recognisable. Many of the lines have small *blob* structures at one end.

To illustrate the performance of the system when a greater variety of features are present, the image in figure 3.26 was subjected to the tracking processes. The results of these processes are shown in figures 3.27 to 3.30. The tracking process seems to perform well in whatever grey level environment it is used. In order to illustrate the variation in detection of ridge profiles with the *factor*, as defined in figure 3.16, a series of images was produced with a varying factor value (see figure 3.28). The optimum setting for this factor was approximately 1.05. Below this value noise in the background regions gave rise to spurious detections and above this value the line work gradually became fragmented.

3.4.3 Limitations Of Data Driven Line Tracking

The tracked segments consist of fragments of lines which have been broken due to a number of factors. One reason for fragmentation is that the raster scan operation in the tracking algorithm first encounters a pixel in a line at a local maximum in that line, the line will first be tracked in the direction with maximal grey level and then a new segment will be started to follow the line in the opposite direction. The nature of generation and storage of chain code data does not allow a connected representation to be created during the tracking process when this type of fragmentation occurs. However, it is a trivial matter to reconnect these segments after tracking on the basis of coincident end points. Alternatively, fragmentation can be caused by the thresholding operation within the tracking routine. It may be that a line

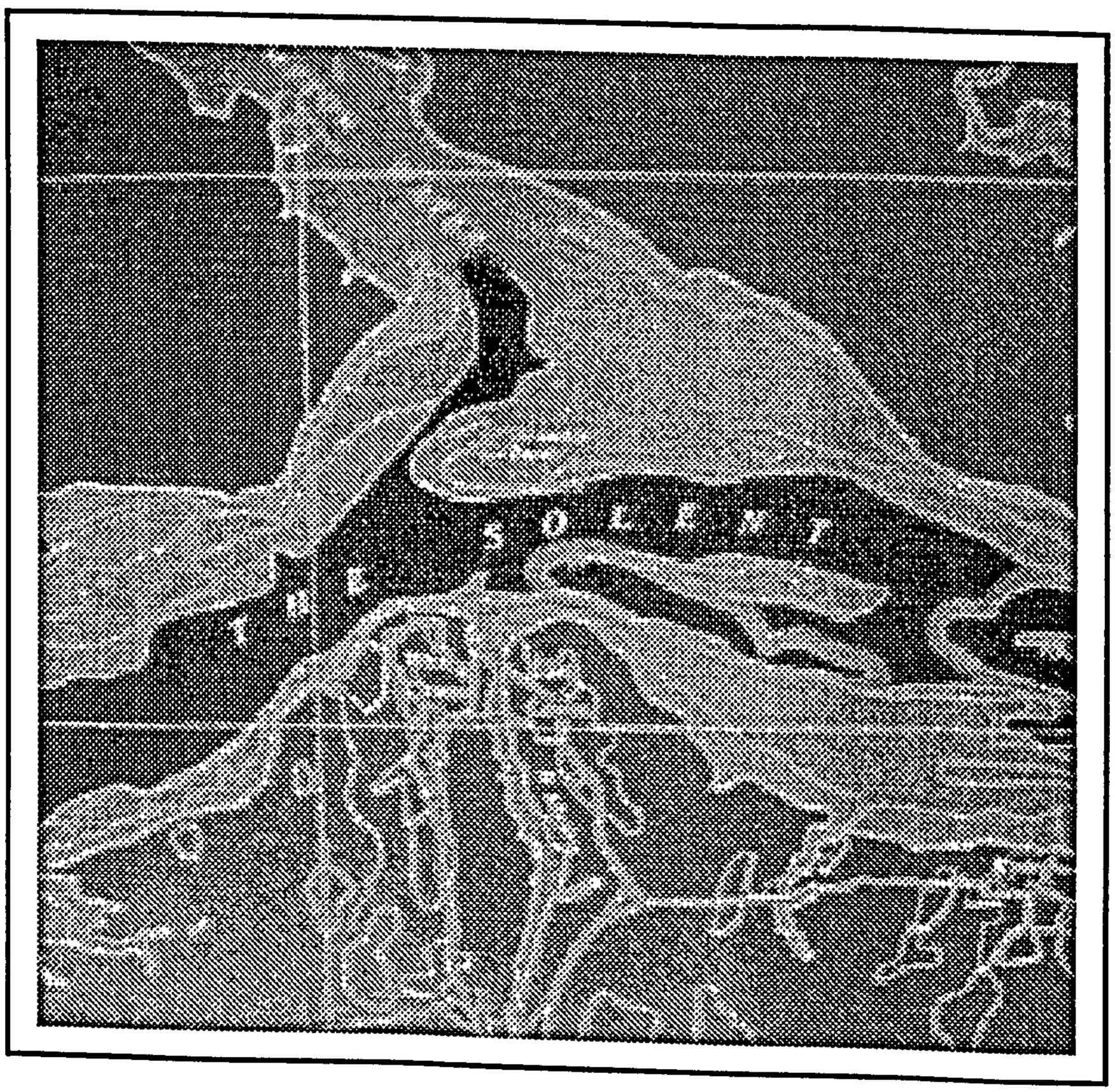


Figure 3.26: A chart image including coastline and shallow water

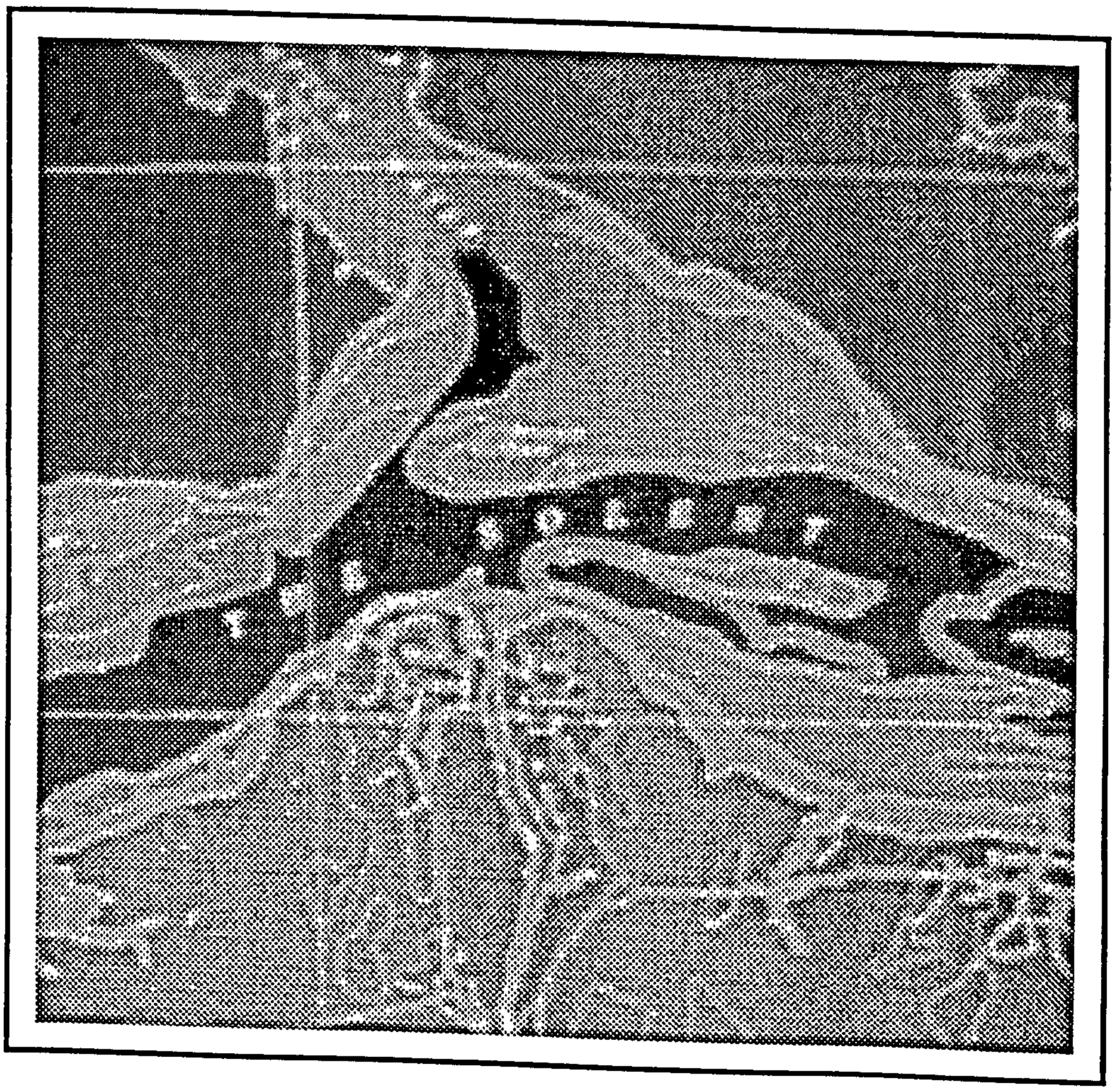


Figure 3.27: The image in figure 3.26 after Gaussian Blurring

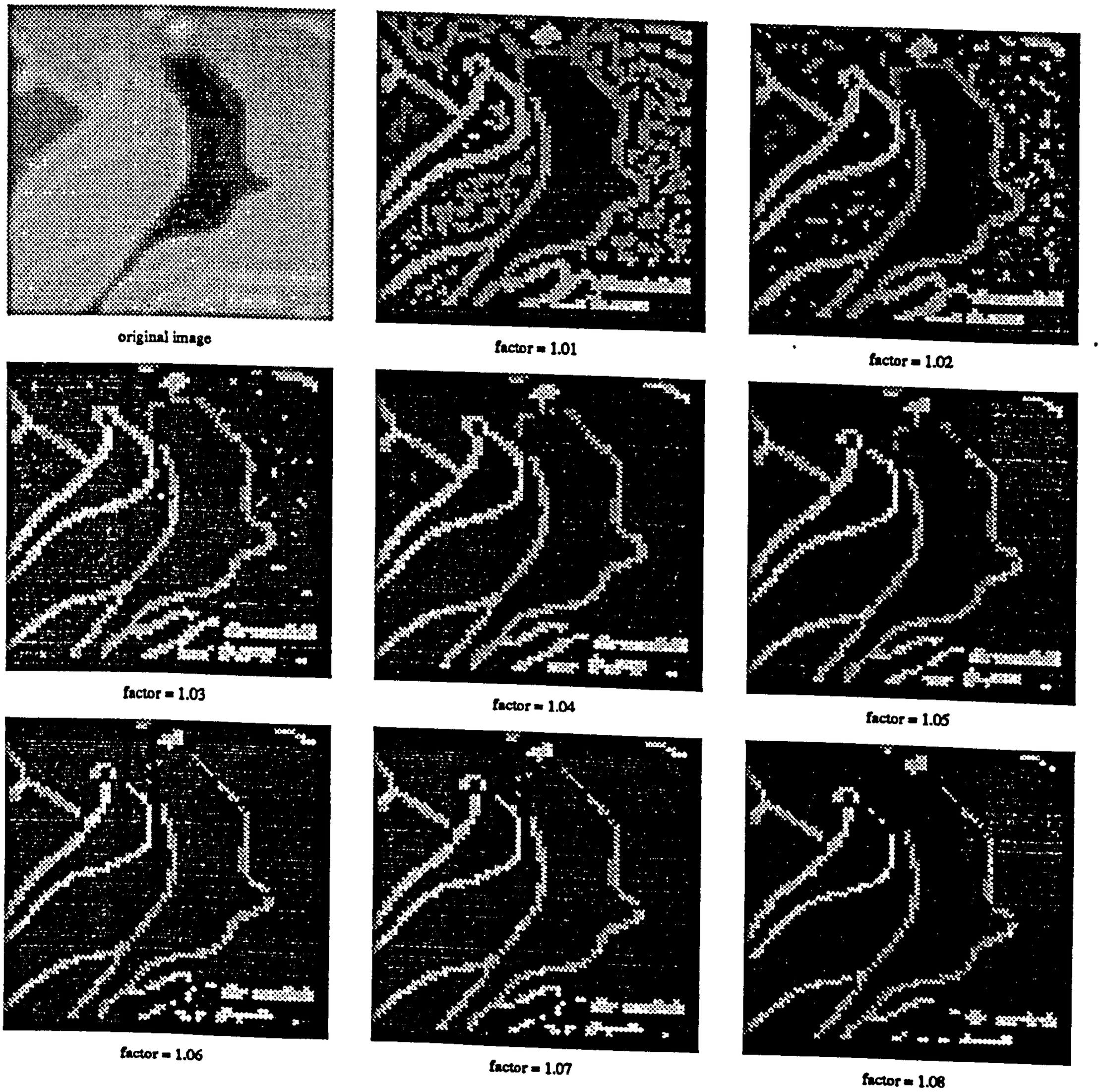


Figure 3.28: Result of variation of the *Factor* in Double Adaptive Thresholding

Figure 3.29: The image in figure 3.26 after Double Adaptive Thresholding

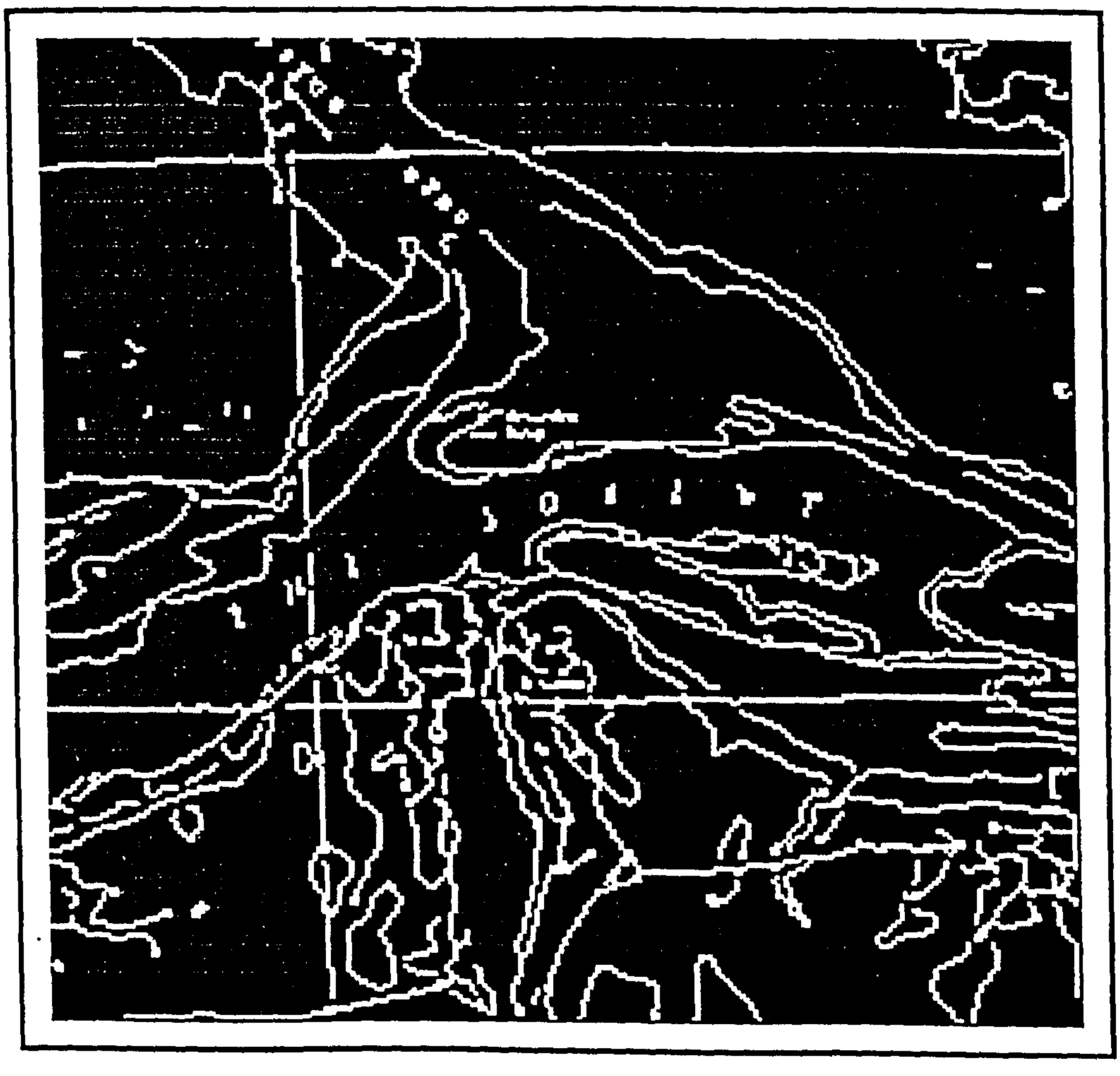


Figure 3.30: The image in figure 3.29 after tracking

which should obviously be continuous is not recognised as such because it does not quite meet the thresholding criteria. A further reason is that the line in the image is interrupted by graphics; perhaps a depth indication to mark a contour or an intersection with a map grid line or compass rose.

To assess the level of fragmentation an interactive routine was written which allowed observation of the individual segments. The routine allows display of the segment data on the frame store monitor or it allows the user to view the data in numerical form in its raw storage format or in an interpreted form. Figure 3.25 demonstrates the fragmentation present in an apparently contiguous line segment in an image produced by the chain code analysis routine.

Figure 3.31 shows a synthetically generated image illustrating features of a type that the line tracker finds problematic. The line tracker finds little problem in generating an accurate visual representation of the features in this image (see figure 3.32). However, the two higher frequencies of sine wave are considerably fragmented due to their high curvature. The angular features were all tracked as single contiguous chains of vector codes. in examples of intersections the features have been split into two vectors. A slight gap is apparent in the vertical bar of the uppermost intersection example. This is because the computation of the cut off value, when considering continuation of the top vertical line incorporates contributions from too many previously marked pixels.

Real images will generally be blurred and contain significant amounts of noise. Figure 3.33 is a blurred version of the previous example with gaussian noise added. The level of noise added to this image is greater than that found in the type of natural images used in this research and is such that the modes of the background and line work in the grey level histogram of the degraded

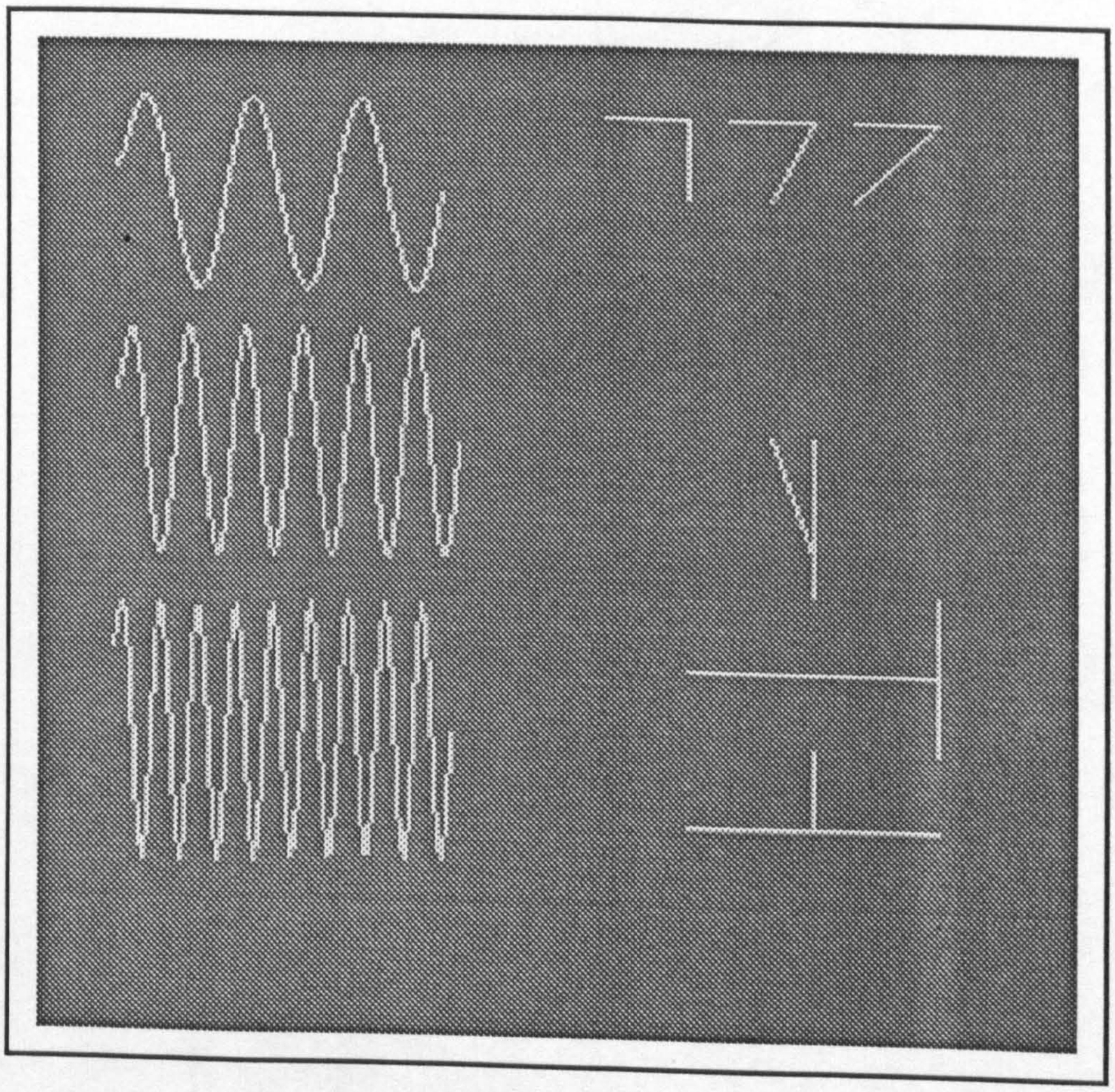


Figure 3.31: A synthetic image for testing the geometric robustness of the Tracking Process

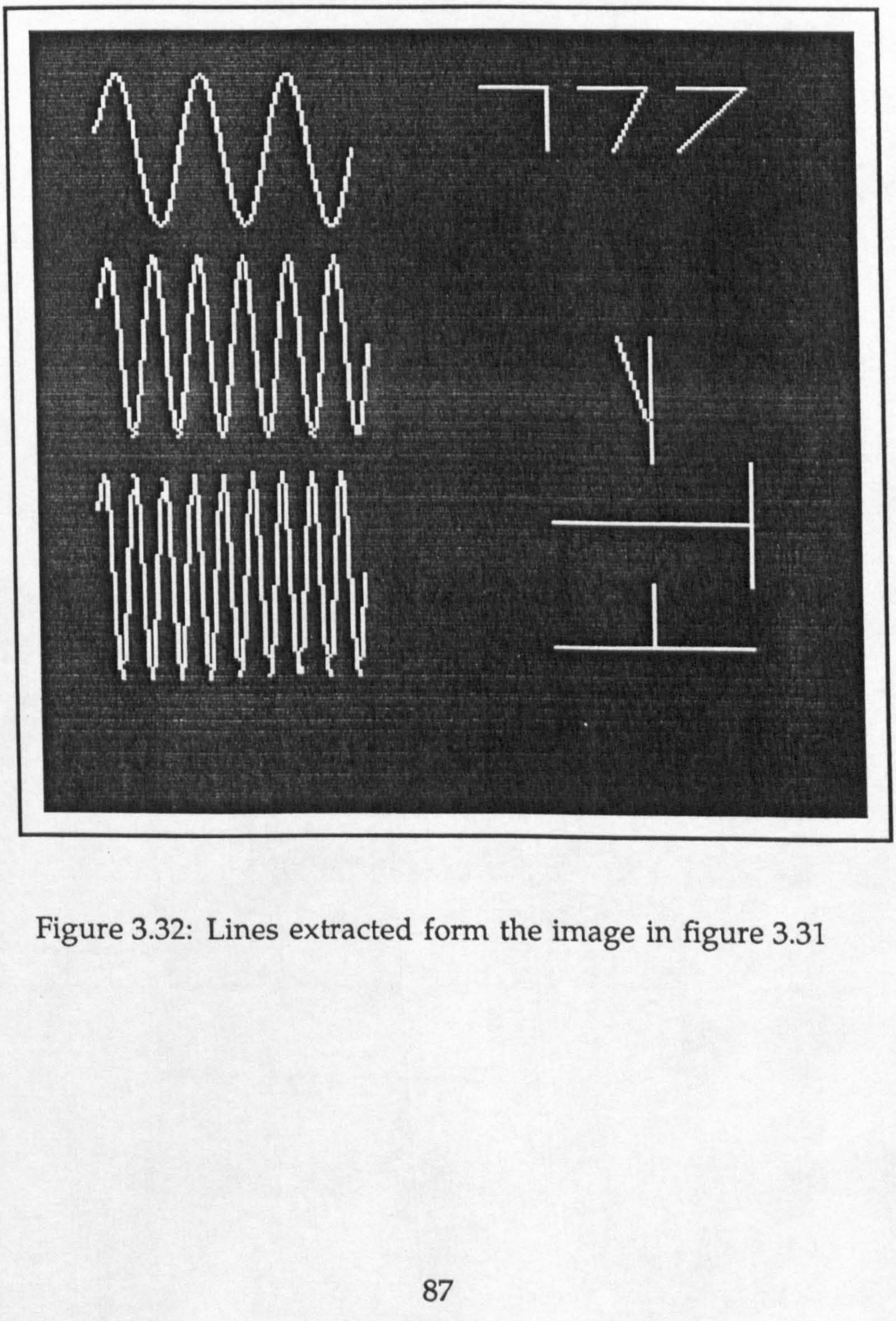


Figure 3.32: Lines extracted form the image in figure 3.31

image significantly overlap.

After tracking this degraded image (see figure 3.34), the visual representation of most of the features is slightly distorted. Fragmentation has occurred in all three of the sine wave examples and the most acute of the angular examples has been broken. The order of fragmentation in the intersection examples is similar to that in the clean synthetic image.

It can be seen that, although the line tracker implemented here is not highly robust, it is able to extract all the linework, albeit in a sometimes fragmented form. The next chapter addresses the problem of building features from the fragmented lines that have been extracted.

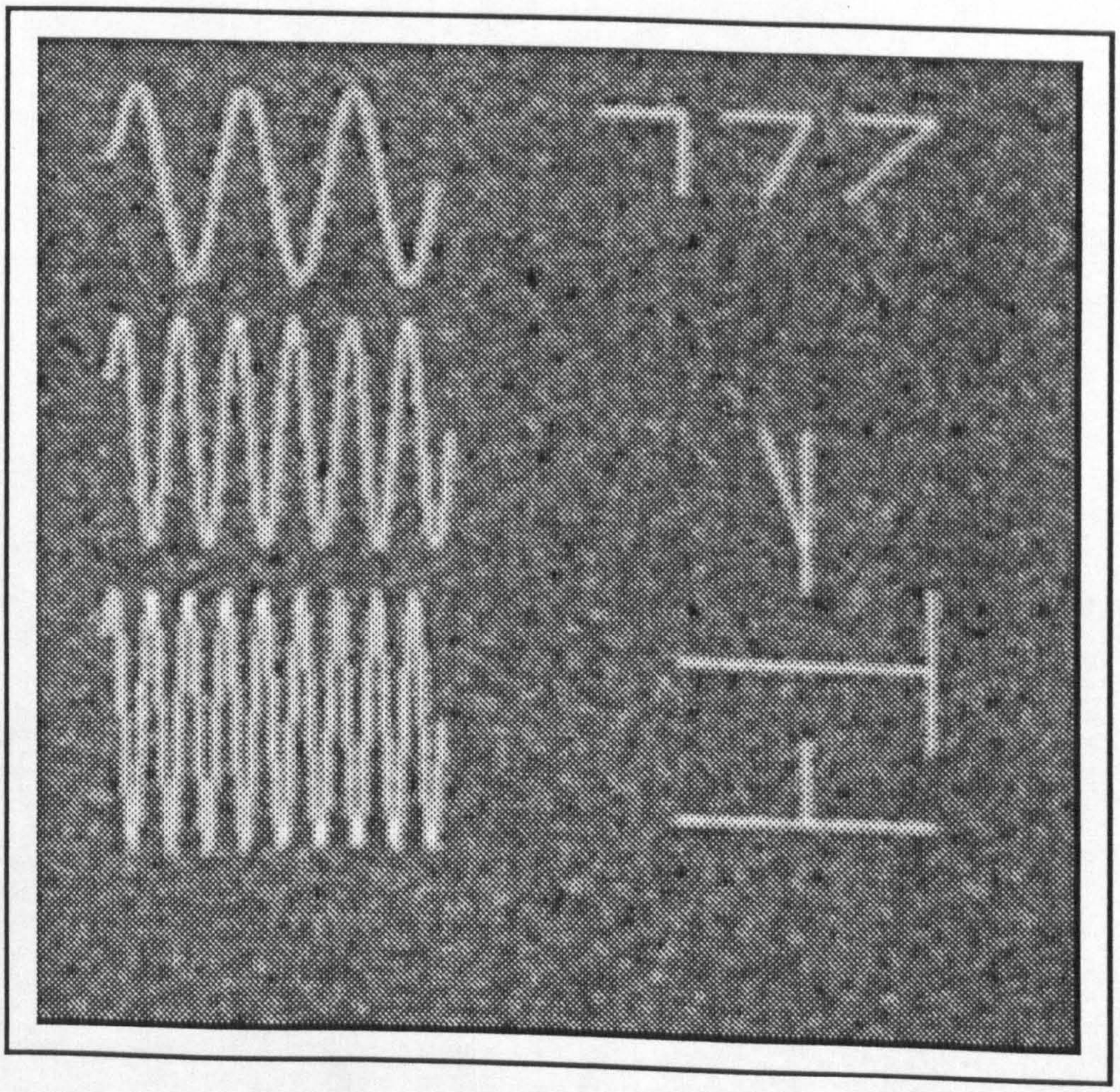


Figure 3.33: The image in figure 3.31 after blurring and addition of noise

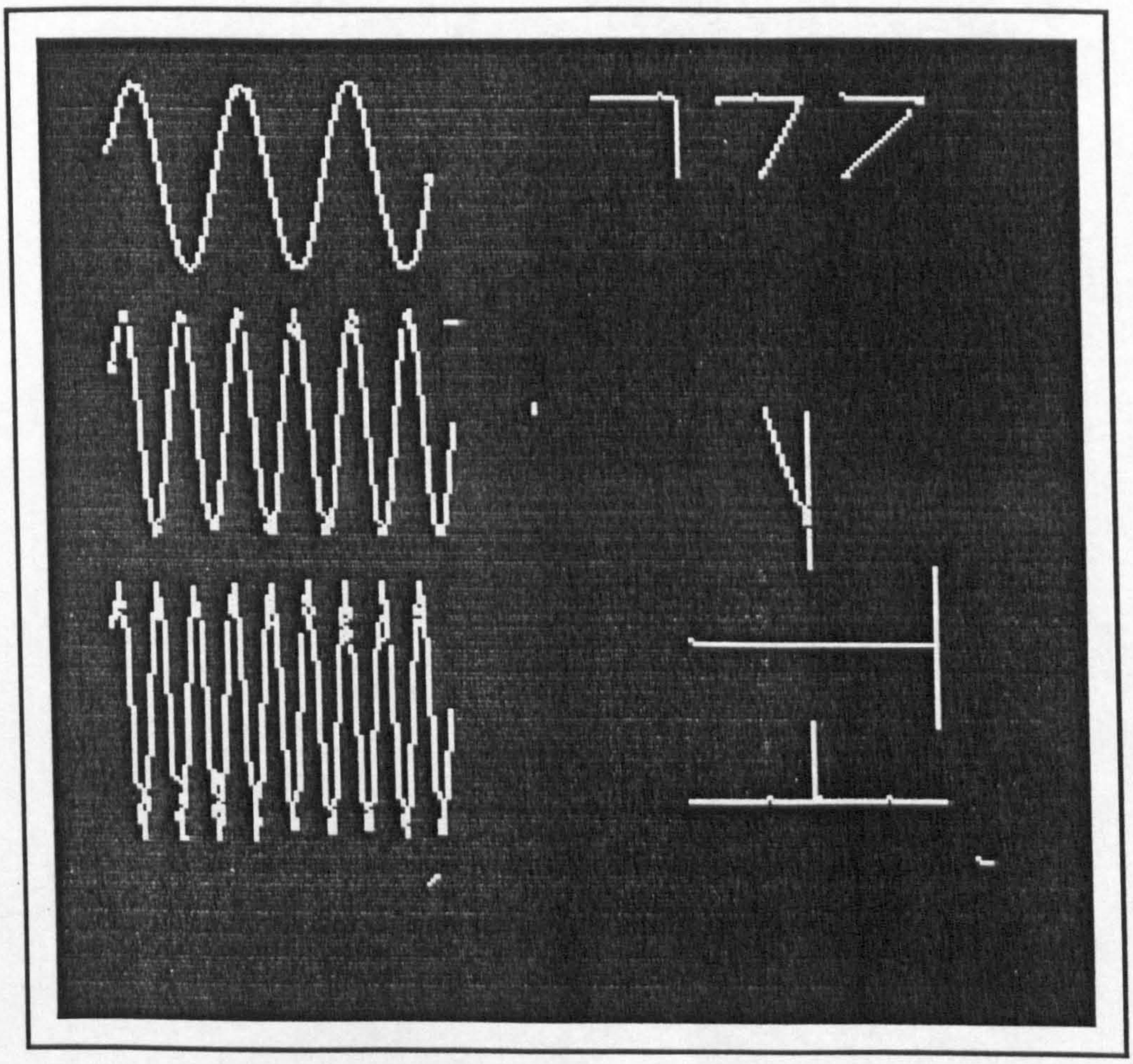


Figure 3.34: Lines extracted from the image in figure 3.33

Chapter 4

The Development of a Semi-Automatic System for Chart Feature Extraction

4.1 Possible Routes to Overcoming the Limitations of Data Driven Line Tracking

The previous chapter described the development of a basic line tracking scheme and illustrated typical results of the execution of this tracker on a chart image. Whilst assessing the line segments produced by line tracking an understanding of the type and quantity of a priori knowledge that could help in the process of connecting the segments in a meaningful manner was obtained. Knowledge about the domains of both images in general and of sea charts in particular could be useful in interpretation of the data. The remaining problem was therefore to develop a system within which this knowledge could be incorporated.

The possible overall schemes which might provide a solution were seen to

be to develop :-

- a conventional feature extraction process containing a complicated set of heuristics able to reliably build features from the low level image data

- a line tracker/feature builder which relies on an intelligent knowledge based system (IKBS) for its decision making processes in building features from low level image data

- a data driven IKBS which could build features by matching configurations of previously derived image segments to rules in its knowledge base

- a semi-automated feature extraction system which performs feature building using line segments from a low level tracker which initially relies on a good deal of user interaction but which could interface to a cooperating goal directed IKBS in order to reduce this interaction

The first of these options would seem to be an enormous programming task, the end result of which would be a huge computer program in which the implicit nature of the expression of the knowledge used would make it difficult to understand when returning to modify parameters or add further knowledge to deal with additional unforeseen situations.

The second option would offer a solution to the problem of developing a system in which the knowledge representation is explicit and easily modified and extended. In general the pixel by pixel decision making process could be made by low level image knowledge in procedural form and the knowledge based system need only be brought in when this low level knowledge could not find a continuation of the line feature. However the drawbacks of this type of system which were foreseen were that the data available for the

knowledge based system to work with would be in the form of pixel matrices and an incomplete set of line segments. Expressing a pixel matrix in the form of knowledge representation used for knowledge based systems would require an enormous amount of computer storage. Performing calculations on this would require creation of rules which would be complicated and difficult to understand and the speed of processing was likely to be very slow indeed. The alternative to dealing with the pixel matrix within the knowledge representation scheme of the IKBS would be to revert to algorithms coded in a conventional procedural form invoked by the IKBS when necessary, but this would be very frequent indeed particularly for the low level pixel based processing, effectively reducing the system to that of the first option. Also the system would only work effectively when an appropriate knowledge base had been established and so would be a very long term research project.

The third option is an attractive one, but would also have to be considered as a long term research project and would not offer a working interpretation system until the development and testing of the system was complete.

The last option was seen to be the most attractive. This could ensure a working system albeit with some user interaction. The system could be automated more fully by recognising the knowledge the user brings to bear at the point of interaction with the system. That knowledge could then be embedded in a rule based system which interacts with the semi-automated system instead of calling upon the user. A complete set of line segments, within the limitations of the low level tracker, would be available for the knowledge based system to refer to when enquiries about the environment were necessary in a decision making process.

It was thought from the outset that a single system which could operate on the whole spectrum of levels of abstraction involved in the process of line

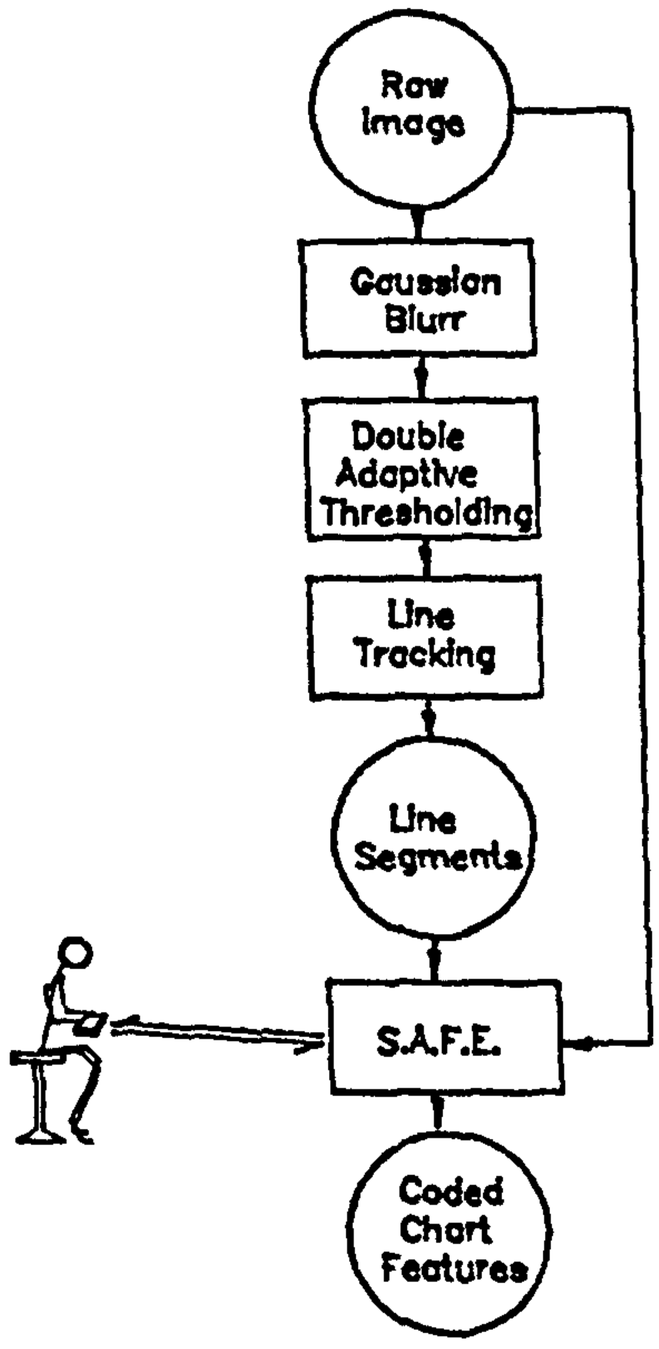


Figure 4.1: Schematic diagram of the SAFE environment

extraction and map interpretation would allow for feedback in the system. For example, the inevitable thresholding in the early stages of processing can often produce misinterpreted data. Evidence could be collected at a later stage for this misinterpretation and the low level process could be re-invoked with appropriate adjustment to the thresholding.

4.2 Development of the Semi-Automatic Feature Extraction (SAFE) System

4.2.1 An overview of the SAFE system

The semi-automatic feature extraction system (SAFE) is designed to build chart features automatically from the line segments of the chart image produced by the line tracker. User intervention is solicited to assist SAFE if it

encounters difficulties in the feature building process, e.g. if it encounters an ambiguity in the choice of segment to link to a particular feature. The SAFE environment is illustrated in figure 4.1.

When working with the SAFE system the user faces a video image of a portion of the sea chart. This is the raw image from which a file of line segments has been created using the line tracker. The process of building up of chart features from the line segments is carried out and this is reflected in graphic overlays, on the chart image, showing the state of the feature. The user interacts with the system, by responding to menus and prompts displayed on the operator console, through the keyboard and cursor. The feature building process may be performed under the strict control of the user. Alternatively a certain amount of decision making in feature building may be performed by the SAFE system by invoking an automatic mode. Each feature must be *opened* and *closed*. The process of opening a feature initialises the internal data store for the feature. Closing the feature causes the system to prompt for identification of the feature that has just been built. The labelled feature is then stored as part of the chart feature data base. To initialise the system the user must name the previously generated file of chain coded line segments with the image in response to a prompt. Table 4.1 shows the options available in the main menu which is displayed immediately after initialisation.

The menu dynamically alters to display only the options relevant at a given instant, for instance the *open* option is not displayed when a feature is open and the *modify* option is not displayed until the current feature contains some data.

There are two working modes; automatic and manual.

open	Reinitialises the internal data store in order to begin building a new feature
write	writes the current feature to the output chain code file after prompting for identification of the feature
close	closes the current feature
modify	takes user into feature modification menu
show	displays the current state of the feature data on the users console
print	prints the current state of the feature data

Table 4.1: Options in the Main Menu of the SAFE system

add	add a synthetic segment to the feature data using the cursor
delete	Delete a segment from the feature
split	Split a segment and retain one portion in the feature
erode	erode a segment pixel by pixel from the end specified using the cursor

Table 4.2: Options in the Modification Menu of SAFE

Automatic Mode

The system starts off in automatic mode, searching for the best line segment with which to begin tracing a map feature. As line segments are added to the current feature they are displayed graphically over the video image of the map in a high grey level.

Once a start segment has been found the start and end coordinates of the chain representing the feature are stored. A search is now made for another segment close to the end of that segment.

It then continues searching automatically for extensions to the feature. No user interaction is required until an ambiguity is reached or no line segment is found to extend either end of the line feature.

When an ambiguity is encountered the system displays what it considers to be the prime candidate for continuation of the feature and asks the user if this is the correct direction in which to proceed. If the response is negative the image is regenerated over the graphic display and the next candidate displayed in the same manner, up to a maximum of five candidates. When all candidates are rejected or no line segments can be found to continue the feature, the system drops into manual mode.

Each time a line segment is added to the feature the coordinates of the free end of the segment are used to redefine the end coordinates of the feature. When no more data can be found to continue the feature in the current direction the feature is automatically extended as far as is possible in the opposite direction. When no more data can be found in the new direction the limit of automatic extension has been reached and the system drops into manual mode.

Manual Mode

The operator is now presented with a menu of options. The menu dynamically alters according to the current state of processing.

A line segment can be found using the *find* option which uses a cursor to locate the line segment which passes closest to the cursor position. Again a maximum of five candidates are offered.

Modifications can be made to the state of the feature by selecting the option which takes the operator into a modification menu. Within this menu the operator can generate his own line segment. He can remove segments which have already been added to the feature, split segments and remove part of the segment from the feature or erode away the end of a segment. These modifications can be made by selection options from the modification menu shown in table 4.2.

Other options in the main menu include looking at the numerical data corresponding to the feature, performing a link up operation, which ties the ends of corresponding segments together, and writing the feature to the output data base, which generates a new chain code of the feature linearly interpolating between the composite segments.

When a feature is complete the user provides a feature code, indicating for example that it is a coastline or a contour of a certain depth, which is stored with the feature for future use.

4.2.2 Data Structures and Algorithms within SAFE

This section briefly describes some of the important aspects of the implementation of SAFE.

Inputs to SAFE

The input data to SAFE consists of the raw image of the portion of the sea chart to be interpreted, and a file of chain coded segments extracted from that image using the techniques described in chapter 3.

Accessing Data

All variables within the SAFE system (i.e. lines, line features, points, regions, areas etc.) are accessed by SAFE through a *domain description file.* This holds information about the types of variables, where to find the data representing the variables and some rudimentary measures of properties of the variables. One advantage of accessing variables indirectly, through the domain description file, is that accessing or deleting a desired line segment in the chain code file without such information is difficult.

When the SAFE system is first initialised it produces a separate record for each of the line segments in this file. This record contains the information listed below.

- A character string "LINE" to specify its type
- A pointer to the start and end of the line within the chain coded input file
- The start and end image coordinates of the line
- The number of odd and even chain code values in the line segment (and hence the total number of points and the line length)
- The state of the variable (e.g. free for inclusion in a feature; included in current feature; deleted etc.)

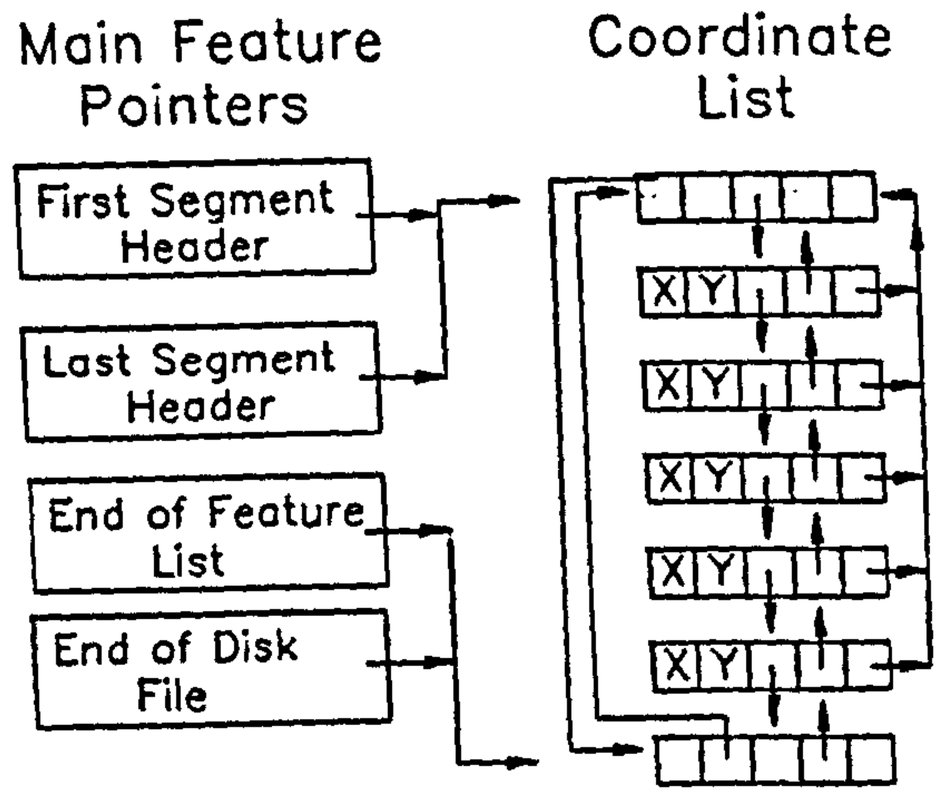


Figure 4.2: The Internal Feature Representation of a Single Line Segment

The Internal Representation of a Line Feature

When building a feature its internal description must be of a form suitable for editing. The scheme adopted here is to explicitly store each image coordinate of a line as a series of cartesian coordinates (see figures 4.2 and 4.3). These are stored as separate records in a direct access data file. They are linked in both the forward and backward sense by a system of pointers.

Each line segment in the feature has a separate header record. These headers provide a higher level of linkage within the feature list in order that the feature data may be scanned quickly, from segment to segment, without the need to refer to each pixel of every segment. In addition, each pixel record contains a pointer to its header record. Four main pointers to the list, shown in figures 4.2 and 4.3, are held in core memory in order to allow an initial access to the feature.

Basis for Finding the First Segment of a Feature

The first segment of a feature can be found in manual mode, using a *find* operation, or in automatic mode. In automatic mode the basis for selection of

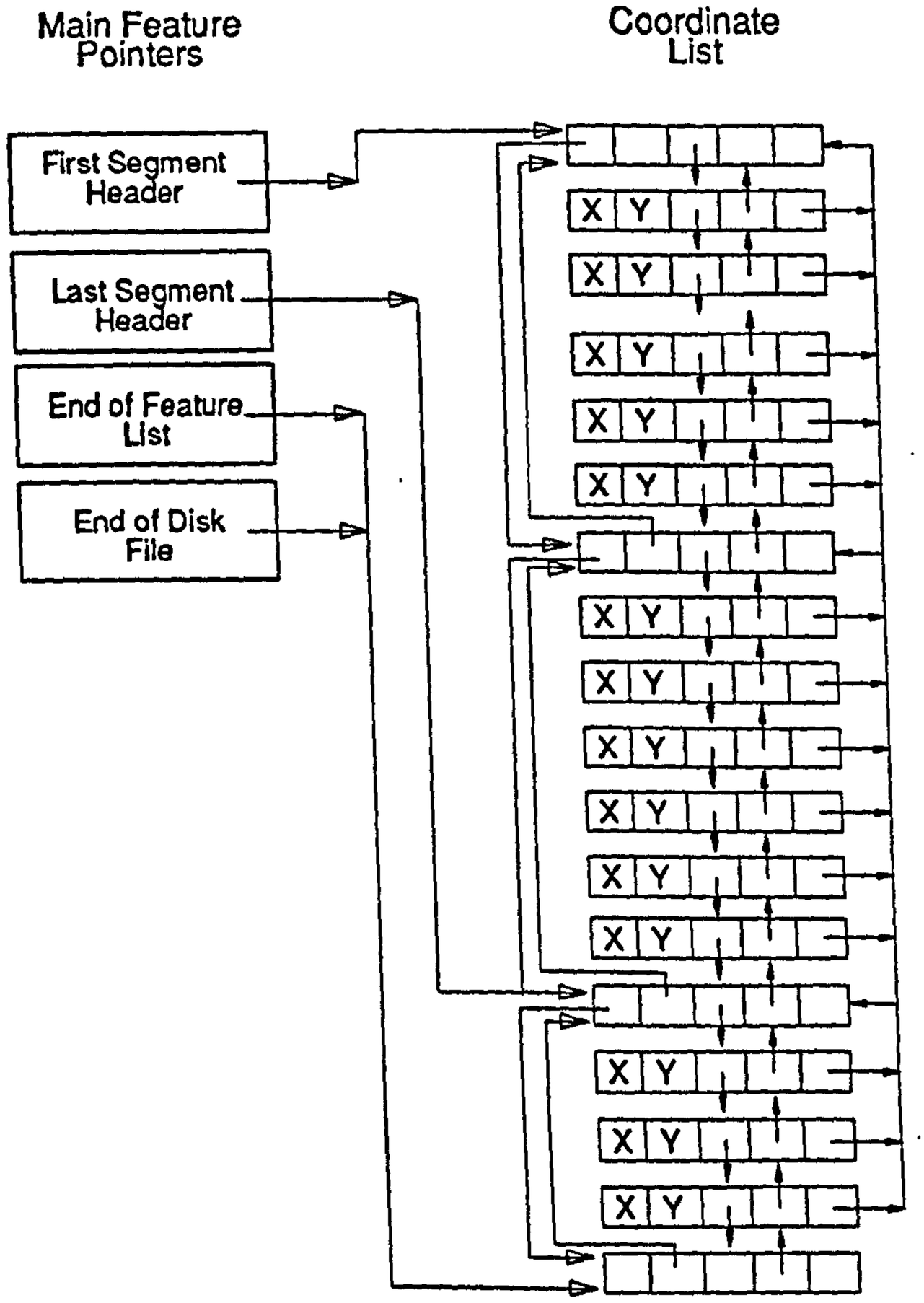


Figure 4.3: The Internal Feature Representation after a number of Line Segments have been added

the first segment is its length. The length is computed from the pixel counts in the environment description file and the longest segment is chosen. The reasoning behind this is that when selecting a first segment manually, the user is generally drawn to a long uninterrupted segment of a feature of the chart in an uncluttered area. The low level tracking works well in such areas and therefore is likely to have produced its longest segments from these areas. Initially it was thought to be logical to search for a start segment near to the border of the image. Feature building would then only need to proceed in one direction, but this was found to be an unnatural way of building the feature and could not be used for features which do not approach the image boundary.

Criteria used in Searching for an Extension Segment

The search for a segment to extend a feature is performed very simply within SAFE on the basis of proximity of one end of the feature to an end of a line segment. A 10 pixel square box centred on the end of the feature is used for this operation. Three cases must be considered :-

1. If one, and only one, line segment ends within the 10 pixel box then it will be linked automatically and SAFE will proceed uninterrupted.

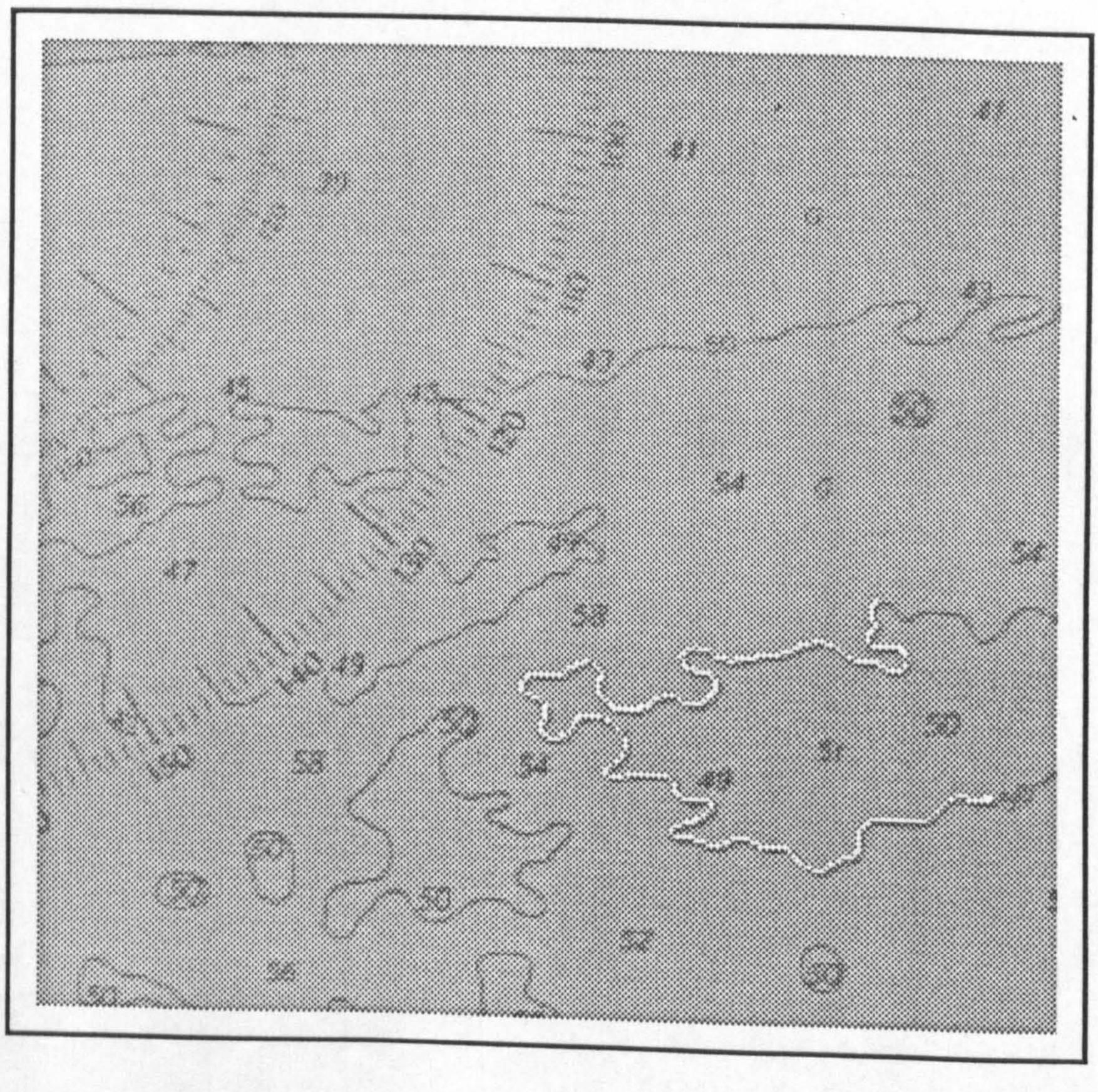
2. If no line segment end points are found within this box then the system reverts to manual mode and the user is presented with the main menu.

3. If more than one line segment end point is found within the box then each is presented to the user as a candidate extension and the user is asked to accept or reject each one until the correct segment is presented or until all segments found by SAFE are rejected.

4.3 Sample Terminal Session with the SAFE System

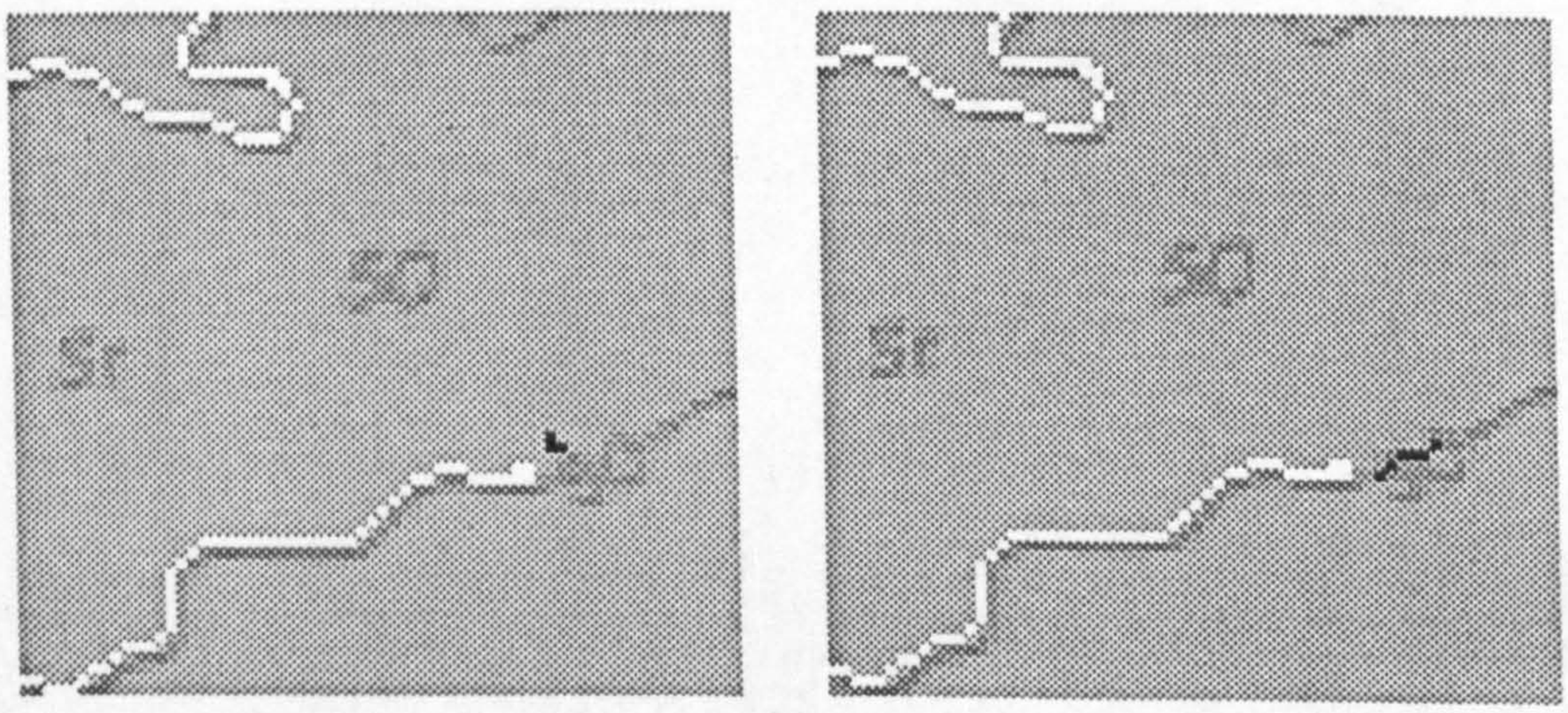
4.3.1 Initialising the System

The user is presented with an image of the chart portion to be interpreted and is asked to associate a file of chain code data with the image to initialise the system. The system finds an appropriate line segment to begin the interpretation.



4.3.2 Reaching an Ambiguity

It then finds an ambiguous situation where two small segments belonging to a depth marking within the contour are the candidate extensions. These are presented to the user, in order of proximity to the current end of the feature. The segments which have been found are obviously not good candidates for extending the feature. This is one situation where a small amount of extra knowledge could reduce the need for user interaction.

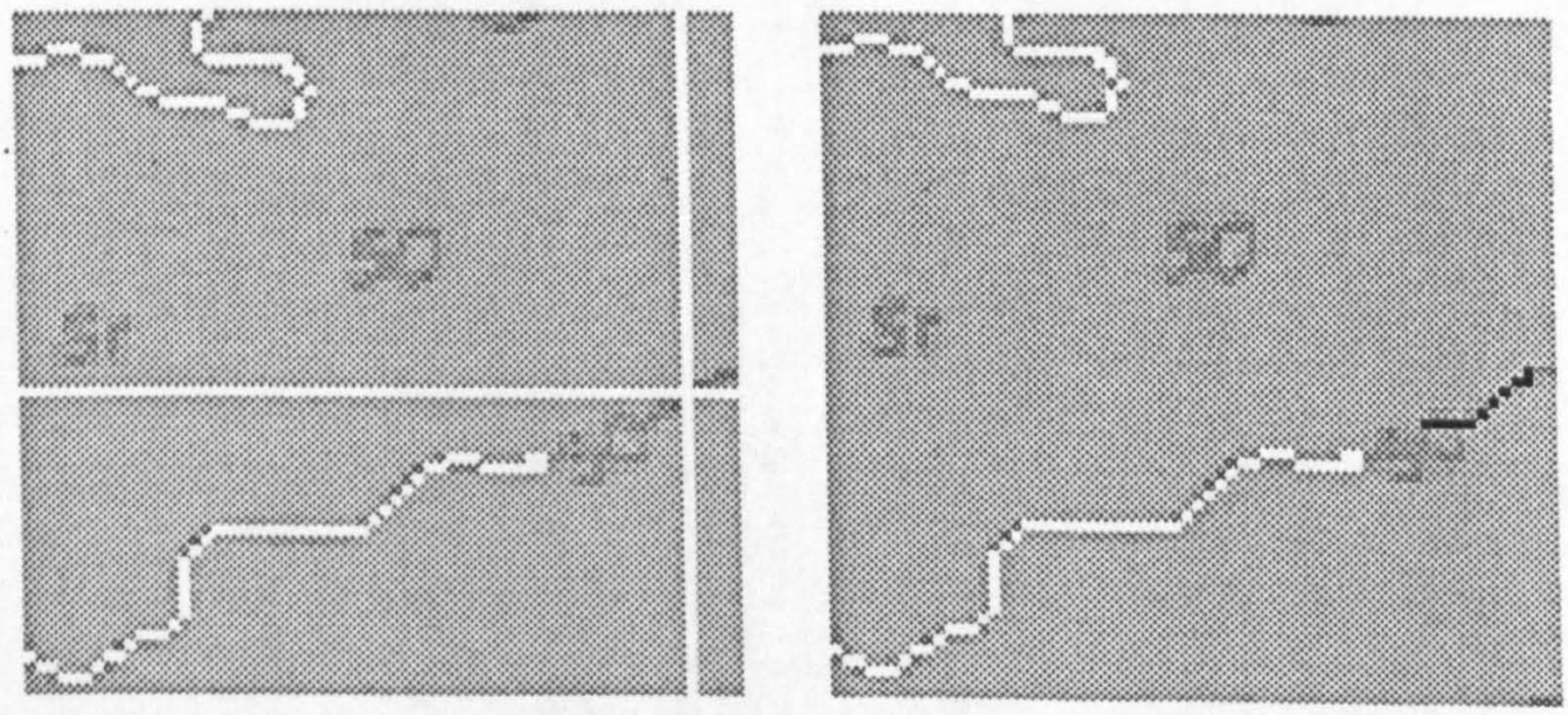


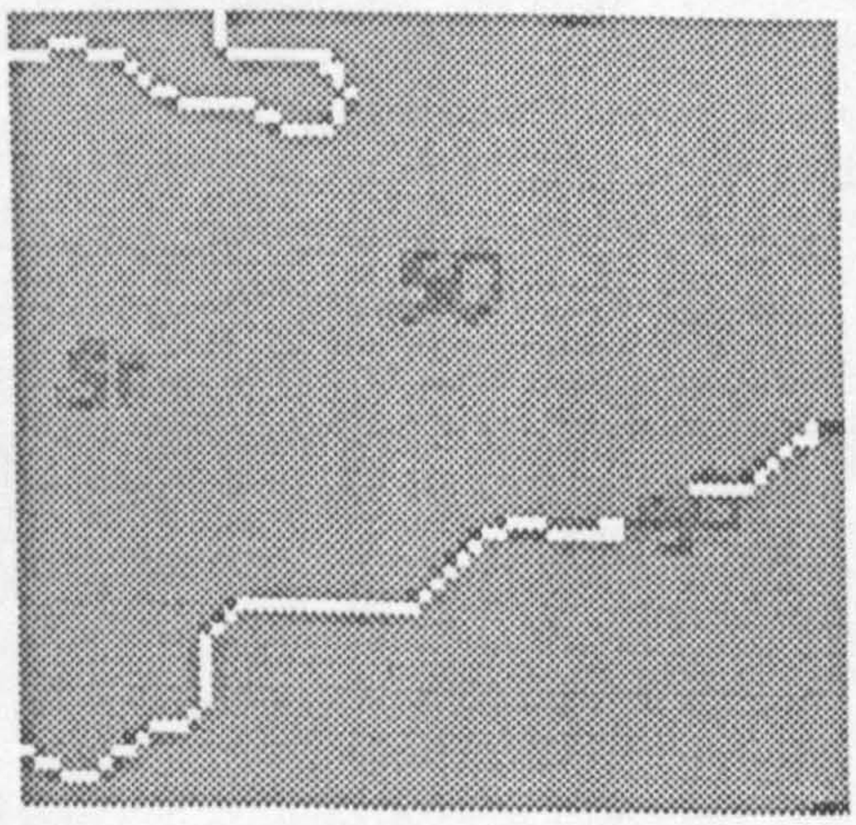
The user rejects both of these segments which are erased from the screen display.

4.3.3 Finding a Segment in Manual mode

The user is now presented with the main menu and selects the *find* option and moves the cursor to the portion of the line beyond the depth marking.

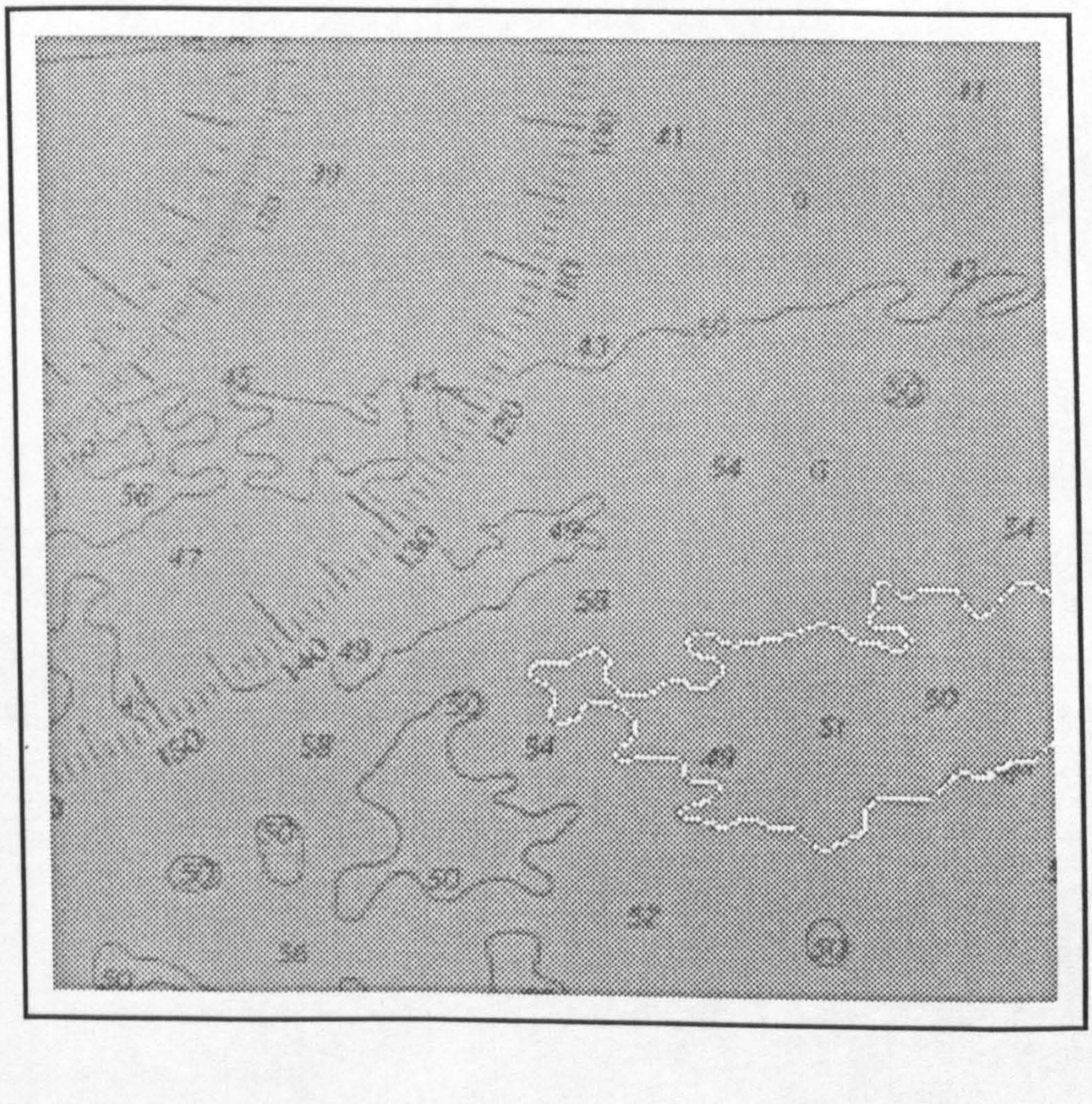
The system finds this segment in the chain code file, displays it in black and asks the user if that is the one required. The user replies that it is, at which time the new segment is displayed in white. If left, the gap between the two segments currently forming the feature will be interpolated as a straight line.





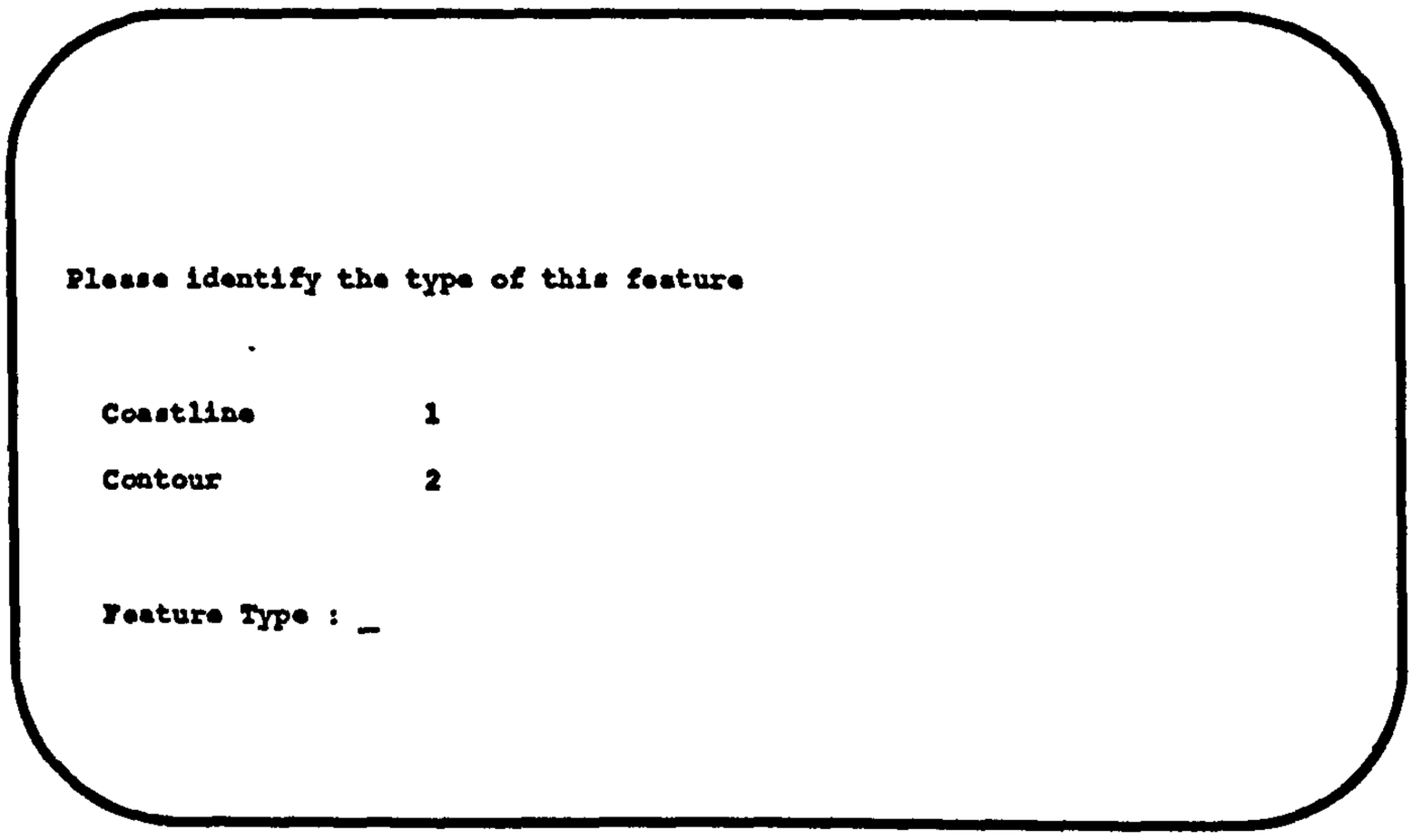
4.3.4 Returning to Automatic Mode

The user can now put the system back into automatic mode. The system will recognise that it has reached the fullest extent of the feature in the current direction. It therefore searches in the opposite direction starting at the other end of the original segment. One segment remains to be added to the feature which can be done without interaction as there is no ambiguity present.



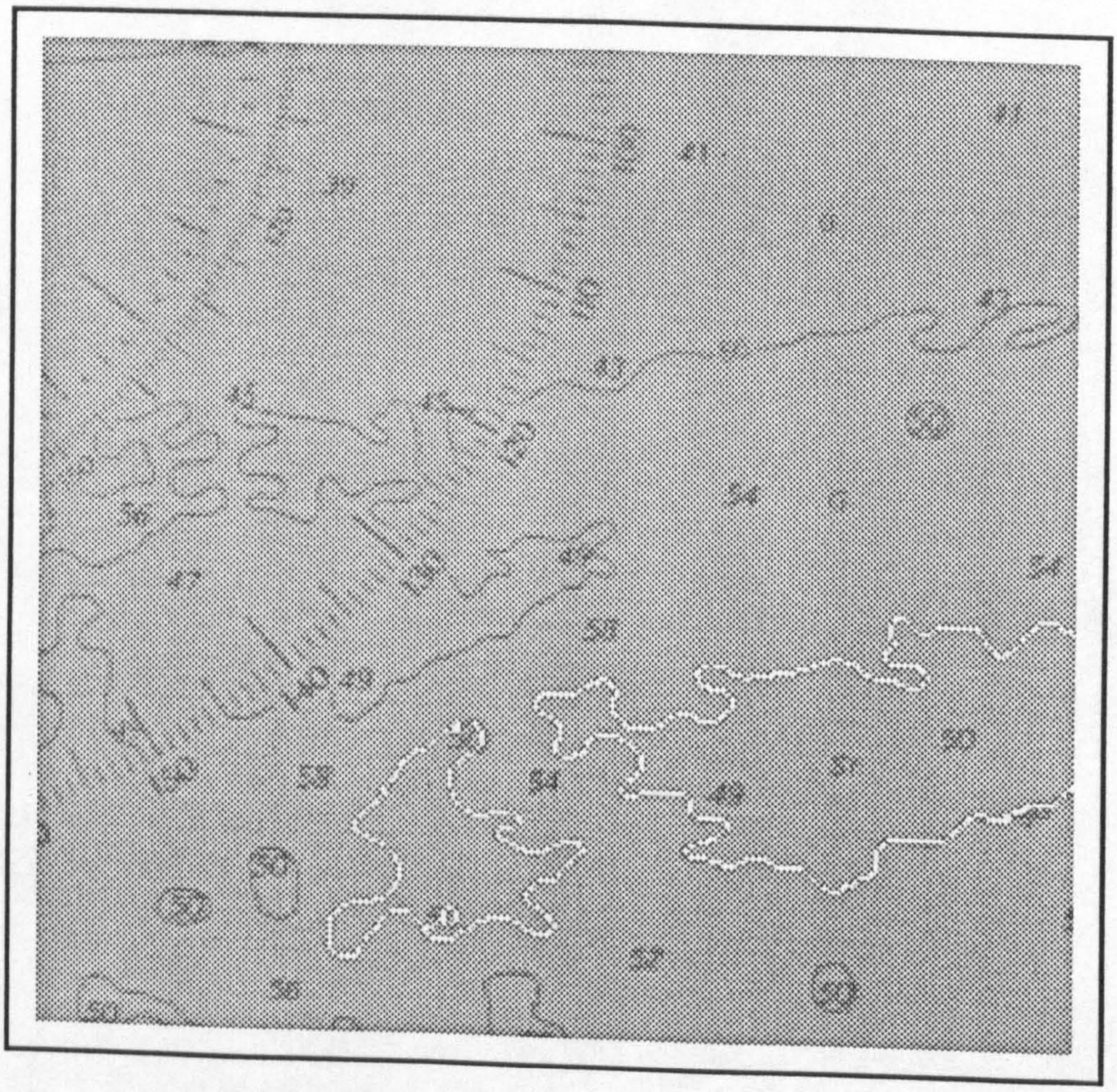
4.3.5 Terminating a Feature

Having collected all the line segments belonging to the feature, the system drops into manual mode. The user then terminates the feature by selecting the *write* option and the *close* option. The user is prompted to mark the feature with an identification code indicating the type of feature (e.g. coastlines, contours, buoys etc.) and it is then written as a single contiguous feature in the form of a chain code to the feature file.



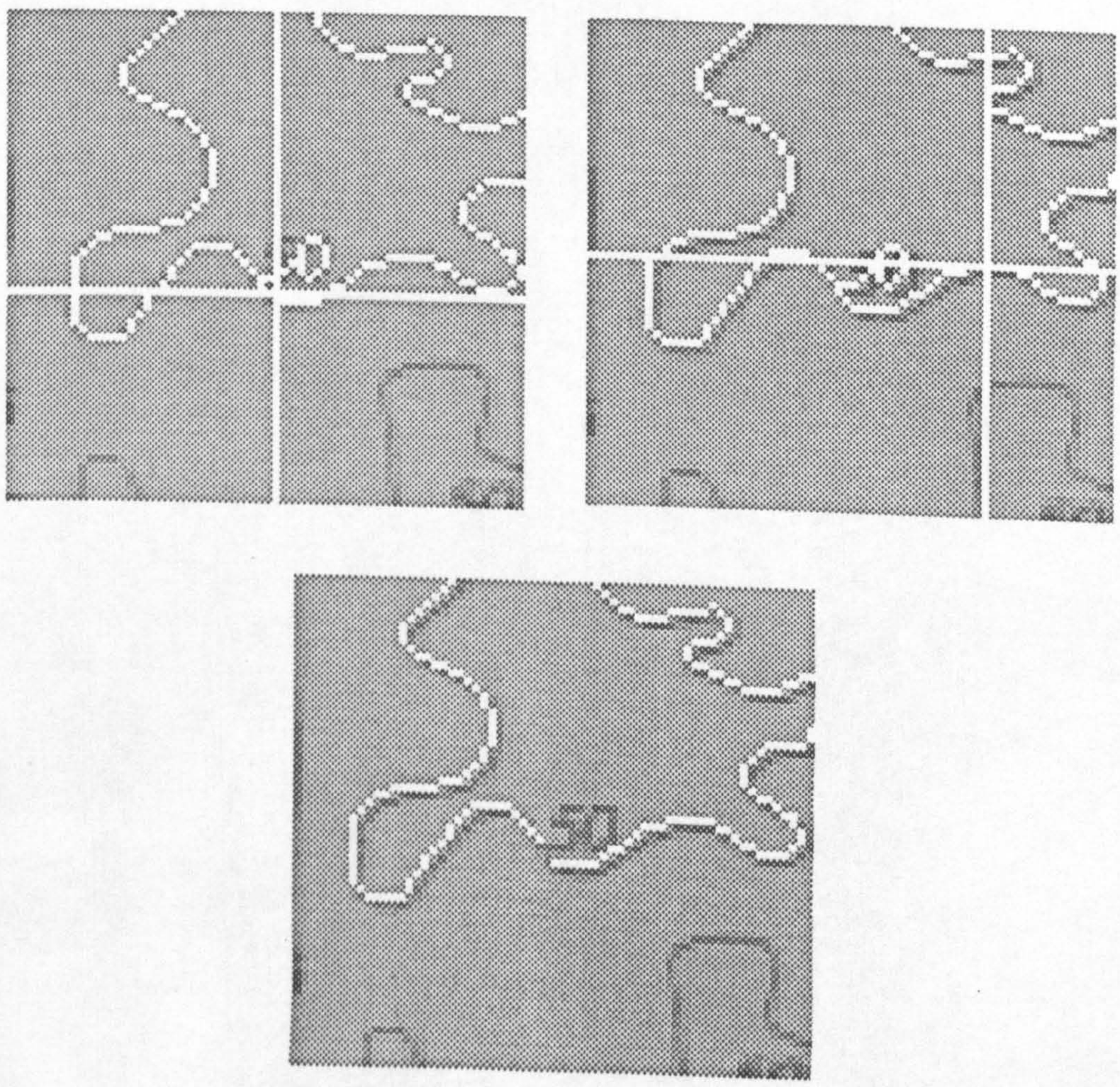
4.3.6 Building a Second Feature

By selecting automatic mode once more the user invokes the system to initialise another feature. The whole of this feature is created automatically, but deficiencies in the tracking of the feature necessitate slight modification before closing the feature. These modifications (segment splitting, erosion and removal) are shown in the next three pages.



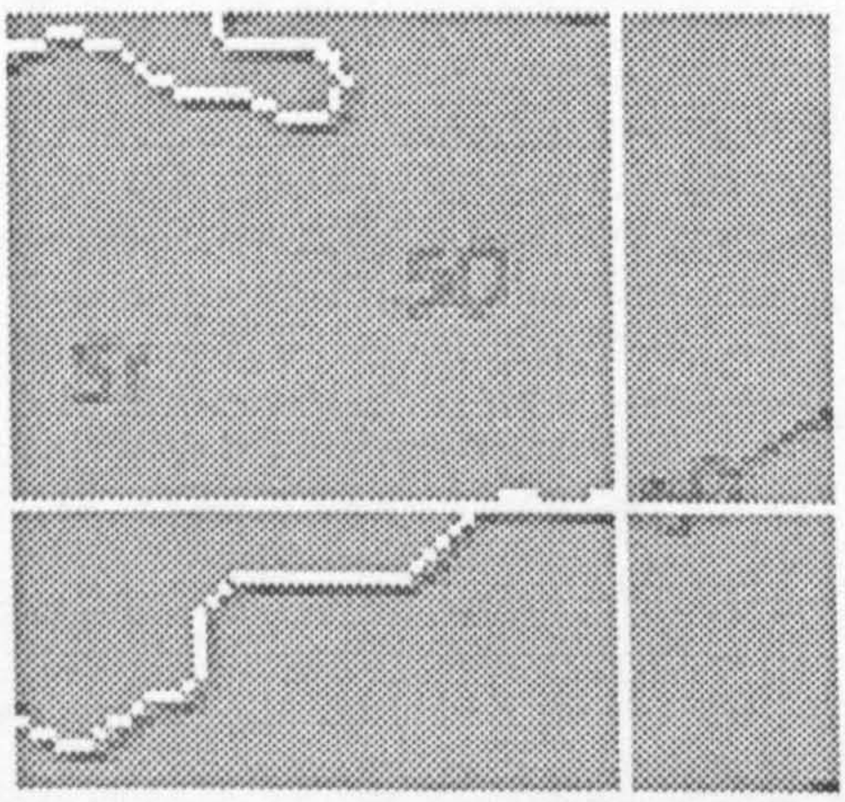
4.3.7 Splitting a Segment

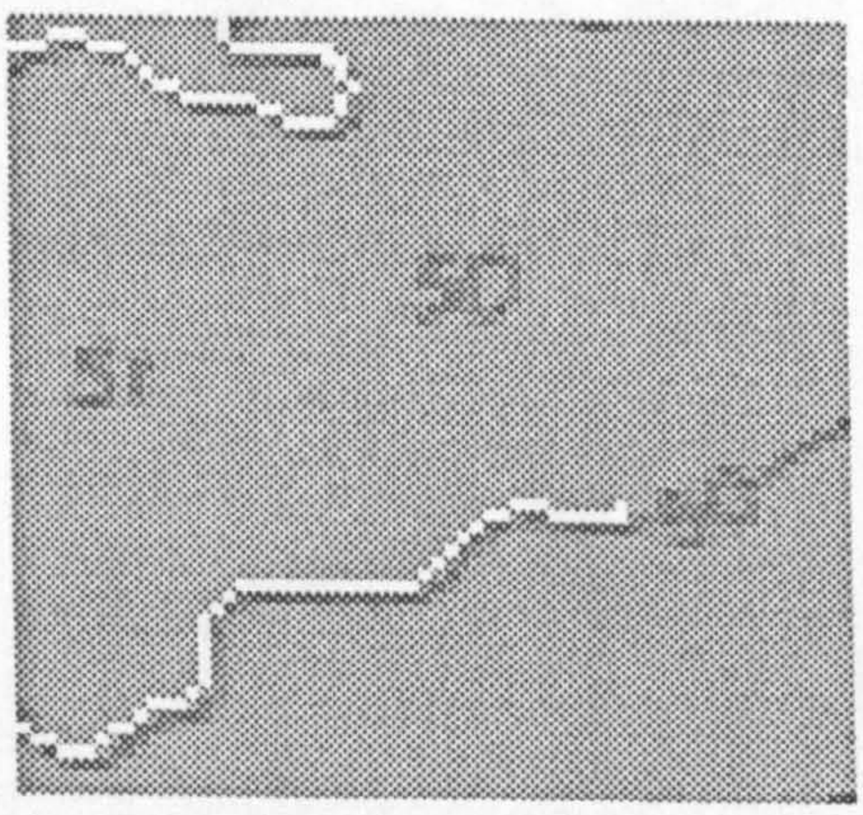
One of the segments found incorporates part of a line belonging to a depth marking. The user can split the segment and select which portion to retain within the feature. The discarded part of this segment is once again available for inclusion in later features.

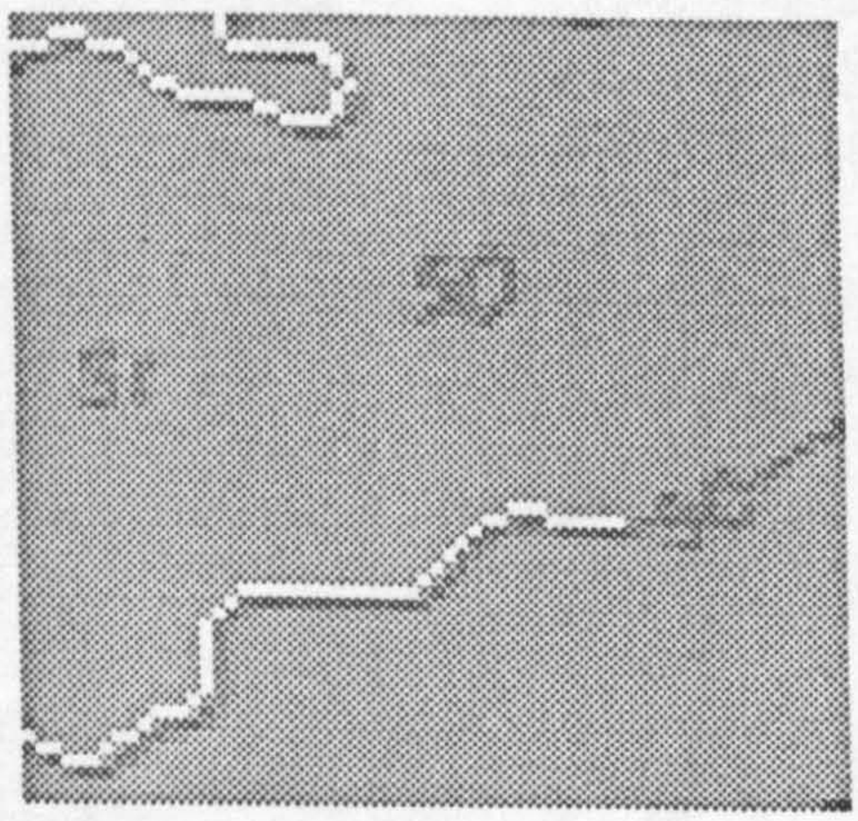


4.3.8 Eroding a Segment

The thickening of one of the segments is characteristic of a deficiency in thresholding in the line tracking process. An *erode* facility has been incorporated into the system to deal with this situation. The end of the segment to be eroded is located with the cursor, then for each carriage return typed by the user a single pixel is eroded from the segment until the segment appears to be correct. A delete facility is also offered for the occasion when the system might automatically select a wrong segment.

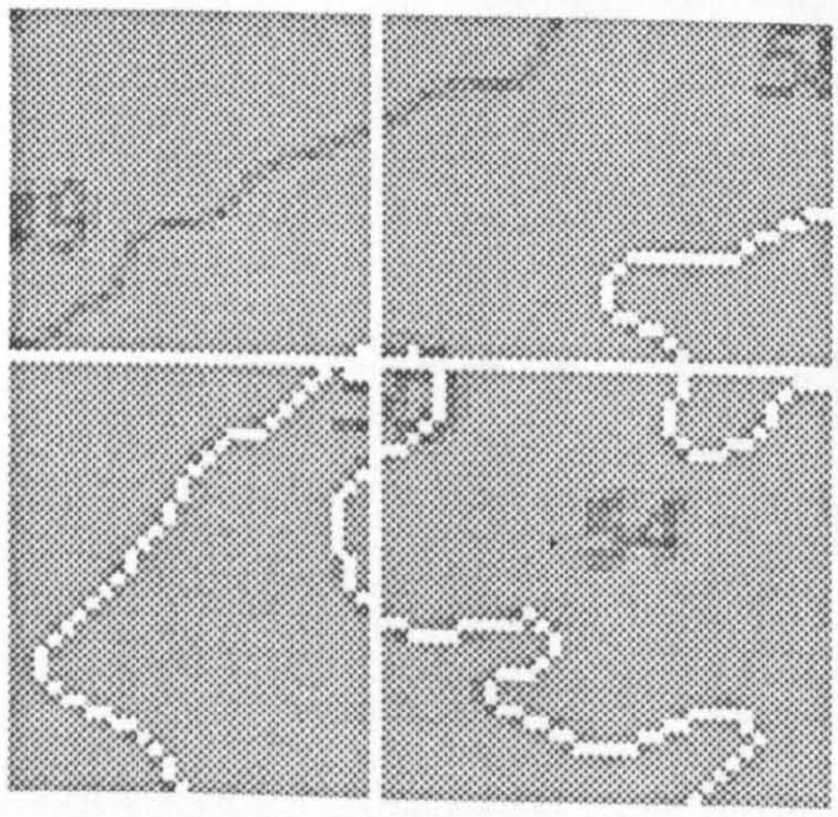


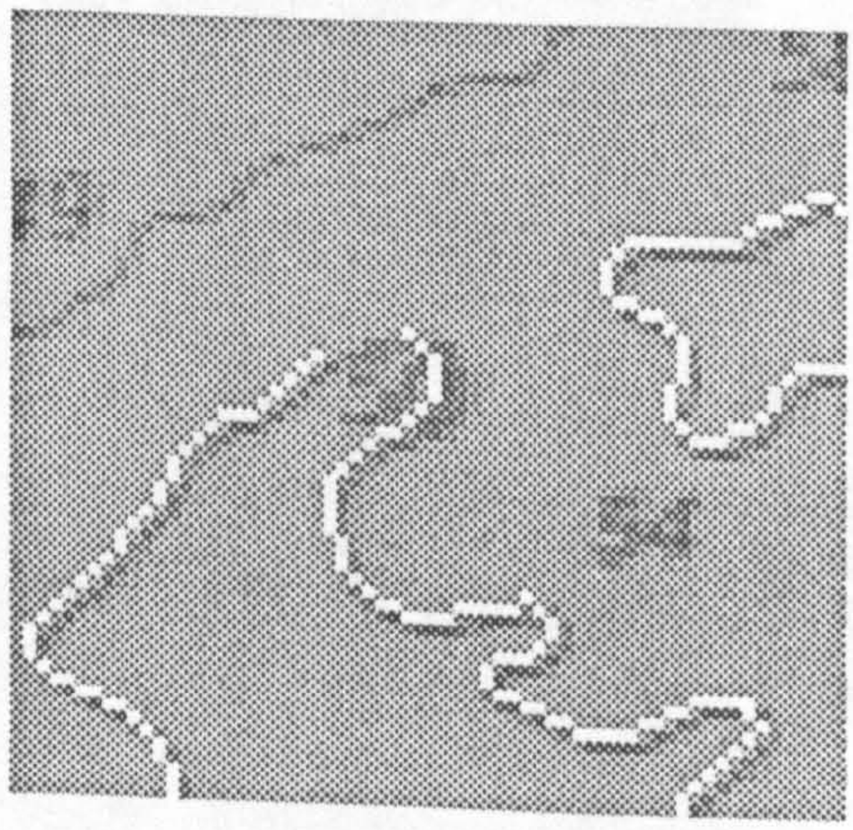




4.3.9 Removing a Segment

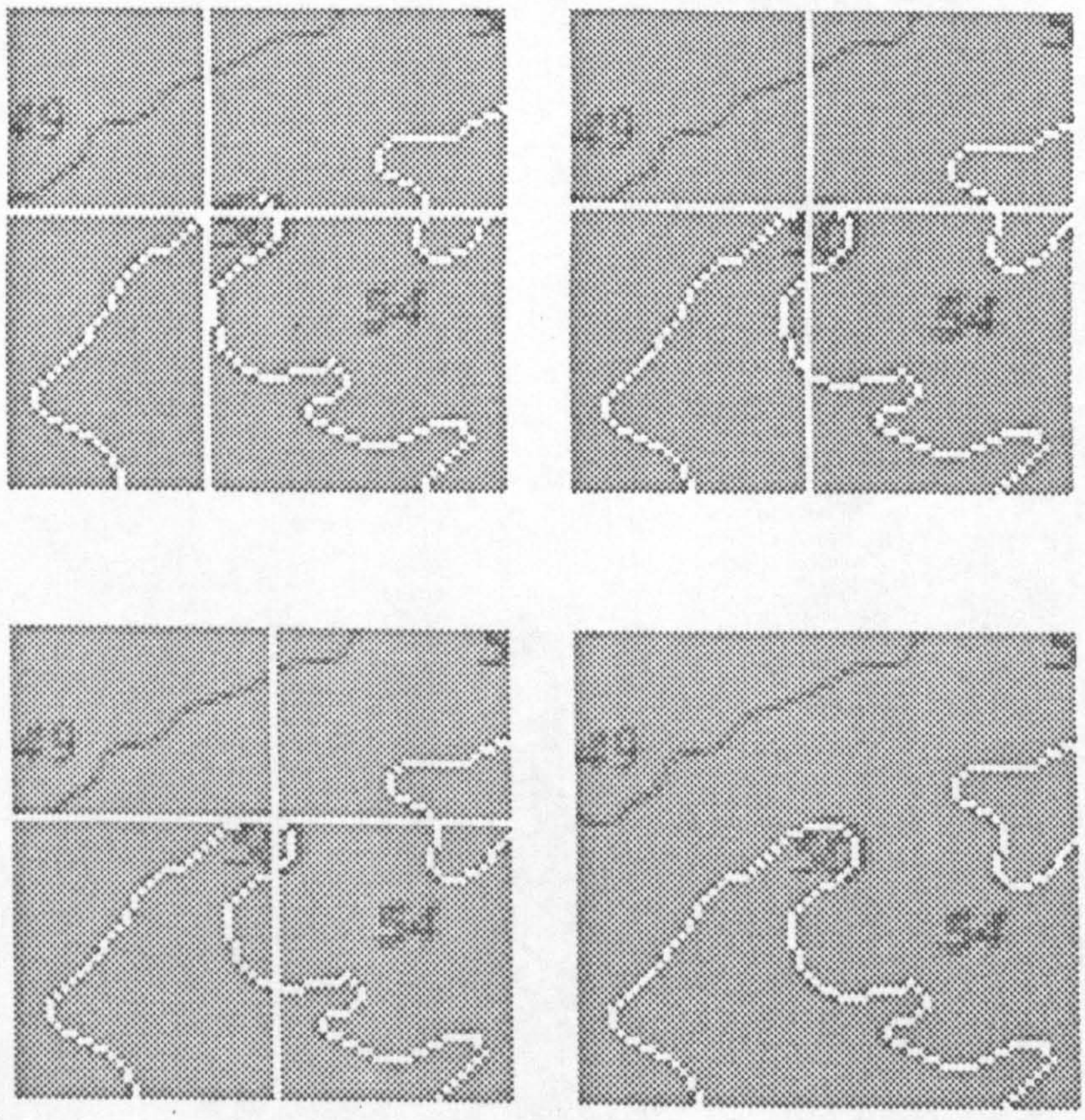
The spurious segment which has automatically been added can be removed using the delete option. The deleted segment is once again available for inclusion, manipulation at a later stage.





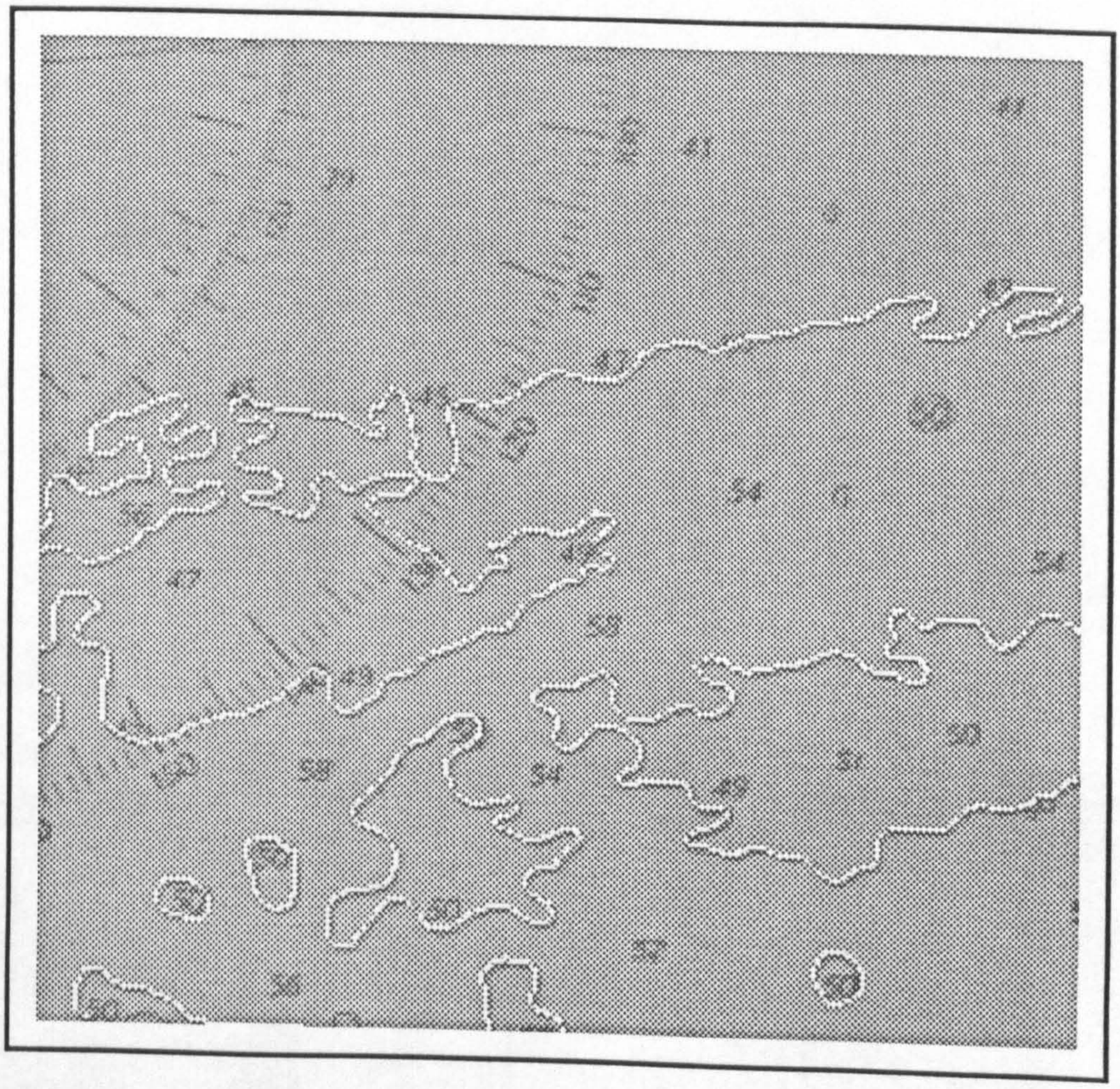
4.3.9 Creating an Artificial Segment

The gap between segments can be bridged using the *add* option which allows a piecewise generation of a new segment using a cross-wire cursor. The new segment is finished when the same cursor position is selected twice.



4.3.10 The Completed Feature Interpretation

Feature building continues until all desired features have been extracted.



4.4 Discussion of the SAFE system

The aim of this section of the research has been to develop a semi-automated feature extraction system which, although initially reliant upon a good deal of user interaction, is able to extract features reliably from line drawing images and could eventually be interfaced to a cooperating intelligent knowledge based system. The system described above fulfills these aims. Its development has necessitated the creation of a large software package comprising algorithms for creating and editing representations of linear features, and interfacing with the user through the image display and operator's console.

The limitations of the development environment described in section 3.2.2 made the creation of this package a significant task. The program code of the system has been written in as transportable a fashion as is possible, as it was envisaged that the processing power of the system upon which SAFE was developed would not be adequate for its comfortable usage. However, the system performs all the tasks that were initially stated to be required of it.

In chart areas where the chain coded representations of linear features are uncluttered and almost contiguous the user is rarely called upon to interact with the system. However, when chart features are interrupted, by for example depth markings or compass roses, user interaction is generally required for most feature extensions.

Occasionally the system will leave the true path of a feature under consideration and automatically extend it in the wrong direction. This may be because the line tracker has produced a single segment to which two separate chart features contribute, or because SAFE wrongly considers an extension of a feature to be unambiguous when in fact it is incorrect. When such an error

occurs it is easily rectified. However, the user must wait for automatic extension to cease. The creation of an *interrupt* facility within the development environment was investigated but could not be achieved without expending a considerable effort in programming at machine language level.

In ambiguous situations the user is asked to supply a *yes or no* response to prompts generated by SAFE. These situations are considered to be the points at which an IKBS could offer most assistance. This simple interface, or an extension of it to allow measures of uncertainty in the response, provides an elegant method by which SAFE can deal with interaction with the user and the IKBS in similar ways.

Chapter 5

Implementation of a Cooperating Intelligent Knowledge Based System

The remaining goal of the research was to increase the sophistication of the interpretation system in order to reduce the need for user interaction.

In section 4.1 it was argued that the best approach to implementing a sophisticated interpretation system was to augment the Semi-automated system with a *cooperating intelligent knowledge based system*. This chapter describes the design and implementation of such an *IKBS*, in the format of a production rule based system. The knowledge representation scheme and the mechanism by which the *IKBS* and the *SAFE* system interact are described in detail and some preliminary experience of the combined system is presented.

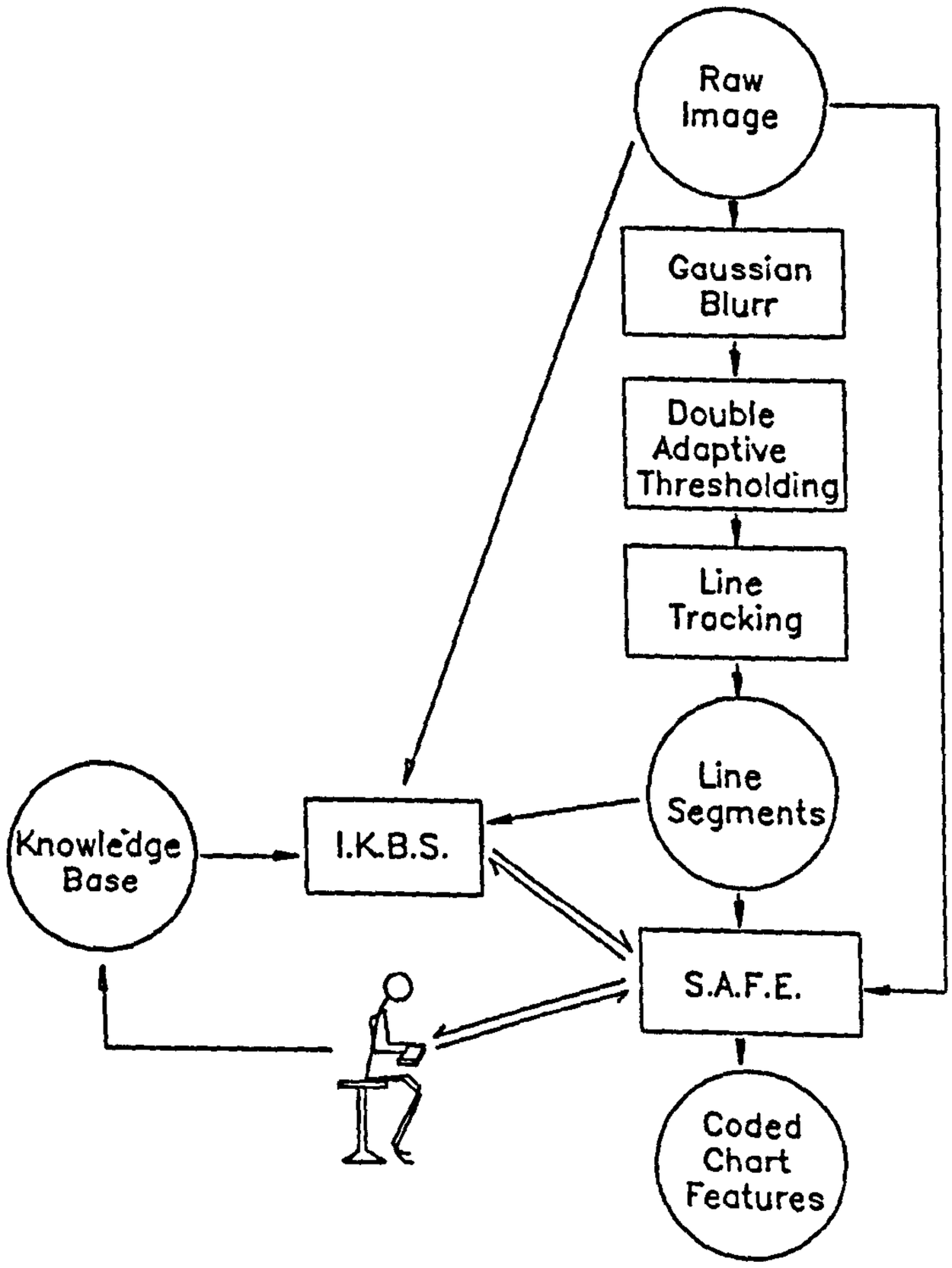


Figure 5.1: Schematic Diagram of the Design of the Cooperating System

5.1 Design Aims

The design of the combined system should maintain a clear separation between *SAFE* and the *IKBS*. The interface should be simple and elegant and preserve the independence of the *SAFE* system from the *IKBS*. It is implicit in the definition of a production rule based system that the domain knowledge is separate from the inference mechanism. A clear division would therefore exist between the *SAFE*, the inference engine of the *IKBS* and the additional domain knowledge (see figure 5.1). This was seen to be a highly desirable aspect of the design structure.

Conventionally, expert systems are consultative systems in which the user effectively has a conversation with the expert system at the keyboard. In the approach developed here, the feature extraction system consults the *IKBS* automatically by passing it goals to be satisfied. Provision must be made for allowing the combined system to distinguish e.g. between the suitability of a number of candidate feature extensions. This can be achieved by using a scheme for representing uncertainty in the *IKBS's* reasoning processes. The *SAFE* system therefore receives a certainty factor from the IKBS for each goal submitted. These certainty factors provide the extra decision making capabilities of the *SAFE* system.

5.2 Knowledge Representation

5.2.1 Possible Schemes

The options available for the knowledge representation scheme were to use :-

- a formal logic approach (e.g using the predicate calculus or *Prolog* programming language (Clocksin and Mellish [1984]))
- a semantic net representation
- a production rule based system using *triplets* as the atomic knowledge representation structures

Predicate Calculus Type Representation

The main scheme currently available for implementing predicate calculus is the Prolog programming language. This option was investigated but was found to be unsuitable because there was no version of Prolog available at that

time which could run on the development environment in use and interface to procedural code.

Semantic Nets

The type of knowledge to be represented appeared to map more cleanly on to independent rules rather than on to a network of related knowledge and a route to the implementation of searching and matching algorithms for semantic nets was unclear.

Production Rules and Triplets

This scheme was adopted because it is a well established method of knowledge representation and because it offers an intuitively appropriate method of transcribing the manner in which we formalise the type of knowledge to be represented, e.g. using rules of thumb.

5.2.2 The Chosen Knowledge Representation Scheme

A form of production rule was described in section 1.3.2 which breaks down each condition and action found in such rules into a set of three elements known as a triplet. For the problem considered here, it is necessary to consider more than just one entity and therefore the knowledge representation scheme must make provision for explicit reference to an entity or set of entities, in addition to the triplet, within each clause.

An atomic clause is therefore written in the form :-

Attribute, Predicate, Value, (Entity list)

for example

Length,Is Greater Than,10.0,(Line-A)

FeatureType,is,Contour,(LinFtr-A)

EndSeparation,is,high,(LinFtr-A,Line-A)

This representation scheme, whilst it considers the classification task at a very high level, is not a particularly elegant format for presenting information to a user of the system. In particular the attribute in this terse form is open to misinterpretation. The knowledge base is therefore augmented by a set of translations which describe in a more verbose form the characteristic represented by the attribute name. A single translation is held for each attribute which will be used whenever the inference mechanism finds the need to communicate with the user (see section 5.4.2). This separate data set also facilitates storing other information which is unique to the attribute but applies generally to any occurrence of the the attribute. An optional prompt may be defined for each attribute which allows the inference mechanism to solicit information from the user in order to define the state of a given goal. An example of a translation and prompt is given below.

Attribute Name	Translation	Prompt
FeatureType	the type of the feature is ..	Is the type of the feature .. ?

Atoms can exist in three different states according to their function within the inference mechanism. An atom may be a clause within a rule, a goal or a fact.

Facts

Facts are individual atomic clauses with associated certainty values indicating the certainty with which the fact is known. They are held in short term memory and are constantly updated and added to during the inference process. An example of a fact is :-

Line-A,FeatureType, is, contour CF 0.9.

Rules

Rules are supplied to the inference mechanism in the form

IF Conditions THEN Conclusions

where the conditions and actions are conjunctions of atoms. For example

IF End Separation, is less than, 10 units (Line-A,LineFeature-A)
AND End Orient'n Difference, is less than, 1 radian
(Line-A,LineFeature-A)
THEN Are Connected, is, true (Line-A,LineFeature-A) CF 0.8

Associated with each conclusion is a certainty value. This is a measure of the certainty with which the assertion may be made if the conditions are all known with absolute certainty. The certainty value lies in the interval -1.0 to 1.0 inclusive. A value of 1.0 implies that the fact is known to be true with absolute certainty. Similarly a value of -1.0 indicates that the fact is known to be false with absolute certainty. An intermediate value represents a measure of the certainty of the fact, with 0.0 representing a wholly unknown state.

A rule clause may also include certain directives which alter the manner in which they are used within the inference process. A condition clause may be preceded by the word *function.* This indicates that there is available a procedural function which is capable of producing an initial certainty for the state of a goal. Preceding an action clause with the word *procedure* indicates to the inference mechanism that this clause does not represent a conclusion which may be asserted into the data base of facts, but instead is a block of procedural code which should be executed whenever the accompanying condition set is satisfied.

Goals

A goal is an individual atomic clause which does not have a directly associated certainty value. For example,

AreConnected, is, true (LineFeature-A, Line-A)

essentially asks the question *"Is Line A connected to Line feature A ?"* A goal may be a member of a set of goals resulting from the invocation of a rule.

5.2.3 Content of the Knowledge Base

Long Term Knowledge

The long term knowledge base consists of all the rules supplied to the inference mechanism and a set of analogical models, i.e., all knowledge about charts and images that does not change from one application of the system to the next. Analogical models are implemented as conventional procedural computer program routines and perform calculations which allow estimation of an initial certainty factor for some goals. Results generated by such models are stored in the short term memory of the inference engine and are then

available for modification or use by propositional rules. Analogical models produce their inferences by having access to the data which represent object descriptions of the entities contained within the application domain.

Short Term Knowledge

The short term knowledge base consists of the facts pertaining to the current state of the domain, i.e. all facts known about the chart image currently under consideration. Facts may be established through inferences using the propositional models contained within the structure of a rule or by using analogical models. Facts are volatile and do not exist at the beginning of a session with the *IKBS*.

5.3 The Inference Engine

The inference engine is an example of a backward chained inference mechanism. Goals, facts, and rules are maintained in stack data structures. Figure 5.8 on page 138 describes in pseudocode the overall control mechanism of the inference engine. A main goal is placed at the bottom of the goal stack and is tested against the conclusion sets of rules, each rule found that matches the goal is fired immediately. The result of firing a rule is that the condition set of the rule is placed on top of the goal stack and the new top goal is then tested against the rule base. Hence conflict resolution is achieved via a priori ordering of the rule set.

5.4 The Man - Machine Interface

5.4.1 Preparatory Aspects

Prior to invoking the inference mechanism it is necessary to define the domain knowledge which it will use to perform inferences. All preparation is carried out with the aid of a standard text editor. There are two elements of preparation which must be carried out. The first is to define the rules, along with the information associated with individual attributes. The second is to provide system initialisation information. This information indicates to the inference mechanism the location of the rule files to be consulted and sets parameters which alter certain aspects of the control structure characteristics.

A grammar has been defined for the supply of information to the system which while not particularly obtrusive has reduced the heady and somewhat irrelevant task of parsing the rule clauses. The grammatical structure is similar to that found in many text processing systems. It is based on the placing of *dot commands* within the text in order to define the nature of the data structure which the ensuing text represents. The dot commands are always the first text elements on a line. Certain other elements of punctuation assist in the interpretation of atomic clauses by making explicit the division between the individual elements of the clause. These have already been illustrated in the previous examples of atomic clauses. To consider first the preparation of a text file which contains rules, the highest level of division of the elements lies in defining the extent of rule and attribute definitions. This division is performed by the dot commands illustrated in table 5.1.

Rules must contain one set of conditions and one set of actions which are introduced by the dot commands .CS and .AS respectively. The counterparts of these commands which terminate the sets of clauses are .CE and .AE. Each

.RS	Start of a rule
.RE	End of a rule
.AS	Start of attribute information
.AE	End of attribute information

Figure 5.2: Rule and Attribute Introducer Commands

condition clause is introduced by the command .CO and requires no termination, this being implicit in the construction of a complete clause. Similarly each action clause is introduced by .AC. The specification of a rule within the above constraints is of a relatively free format in that case has no meaning (i.e. all characters are converted to upper case to form the internal representation) and carriage returns and spaces may be used freely to clarify human interpretation. In addition all characters which follow the first '!' within a line are considered as comments and have no bearing on the internal representation of rules etc.. A rule is therefore given to the inference mechanism in the format shown in figure 5.3.

The definition of information associated with attributes follows a similar format and may be interspersed between the rules within a rule file. The complete data structure is introduced with the dot command .AS and terminated with .AE. Each element within the data field has a single introducer. These are; .AT for the attribute; .TR for the translation and .PR for the prompt. The elements may occur on the same line as the dot command or wrap on to following lines. The specification of a prompt is optional and its presence defines whether, in the event that the inference mechanism decides that it should refer to the user to gain information, user information can be solicited for this particular attribute.

An example of the definition of attribute information is shown in figure 5.4.

```
.RS
.CS
.CO End Separation, is less than, 5,(Line-A,LinFtr-A)
.CO End Orientation Diff, is less than, 1,(Line-A,LinFtr-A)
.CE
.AS
.AC Are Connected, is, true,(Line-A,LinFtr-A),0.8
.AE
.RE
```

where Line-A and LinFtr-A are variables which indicate the features concerned.

Figure 5.3: Format of Rule Specification

```
.AS
.AT EndSeparation
.TR the distance separating the ends of
.PR Is the distance separating the ends of
.AE
```

Figure 5.4: Format of Attribute Specification

```
.RF
Connected.Rul   !deals with connected ends of linear segments
ElemType        !deals with classification of feature types
.RE
```

Figure 5.5: Format of Specification of Rule Base Segmentation

The second aspect of preparation is the production of an initialisation file which must be given the name 'Infer.Ini'. This file contains the names of all rule files to be consulted. In this manner the rule base may easily be segmented. The dot command .RF introduces all rule file names and .RE indicates the end of the list. An example is illustrated in figure 5.5.

As previously mentioned, certain aspects of the control structure characteristics may also be altered from within the initialisation file which generally deal with the man machine interface properties of the system whilst it is in operation. These will be detailed in the next section.

5.4.2 Operational aspects

It is important that any communication between the inference engine and the user of the system should be easily understandable and unambiguous. When dealing with high level descriptions in a textual form the scope for ambiguity is large. The inference mechanism developed for this project is not primarily a conversational system, that is to say that its reasoning processes do not rely on user supplied information. Indeed it is intended that, in its fully operational mode, user interaction with the inference mechanism would be kept to a bare minimum. The IKBS should ideally interact solely with another piece of software, the semi-automatic feature extraction system. However, the

provision of a facility within the system to explain the reasoning processes which are carried out is important. This facility allows checking of the control structure and rule semantics in addition to offering a tutorial aspect to the system. It is also useful during the development and testing of new rules to be able to allow the system to function in a conversational manner. For example it may be desired to test a rule which is intended to invoke an analogical model before the model has been defined computationally. If the system is able to solicit interaction when no knowledge can be brought to bear on the satisfaction of selected goals then this provides an elegant development environment.

The explanation and interaction features of the system developed here are controlled from non-volatile system parameters which are included in the initialisation file. These parameters may be changed using a text editor and define the following characteristics.

.DF *state* A translated version of each rule invocation is displayed

.DA *state* A translated version of each assertion made by the system is displayed

.UH *state* User interaction is solicited when no information on the state of the current goal can be derived

where *state* is either *on* or *off*.

5.5 The Rule Base Source

Much emphasis is placed on the formulation of rule bases for the development of *expert systems* within the artificial intelligence research community. Expert systems are a particular class of IKBS which attempt to model human

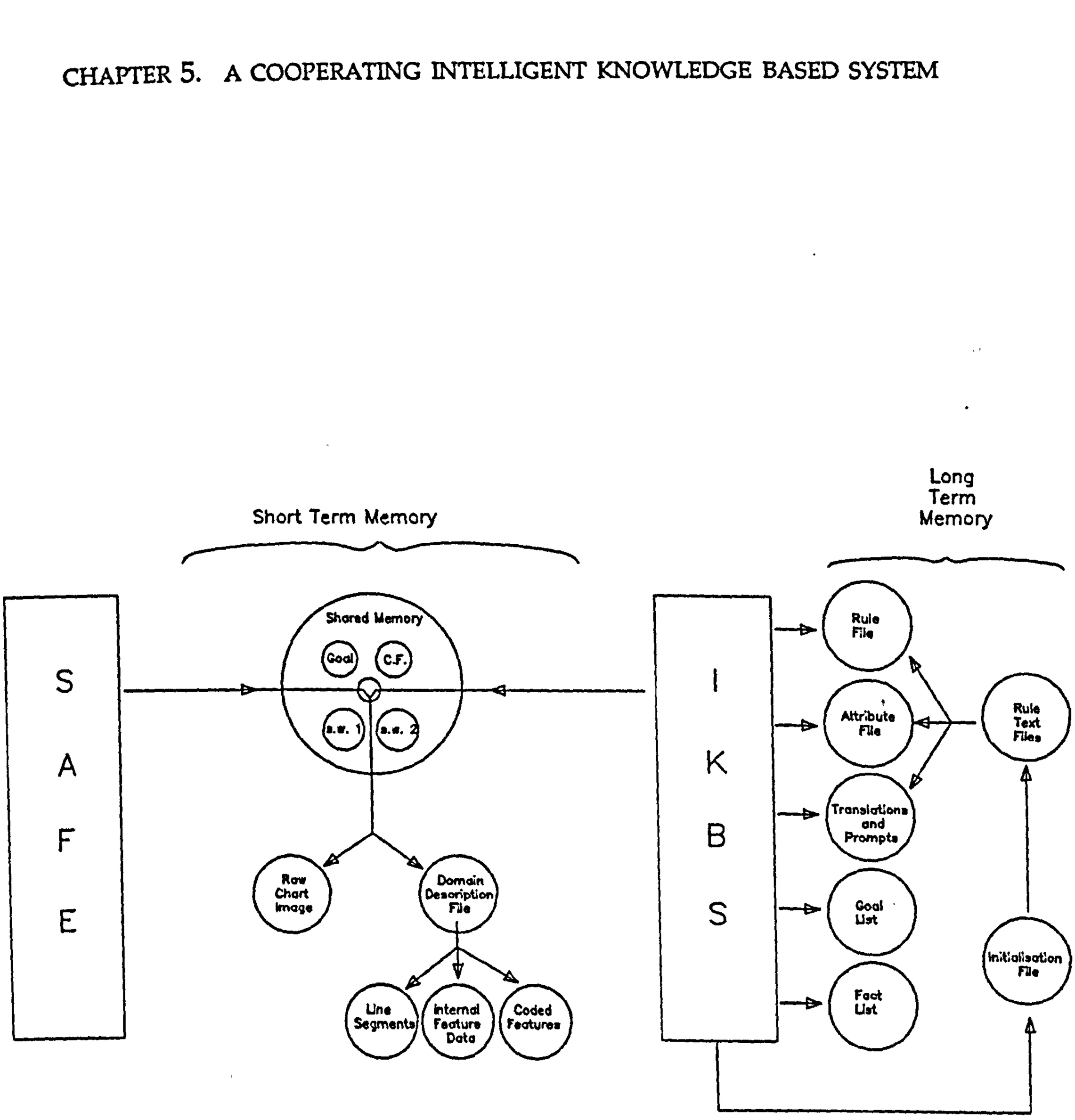


Figure 5.6: The Interface between SAFE and the IKBS

expertise. The subject of knowledge engineering is a recognised discipline and profession. The reason for the importance of the study of structuring of knowledge is that expert knowledge is often ill defined even within the mind of the expert. An expert within a given discipline may deduce information for which it is difficult to explain the line of reasoning which allowed the deduction to be made. To automate the deduction process a working approximation to this line of reasoning must be obtained. A knowledge engineer attempts to elicit and structure the expert knowledge by directed discourse with the expert. Indeed, without this knowledge elicitation process an expert system can be considered to lack credibility.

Here, we are not trying to create an *expert* system in the conventional sense, but rather a system which uses *everyday* human knowledge. Within the framework of the overall system for image interpretation developed here it is possible to code the domain knowledge, which might very well be implicitly dispersed amongst the data structures and instruction sequences of a procedural system, into the coherent form of production rules. The type of knowledge within these rules is analogous therefore to that embodied within procedural systems, for which the justification for its presence is its effectiveness. It is therefore proposed that a structured knowledge elicitation process involving experts for the development of the system considered here is unnecessary. One of the advantages of using knowledge based programming techniques in the manner described here is that the sophistication of the system is easily increased by the end user alone, who would derive models based on the experience of interacting with the system which can be translated into rules. A limitation of this approach is that some of the condition clauses of rules generated in this fashion will refer to low level properties of the image segments which must be derived from procedures. If a new property measure

is required then a procedure must be written to evaluate that measure, and must be incorporated into the program code of the IKBS.

5.6 Data structures and Algorithms of the IKBS

The fundamental characteristics of the inference mechanism developed here are that it is a goal directed system which allows the aspect of uncertainty to be considered. This section details all aspects of the nature and implementation of the inference mechanism.

5.6.1 List Data Structures

Prior to discussing the algorithms which perform the inference operations a descriptions of the data structure of the various lists will now be given.

The Fact List

The list of facts (see figure 5.7) is the simplest data structure in that it is a sequential list. The top item on the list is accessed by a stack pointer. Each item contains a pointer to the next item down in the list. As each fact relates directly to the domain in question, it must have reference to a full set of entities and an associated certainty factor. The entities are held in the entity list, which is a list of pointers to variables in the domain description file (see figure 5.6).

The Rule List

The internal representations of the rules are held in secondary storage as a direct access sequential file. The location of the first condition and action clause of each rule is stored within an array subscripted by the rule number. Currently the number of elements in this array, and hence the maximum

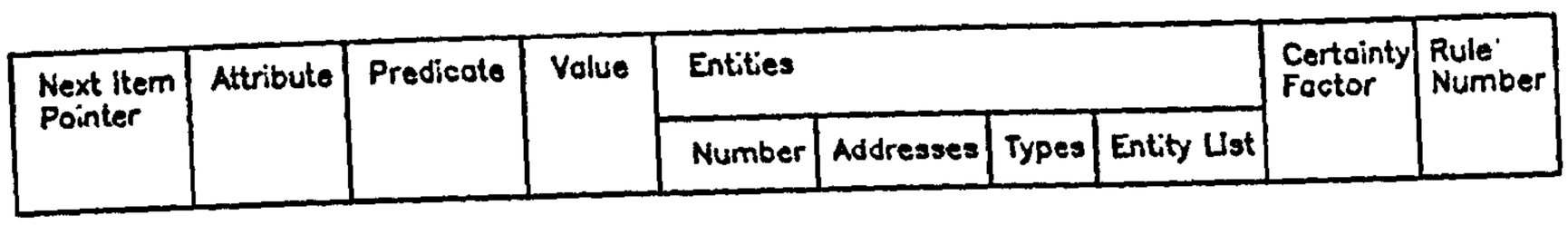


Figure 5.7: The Fact List data structure

number of rules is 50. This figure can easily be increased as the rule base grows. The locations of these first clauses are specified by the record number of the clause within the direct access file. Further clauses within each of the sets are located by reference to a *next item* pointer, which is zero for the last clause in the set. Each clause contains a field which specifies the number of variables that it refers to, which is followed by a list, of this number of elements, containing the addresses and types of the variables. In addition each clause contains fields which specify the rule number to which it belongs, the clause type (condition / action) and a flag which indicates whether there is a function or procedure associated with this clause. Rule action clauses also contain an associated certainty factor.

The Attribute List

The attribute list (see figure 5.9) is maintained in two files. The first is a sequential file, indexed by the attribute name. Associated with each attribute are 2 pointers - one to the translation of the attribute and one to an optional

Rule Item Elements

Next Item Pointer	Attribute	Predicate	Value	Entities			* Certainty Factor	Rule Number	Condition / Action Flag	Available Procedure Flag
				Number	Addresses	Types				

* Used only in action clauses.

Rule List Structure

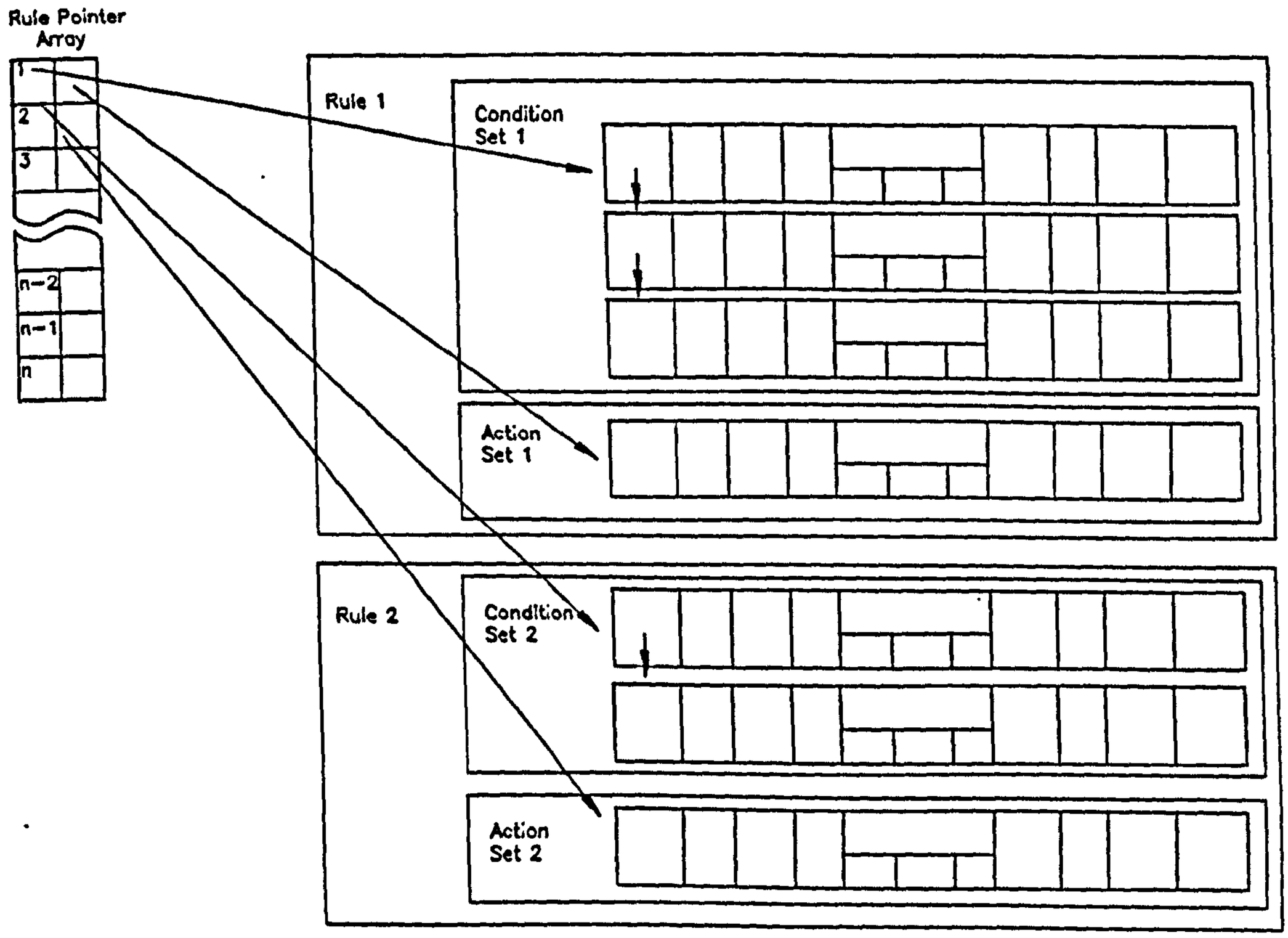


Figure 5.8: The Rule List data structure

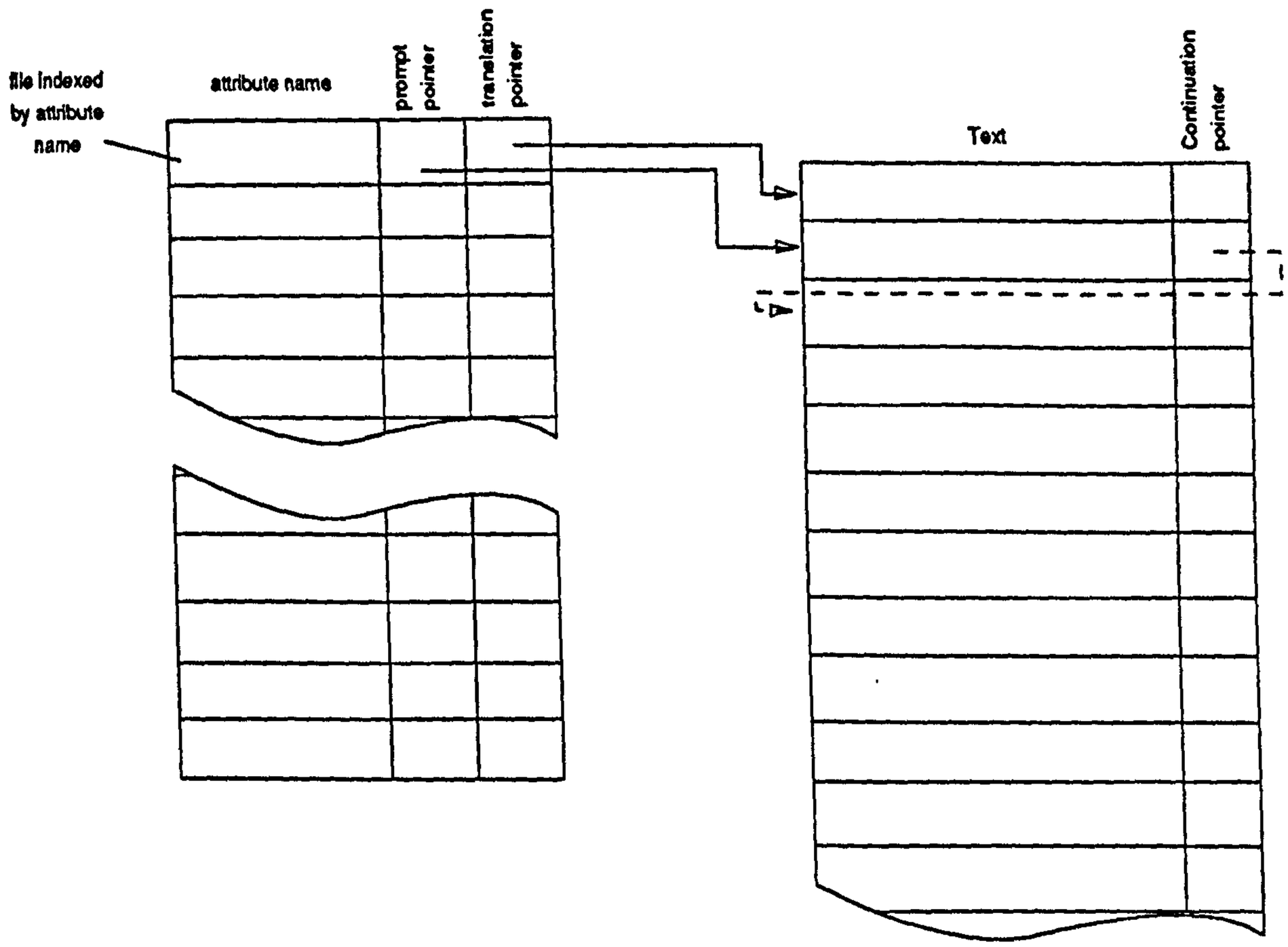


Figure 5.9: The Attribute List data structure

prompt. These pointers point to record numbers in the second file, which contains the text for both translations and prompts. Each text string can be of indefinite length, as a linked list structure is maintained for each translation or prompt.

The Goal List

The goal stack (see figure 5.10) is the most complex of the list data structures. Within it, various pointers maintain list structures and sequencing of these lists. A goal stack pointer points to the current goal under consideration.

The goals which make up an individual subset are connected by both forward and backward pointers. Each goal of the subset contains a pointer to the *super-goal* which caused invocation of the subset. The goal item contained within the dashed line box in figure 5.10 illustrates the concept of a shadow goal. It is the maintenance of shadow goals that imposes the requirement for keeping goals in forward and backward connected subset lists. Goals are only removed from the stack as complete subsets. This allows the capacity for *backtracking* described in section 5.6.5.

An additional pointer for each goal is maintained which provides a cross reference to a corresponding fact to the given goal on the fact list.

A description of the entities in a goal item includes an additional list of the *instantiation states* of each of the entities. When a goal is set up with a variable for which only the *type* is specified then the variable is said to be *uninstantiated*. In practical terms, the goal's variable, which is in fact a pointer to a data structure held elsewhere, is set to zero and the value of the corresponding instantiation state parameter is set to reflect this state. For example a goal with an uninstantiated line segment might effectively ask " Is there a line segment to which this feature is connected ?".

Goal Item Elements

Next Goal Set Pointer / Last Item Pointer / Next Item Pointer	Attribute	Predicate	Value	Entities					Rule Number	Context Stack Cross Reference Pointer	Certainty Factor	Available Procedure Flag	Last Rule Marker
				Number	Addresses	Types	Instantiation State	Entity List					

Goal List Structure

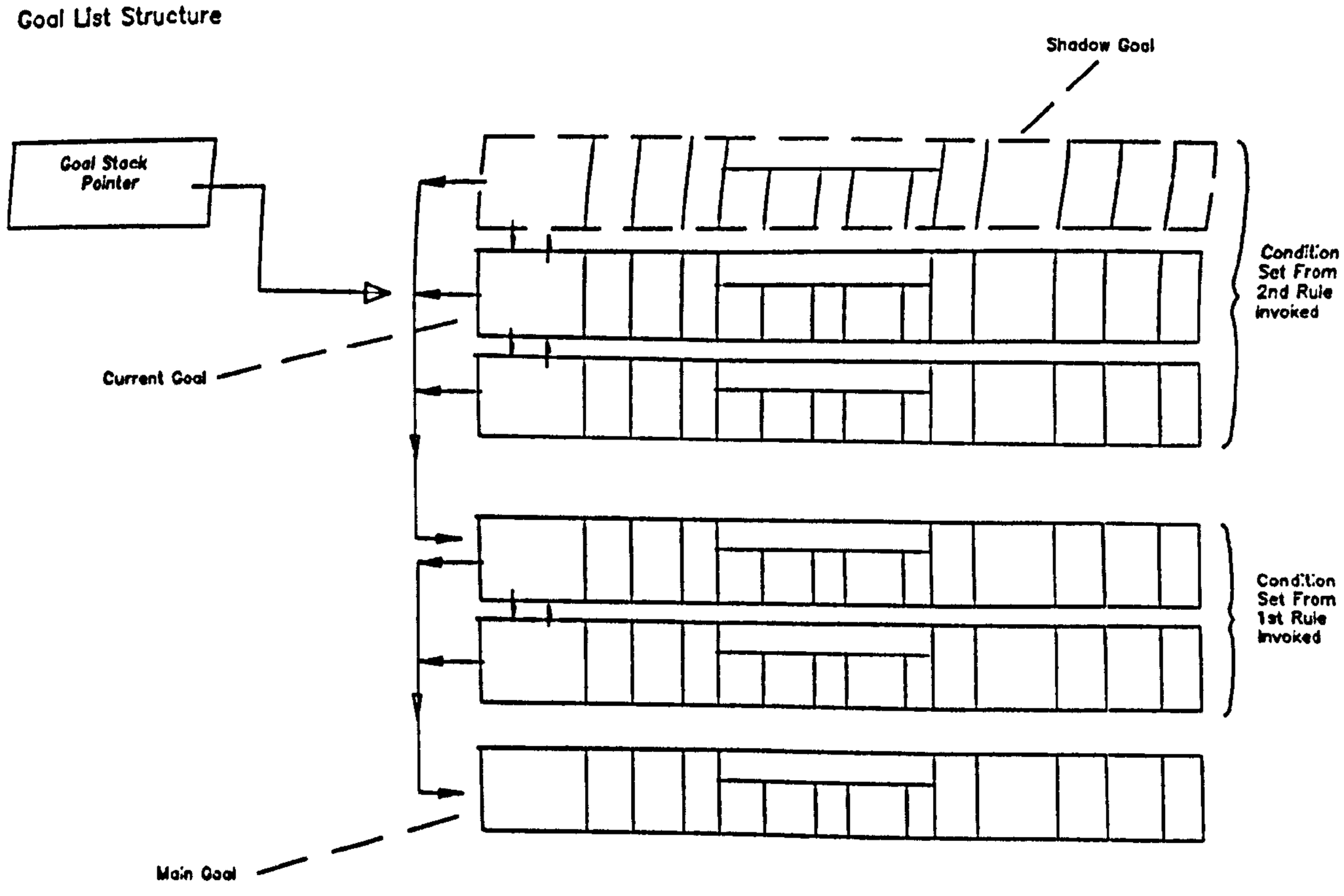


Figure 5.10: The Goal List data structure

```
Initialise inference engine;
REPEAT
   Monitor semaphore signals from application routine (Signal);
   CASE Signal OF
      New_main_goal : BEGIN
                         Perform inference process;
                         Report findings to application routine;
                      END;
      Acknowledge   : Return acknowledgement signal to application routine;
   END;
UNTIL Signal = Terminate;
Close down inference engine;
```

Figure 5.11: The Overall Control Structure

5.6.2 Algorithms

Figures 5.11 to 5.14 are a pseudo-code version of the main control process which provides a basis for the explanation. The pseudo-code reflects a non-recursive implementation structure. The control structure of inference mechanisms is generally well suited to the use of recursion, however this property was not available within the programming language used for this research.

5.6.3 The IKBS Initialisation Process

Part of the initialisation process referred to in figure 5.11 is the consultation of the user's rule files. The system reads from the initialisation file the names of all the rule files that should be consulted. As each rule is added to the rule base it is numbered sequentially, from 1 upwards. Those elements which are defined explicitly within the rule base text are mapped directly to the internal representation by a parsing algorithm. Each of the entities within a given rule is given a unique address local to that rule, in order that the integrity of the clauses within the rule is maintained.

```
BEGIN
   Get main goal;
Put main goal as first goal on goal stack;

WHILE NOT Inference process complete DO
    BEGIN
       Initialise top goal;
       Certainty := Current certainty of top goal;
       IF Certainty < 1.0 THEN
          BEGIN
             IF Can find a rule to help with top goal THEN
                Put condition set of this rule on top of goal stack
             ELSE
                BEGIN
                   IF User help required AND allowed AND
                      Can find a prompt associated with the current goal THEN
                         Solicit user information;
                   IF Certainty implies top goal is Known THEN
                      Consider the next goal down on the stack;
                   ELSE
                      BEGIN
                         IF Backtracking succeeds to reinitialise a
                                                goal in the current set THEN
                            Let this reinitialised goal be the new top goal
                         ELSE
                            Remove the current top goal set;
                      END;
                END;
       END
    ELSE
       Consider the next goal down on the stack;
END;
END;
```

Figure 5.12: The Body of the Inference Mechanism

```
BEGIN
   Display the prompt;
   Get the user's response;
   Modify the context item state according to the user's response;
END;
```

Figure 5.13: Soliciting User information

```
IF Top goal is the main goal THEN
   Flag inference process complete
ELSE
   BEGIN
      IF top goal is last in current set THEN
         BEGIN
            Assert action set of rule which created current goal set;
            Remove current goal set from goal stack;
            Let goal which invoked the previous goal set be the new top goal;
         END
      ELSE
         Let next goal down in current set be the new top goal;
END;
```

Figure 5.14: Moving to the Next Goal

5.6.4 Communication between SAFE and the IKBS

The IKBS and SAFE are implemented as two separate computer programs. They communicate via a common region in computer memory (see figure 5.6). This region holds pointers to shared data structures, such as the name of the chain code file for segment input etc.. This region also contains space for goals supplied by the SAFE system for the IKBS to attempt to satisfy. In addition, two *status words*, one for each program, allow a semaphore like communication between the two programs.

To pass a goal to the IKBS, SAFE places a knowledge triplet and entity list, in an agreed internal format, into the common region. This information forms the main goal for the IKBS. Having processed this main goal the IKBS will return a certainty factor in a similar fashion.

Only one of the programs performs useful processing at a given time. When SAFE has generated a goal and placed it in the common region of memory it indicates to the IKBS, by changing its status word value, that it wishes for a new goal to be satisfied. SAFE idles until the IKBS indicates that processing of the goal is complete, by using its status word.

SAFE can issue a request for the IKBS to perform an acknowledgement process. This allows SAFE to know whether it is functioning with or without the aid of the IKBS. If SAFE does not receive the acknowledgement from the IKBS within a small time interval (0.1 seconds) then it assumes that it is functioning unaided.

5.6.5 Goal Satisfaction

Figure 5.12 illustrates the main aspect of the inference process. When a main goal is received by the IKBS it is placed as the first item on the goal stack. The inference process then commences satisfying this goal. During this process the

goal stack grows by invocation of rules and shrinks by proving or disproving goals. The inference process is complete when one of the following conditions is met :-

- the main goal is found on the fact list before any rules have been applied;
- the main goal is proved with a certainty factor of 1.0;
- all appropriate rules have been applied to the main goal.

Goals are satisfied in a *depth first* fashion. That is to say that, in attempting to satisfy a goal, rules are sought, the top condition clause of which becomes the new current goal. The original goal will be returned to when all the conditions of the new condition set have been dealt with.

Initialising a Goal

At each analysis the current goal is initialised. The following paragraphs explain each aspect of the initialisation process.

Instantiation If any entities of the current goal are marked as being uninstantiated then the initialisation process will instantiate them to some instance of an entity of the appropriate type. Instantiation of an individual entity may be performed many times during the complete inference process in order that a rule may refer to something that has not been defined in the goal supplied by SAFE. Each reinstantiation must select a different entity to the last. Instantiation is therefore a process of searching for a variable of the appropriate type in the domain description file, beginning the search at the next variable after the one to which the entity had previously been instantiated. The entity is marked as *instantiated but encountered uninstantiated*. The *Last rule marker* of this goal is set to zero to indicate that it has not yet been processed in its new state.

First Encounter With A Goal If the initialisation process finds that the *Last rule marker* is zero then it searches for a corresponding item on the fact list to see if this goal has previously been satisfied (either during satisfaction of the current main goal or any previous main goals of the session). If the corresponding item is found then a cross reference pointer is established from the goal to the fact and the goals last rule marker is set to the highest rule number. This indicates that no further processing of the goal is required.

If the goal has not already been proved then a corresponding fact item is established on the fact list with a certainty factor of 0.0 and a cross reference pointer to this fact is created. The *available procedure flag* of the goal is then checked to see if it is set. If this is so, then a procedure, indicated by reference to the attribute name is executed to obtain a first value of the certainty factor for the goal. The goal's last rule marker is set to -1 in order to indicate that a function has been used.

Searching for rules

If the goal requires further satisfaction, i.e. its certainty factor is less than 1.0, then production rules are sought. An appropriate rule is found by matching the goal to an action clause of a rule. This is the essence of a backward chained production rule based inference mechanism. The attribute, predicate, value, number of entities and the types of the entities must all match for a rule to be selected. Searching begins at the rule number one greater than the goal's last rule marker, or at rule 1 if the marker is less than zero.

When a rule is found then the condition set of the rule is placed on the goal stack. The entities from the goal which invoked the rule are mapped from that goal to the new goal set. Processing of the new top goal then continues.

If no rule is found, or when the goal is proved with a certainty of 1.0,

then the processing of the goal is complete. For certainty factors less than 1.0 further rules are sought until one of the terminating conditions is met.

Soliciting User Interaction

If no estimate of the certainty of a goal has been possible then the IKBS can solicit user interaction if :-

- the system parameter which controls user interaction is set to allow interaction
- a prompt has been defined to allow the IKBS to formulate a question.

If these criteria are satisfied the user will respond to the prompt with a certainty factor. Usually some estimate of the certainty will have been established and this feature of soliciting user interaction is designed for use during development stages of the system. The low level questions which might be asked of the user would serve as an annoyance. Interaction during routine usage of the system would be only through SAFE, requiring the user to satisfy only high level goals.

Assessing the Usefulness of Satisfied Goals

When some estimate of the certainty has been found then the certainty factor is checked to see if the goal is *known*. A goal is known if its certainty is above 0.2. A value of 0.2 is used as a threshold because this tends to indicate that there is non-negligible evidence that the goal is true. This threshold is clearly adjustable and might be modified after gaining more experience with the system.

A good indication of the condition being false does not allow assertions to the contrary of a rules action clauses. This would be a contravention of the *Modus Ponens* rule (see section 1.3.2). This rule states that :-

IF P then Q, P therefore Q

but

IF P then Q, not P therefore not Q

does not hold.

Therefore, if there is evidence that a condition of a rule is untrue then *no* assertions may be drawn.

If the goal is known then the next goal down in the condition set, if another exists, may be considered.

Making Assertions

When the last goal in a set has been shown to be *known* then assertions may be made, using the action clauses of the rule from which the set arose. One of these action clauses will refer to the goal which invoked the rule.

The certainty of a fact is modified using the minimum certainty from the goal set, CF_{min}, and the action clause's certainty factor, CF_{act}. The new certainty, CF_{new}, is computed using the system developed for the well known expert system, *Mycin* (Alty and Coombs [1984]).

If the original certainty factor CF_{orig} of a fact is 0.0 then the new certainty factor is given by equation 5.1.

$$CF_{new} = CF_{min} \times CF_{act} \tag{5.1}$$

For a fact which already exists with a non-zero certainty factor a modified certainty CF_{mod} is computed by modifying the CF_{new} of equation 5.1 using equation 5.2.

$$CF_{mod} = \begin{cases} CF_{orig} + CF_{new}\,(1 - CF_{orig}) & \left| \begin{array}{ll} CF_{new}, CF_{orig} & > 0.0 \\ CF_{new}, CF_{orig} & < 0.0 \\ CF_{new} \times CF_{orig} & < 0.0 \end{array} \right. \\ -\left(|CF_{orig}| + |CF_{new}|\,(1 - |CF_{orig}|)\right) & \\ \frac{CF_{orig}+CF_{new}}{1-min(|CF_{orig}|,|CF_{new}|)} & \end{cases} \tag{5.2}$$

Backtracking

When a goal is not *known,* at completion of processing it, then a process of backtracking occurs. Each of the previous goals in the goal set are checked in reverse order to see if there are any entities which were encountered uninstantiated. If such an entity is found then it is marked as uninstantiated, and this goal becomes the new *current goal.* This has the effect that, when the goal is next initialised, the uninstantiated entity will be reinstantiated using the process mentioned earlier in this section. If no such entities are found then the whole goal set is removed from the goal stack and the supergoal which invoked that goal set becomes the current goal once more.

Completion of Satisfying the Main Goal

The inference process is complete when no more rules can be found to apply to it. The certainty factor of the main goal is placed in the common region in memory and the IKBS signals to SAFE that the inference process is complete. The IKBS then returns to monitoring the signals from SAFE.

Using the Certainties Generated by the IKBS

SAFE must have some mechanism for interpreting the certainties that the IKBS returns. For the case when only one main goal is issued for the IKBS to resolve, then SAFE must decide whether the certainty is sufficiently high to assume that the hypothesis is correct. SAFE will often issue a series of hypotheses and store the certainties for later comparison. An example of this situation is when a number of candidate feature extensions are found. In this case SAFE must decide whether there is a candidate with a sufficiently high certainty factor and whether the certainty factors of any of the others reintroduce the ambiguity. SAFE currently requires a single certainty factor above 0.8, with

no other factors above 0.7 in order to consider that an hypothesis is correct. Otherwise the IKBS has failed to resolve the ambiguity and user interaction is solicited.

User Functions

User functions are the analogical models which create the first approximations to the certainty of a goal. The program code representing user functions is placed in the code of the IKBS. During execution all functions are called from a single routine in the IKBS. This routine checks the attribute name of the goal which claims to have a corresponding user function, and if that function is found then it is invoked.

The input to a user function is the functional part of the goal. The function generates a certainty factor for the goal as its output. Establishing this certainty factor is a duty of the user function, however an initial general approach to this problem has been developed. The user sets up a mathematical function using a coordinate based system (see figure 5.15). This function, along with the predicate and value of the goal and the computed value for the attribute are passed to a mathematical procedure. This mathematical procedure recognises the predicates: equals; is less than; is greater than and is between. It computes the disparity between the predicate / value combination of the goal and the computed value of the attribute. A certainty factor corresponding to this disparity is then read from the mathematical function.

5.7 Experience with the Combined Interpretation System

The use of the SAFE system with the aid of the knowledge based system is described within this section. The characteristics of the user interface to the combined system during image interpretation can appear exactly as described

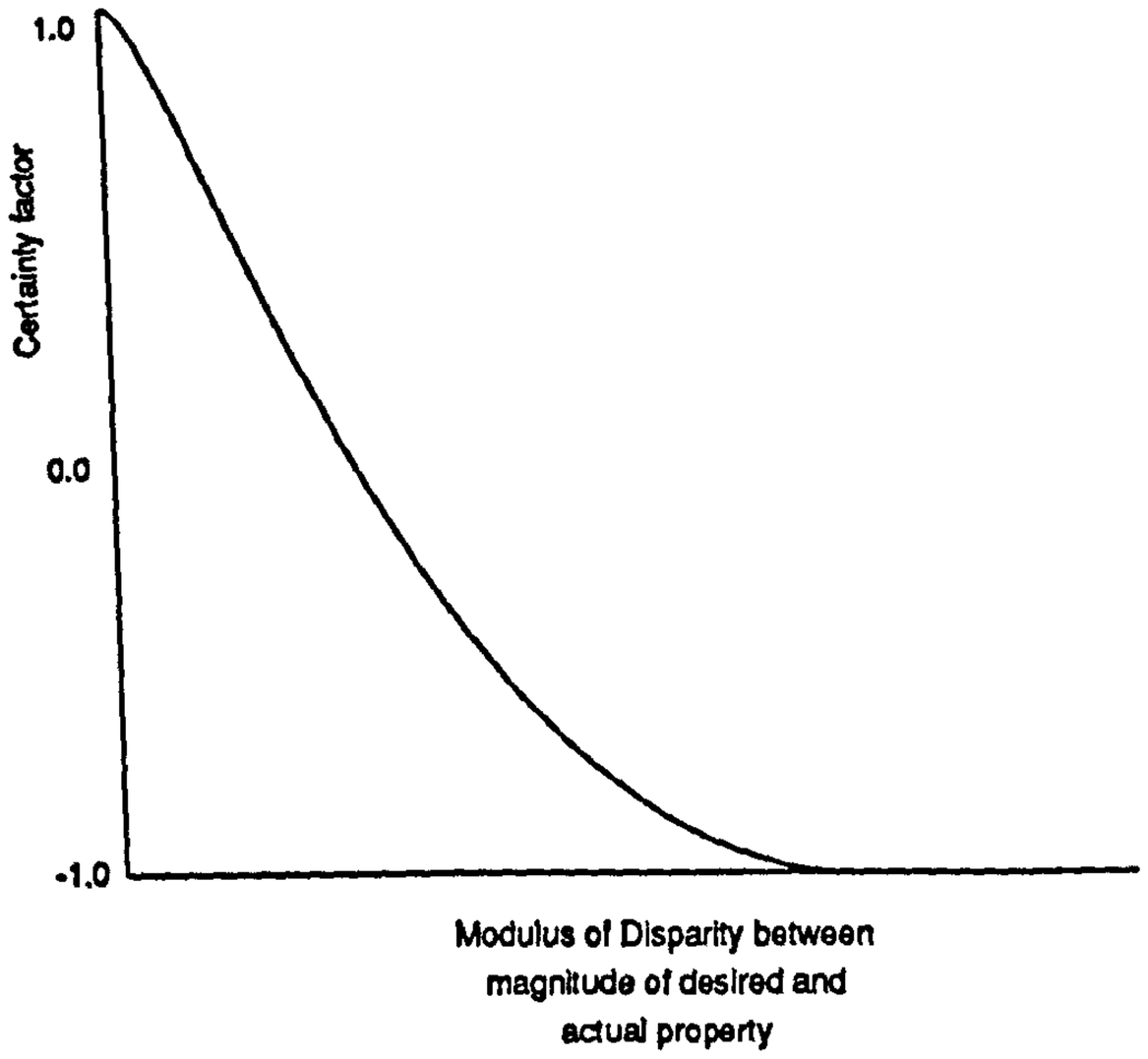


Figure 5.15: A General Method for Generating Initial Certainties

in section 4.3. Differences are only observed when the knowledge based system is instructed to give a running commentary on its actions. Figures 5.16 to 5.18 show an example of each of these modes of interaction. The extra information supplied by the knowledge based system appears on the users screen. This information can take the form of :-

- A display of each goal or subgoal currently being considered
- A display of each rule invoked to help satisfy a given goal
- A display of each modification to the certainty of an hypothesis

5.7.1 A Sample Terminal Session with the Combined System

This section describes in detail some initial results generated using the small rule base in figure 5.19 and the image in figure 5.20. The line segments in this image have been separated to show their fragmentation. The course of the terminal session is described in table 5.1.

```
The goal :-
 there are connected ends of Line-A and Line Feature-A
 is now being considered
```

Figure 5.16: Goal Display

```
Rule 2 has been invoked, which states that :-
  IF the end separation is less than 2.0 for Line-A and Line Feature-A
  AND the length is greater than 10.0 for Line-A
  AND the maximum linear deviation is greater than 2.0 for Line-A

  THEN the certainty of the hypothesis :-

  there are connected ends of Line-A and Line Feature-A
  may be altered using the minimum certainty from the above condition set,
  the certainty for the assertion, and the current certainty of the hypothesis
```

Figure 5.17: Rule Invocation Display

```
The certainty of the hypothesis :-
  there are connected ends of Line-A and Line Feature-A
  has been modified from 0.5 to 0.85 by rule 2
```

Figure 5.18: Hypothesis Modification

Each row of the table considers an hypothesised connection of a line feature to a line segment, corresponding to the numbered segments in figure 5.20. Within each row the certainty for each relevant production rule is given. Whenever the certainty of the main goal can be modified this certainty is presented as an additional column. When all relevant rules have been tested, and the final certainty of the main goal has been returned to SAFE, the action of SAFE is described by the action code sequence in the right hand column of the table. Automatic extension is possible when the certainty for the candidate extension is above 0.8 and no other candidates are above 0.7.

Although this rule base is embryonic and is to a certain extent restricted by the number of basis functions that have been developed, it is possible to see that a large proportion of candidate segments are linked automatically. For the system to be able to cope with discrimination in complex areas, the rule base must be extended considerably; for example, recognising segments which may belong to characters, recursive testing of segment connectivity or testing for constant curvature.

```
.RS
.CS
.CO function End Separation,is, 0.0,(LinFtr-A,Line-A)
.CE
.AS
.AC isconnected,is,true,(LinFtr-A,Line-A),1.0
.AE
.RE

.RS
.CS
.CO function End Separation,is less than,2.0,(LinFtr-A,Line-A)
.CO function Length,is greater than,10.0,(Line-A)
.CO function MaxLinearDeviation,is greater than,2.0,(Line-A)
.CE
.AS
.AC isconnected,is,true,(LinFtr-A,Line-A),0.9
.AE
.RE

.RS
.CS
.CO function End Separation,is between,2.0 and 5.0,(LinFtr-A,Line-A)
.CO function Length,is greater than,15.0,(Line-A)
.CO function MaxLinearDeviation,is greater than,2.0,(Line-A)
.CE
.AS
.AC isconnected,is,true,(LinFtr-A,Line-A),0.9
.AE
.RE


.RS
.CS
.CO function Length,is between,4.0 and 15.0,(Line-A)
.CO function ModePopulation,is less than,50.0,(Line-A)
.CE
.AS
.AC LineIsBlob,is,true,(Line-A),0.80
.AE
.RE

.RS
.CS
.CO LineIsBlob,is,true,(Line-A)
.CE
.AS
.AC IsConnected,is,true,(LinFtr-A,Line-A),-0.5
.AE
.RE
```

Figure 5.19: A Small Rule Base for Initial Testing

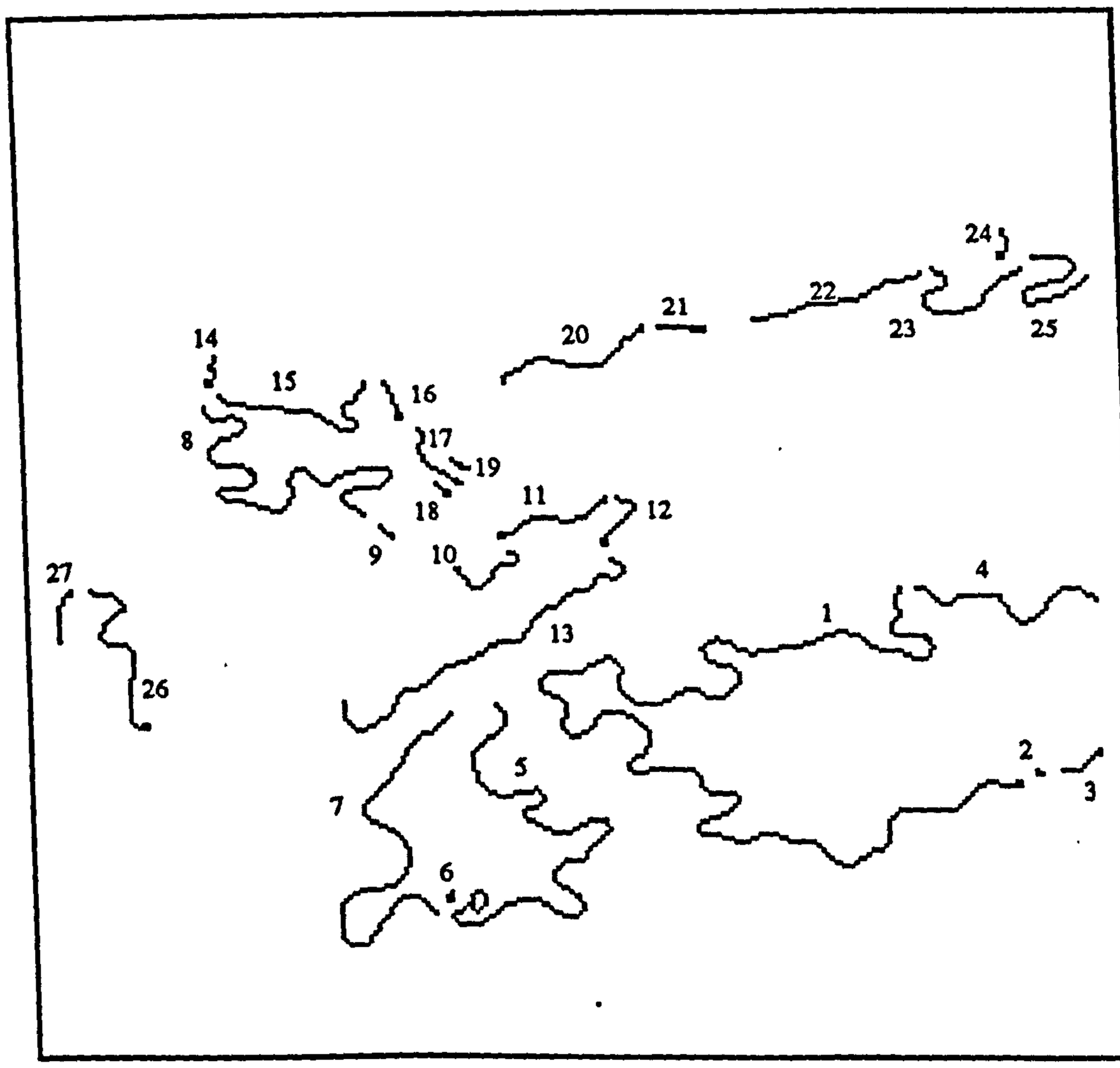


Figure 5.20: Segments involved in Hypothesised Connections

Table 5.1: Certainties of goals during a Sample session with the Combined System

Hypothesised Segment Connection	Certainties for goals														Action Code
	PR1	main	PR2			main	PR3			main	PR4		PR5	main	
	Co1	goal	Co1	Co2	Co3	goal	Co1	Co2	Co3	goal	Co1	Co2	Co1	goal	
1 to 2	0.55	0.55	0.75	0.2	0.87	0.63	1.0	-1.0			0.8	1.0	0.64	0.46	URMF3A
1 to 4	1.0	1.0													EMCA
5 to 6	0.5	0.5	0.7	0.4	0.91	0.61	1.0	-1.0			1.0	1.0	0.8	0.47	URMF7CA
8 to 9	0.64	0.64	0.84	0.48	0.87	0.80	1.0	-1.0			1.0	0.25	0.2		URMF10
10 to 11	0.78	0.78	0.98	1.0	0.99	0.97	1.0	1.0	0.99	0.99	-1.0				E
11 to 12	1.0	1.0													E
12 to 13	0.55	0.55	0.75	1.0	1.0	0.86	1.0	1.0	1.0	0.986	-1.0				E
8 to 14	0.64	0.64	0.84	1.0	0.93	0.91	1.0	0.58	0.93	0.96	1.0	0.4	0.32	0.95	N
8 to 15	0.5	0.5	0.7	1.0	1.0	0.81	1.0	1.0	1.0	0.98	-1.0				US15
15 to 16	0.78	0.78	0.98	1.0	0.9	0.96	1.0	0.72	0.9	0.98	1.0	0.04			E
16 to 17	0.8	0.8	1.0	1.0	1.0	0.98	1.0	1.0	1.0	0.99	0.37	0.33	0.26		E
17 to 18	0.64	0.64	0.84	0.58	0.9	0.83	1.0	0.08			1.0	0.7	0.56	0.76	N
17 to 19	0.55	0.55	0.75	0.54	0.88	0.77	1.0	0.04			1.0	0.1			URMF20A
20 to 21	1.0	1.0													EMF22A
22 to 23	1.0	1.0													E
23 to 24	0.8	0.8	1.0	0.98	1.0	0.97	1.0	0.46	1.0	0.99	1.0	0.5	0.4	0.98	N
23 to 25	0.70	0.70	0.78	0.24	0.86	0.67	1.0	-1.0			0.84	-0.5			US25MF26
26 to 27	1.0	1.0													E

Column header codes:

PR n Production rule n

Co n Condition clause n

Action Codes:

A	Reinvoke automatic mode	M	System reverts to manual mode
C	Close feature	R	User rejects all candidates presented
E	Feature automatically extended	Sn	User selects segment n from candidates presented
Fn	segment n found manually	U	User prompted to interact
N	consider next goal before decision		

Chapter 6

Conclusions and Further Work

The main intention of this thesis is to describe a novel approach to the task of automated feature extraction. The use of IKBS techniques in image processing is increasing. In general the approach is to embed a data driven inference mechanism within an image interpretation system in order that image segments may be classified by recognising segment configurations described in the condition sets of rules (Stansfield [1986]). The performance of such systems is reliant upon generating a complete and reliable IKBS system. Such systems cannot therefore be of practical utility until such time as the IKBS is sufficiently reliable to generate a full interpretation.

It would be unrealistic, considering the current state of the art of image processing and interpretation, to consider generation of a completely automated interpretation system for providing a feature database. However, the novel approach to utilising IKBS techniques described here offers a fully working interpretation system whilst the knowledge base of the IKBS is still in its infancy. The processing within the IKBS is constrained due to its goal directed philosophy and the system has the clarity and ease of extensibility characteristic of intelligent knowledge based systems.

To summarise, this project has achieved

- the derivation of low level image processing techniques for line feature extraction to give smooth representations of line feature fragments
- the derivation of a semi-automated feature extraction (SAFE) system to overcome the limitations of low level data driven line tracking techniques
- the establishment of an IKBS with an interface to allow cooperation with the SAFE system in order to reduce user interaction.

The system has potential applications in domains other than interpreting sea charts. Characteristics of the interpretation system may be altered by removing knowledge specific to the sea chart domain and replacing it with rules appropriate to the new application domain. Two potential applications could be the interpreting of engineering drawings or of sketch diagrams like those used by archaeologists to record the shape of artifacts. The latter task is mentioned as it is the authors intention to pursue this using the techniques developed in this thesis.

The continued development of this system will require expansion of the knowledge base for the IKBS and the extension of the SAFE / IKBS interface to deal with other situations where the user is called upon for information. For example the task of automatically labelling features with a classification is an obvious area for exploration. The ability to satisfy goals with partially instantiated entity lists has been incorporated into the system developed here, and further investigation into the behaviour of the IKBS when such goals are set up would also be of interest. The SAFE system could then ask the IKBS *"Can you find a segment which is connected to this feature?"* rather than the more limited *"Is this segment connected to the feature?"*.

This research has not tackled the problem of extraction of the character information from charts. The extension of the interpretation system to include this capability would require a thorough investigation of the large amount of research into character recognition that is currently being carried out. In addition the task of resolving image to image feature correspondence must be solved, i.e. linking map features fragmented that happened to lay across the border of an image. This might be solved by dealing with matrices containing images of full charts, acquired by some alternative image capture system.

The contribution which this thesis makes to the current state of interpretation of digital images is to demonstrate that with careful design, those tasks which it is currently highly desirable to perform in an automated fashion but which cannot be fully automated with the current state technology may be approached using the methods described here. Those elements of the task which require the domain knowledge for performing interpretation are passed to a cooperating IKBS. The IKBS can offer solutions more frequently as its knowledge base is enhanced.

Appendix A

Basic Theory

A.1 The Image Function

Image acquisition requires scanning of a *natural* image, which may be represented by a continuous scalar valued function f(x,y) where x and y are spatial variables and f(x,y) is the grey scale brightness level at the point (x,y). The scanning process involves quantisation of the image both spatially and in amplitude. Spatial quantisation is termed sampling. The digitised grey values of the picture are transmitted to a computer for storage in the form of a two dimensional matrix. This matrix may be represented by the discrete function f'(x,y) where x and y are integer variables and f'(x,y) is the quantised grey scale brightness level at the point (x,y).

A.2 Edge Enhancement and Detection

The most common methods for performing edge enhancement are those of digital convolution, cellular logic operations and modification of the Fourier spectrum of the image. The first two of these processes rely on computing an output pixels value based on the properties of a surrounding neighbourhood

of the pixel in question. The latter makes use of the Fourier transform which allows separation of those components in an image which confer fine detail and those which provide the smooth grey level changes.

Convolution in it's continuous form can be expressed by the equation :-

$$g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)h(x-x',y-y')dx'dy'$$

This equation demonstrates how each point in an image function $f(x,y)$ is spread by convolution into a the form of the function $h(x',y')$ with amplitude defined by the amplitude of the image function. The combination of these point spread functions is the convolved image $g(x,y)$.

The function $h(x',y')$ is usually finite. This feature along with a discretised form of the convolution function allows for the convolved image to be generated by machine computation. The discrete form of the equation is.

$$g'(x,y) = \sum_{y'=1}^{m}\sum_{x'=1}^{n} f(x,y)h(x-x',y-y')$$

The design of the operator function $h(x',y')$ defines the nature of the resultant convolved image. Operators defined in terms of a matrix, often termed masks, can be designed to enhance edges. Operators can be designed to approximate the first or second differentials of the image function, in which case edges are found at the maxima or zero crossings, respectively. Masks can also be designed in sets which attempt to model the image function surface. The latter allow edge orientation to be estimated.

The Fourier transform of an image is defined by the equation :-

$$F(U,V) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-2\pi i(Ux+Vy)}dxdy$$

and the inverse transform is given by

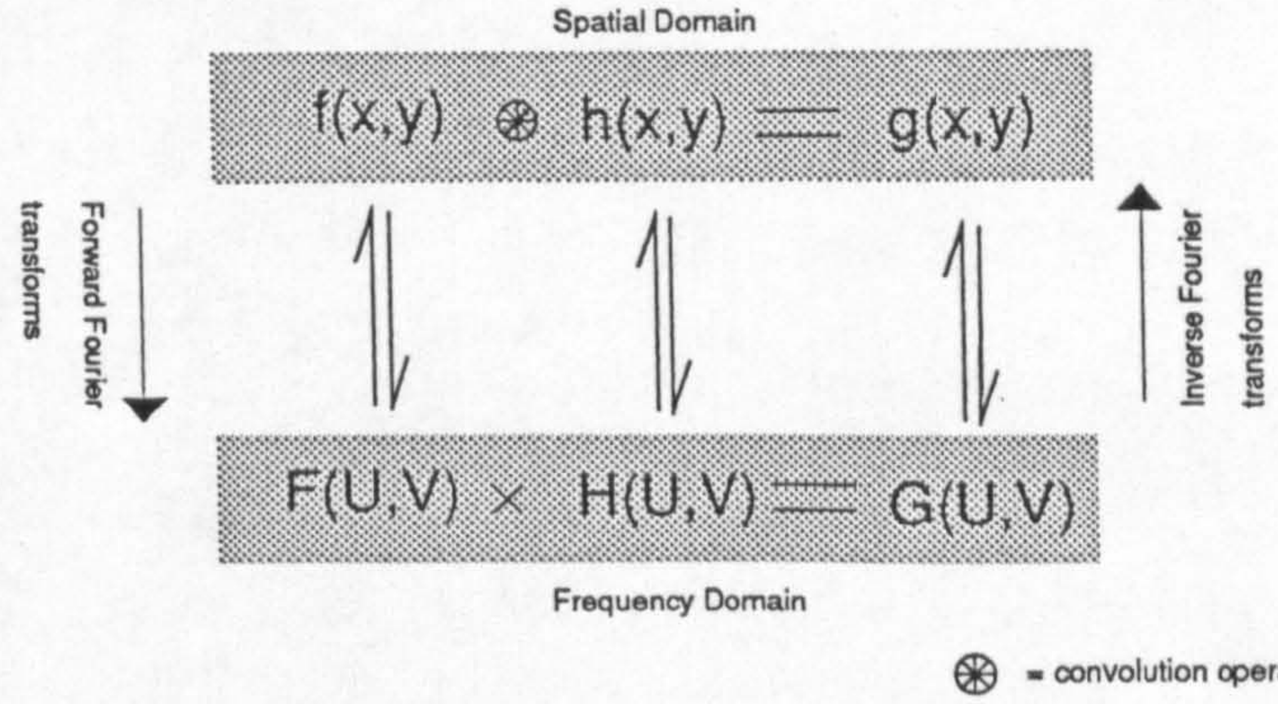
APPENDIX A. BASIC THEORY

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(U,V)e^{2\pi i(Ux+Vy)}dU\,dV$$

By representing the information contained within an image function in terms of the Fourier transform of the image the information in the image which confers fine detail to it can be separated from that which confers the smooth changes over the image. A region in the fourier spectrum (the modulus of the fourier transform) which contains strong peaks around the origin of the transform contains a high degree of smooth changes across the image. Further away from the origin the information corresponds to the fine detail in the image. Modifying the fourier transform by multiplication with a radially symmetric function allows enhancement of either the smooth changes or the edge detail information. A modification function, known as a spectral filter, with a central peak around the origin emphasises the low spatial frequency smooth changes and suppresses noise and is termed a low pass filter. One which has a central trough emphasises the higher spatial frequency edges and noise and is termed a high pass filter. After calculation of the inverse fourier transform the resultant image in the spatial domain is the desired modified image.

By the convolution theorem it can be shown that theoretically the processes of convolution and spectral filtration are the same. This theorem can be expressed graphically.

APPENDIX A. BASIC THEORY



This diagram shows that the results of convolution can be achieved by creating the fourier transforms of the two functions to be convolved, multiplying them together and performing the inverse transformation. Both convolution and fourier transformation are simple processes to implement computationally. The above theorem has more important implications when it is desired to perform the process of deconvolution. This is the process of regaining the original image function given a knowledge of the degrading function which has operated upon it. Attempting this process in the spatial domain is difficult. However, if the degraded image is Fourier transformed and the resultant function is divided by the transform of the degrading function then the original image may be regained by performing an inverse transform. Some practical problems occur in this operation such as when the degrading function falls to zero then the resultant transform tends to infinity. The new image is therefore very often an approximation to the original.

Appendix B

Extended Literature Review

In the relatively short history of the science of Image processing the new technology has been put to work in many areas. Industrial applications include automated inspection of products (Li et. al. [1982]) and machine tools (Hashim and Clements [1984]), robot vision for pick and place activities, welding etc. (Pugh [1983 & 1984]), medical uses include enhancement of radiographs (Cocklin et.al. [1984]) photomicrographs and computer aided tomography. It has been widely used for enhancement of satellite images (Schowengerdt [1983]) and aerial reconnaissance photographs, and is incorporated into instruments such as electron microscopes.

B.1 Segmentation

Haralick [1983], in his survey on segmentation states that there is no theory of image segmentation. Processes rely on ad-hoc methods of producing regions of coherent properties. Fu and Mui [1981] define a segmentation of an image matrix X in terms of a uniformity predicate P so that X is partitioned into

non empty subsets, $X_1, X_2, X_3, ..., X_n$, such that :-

$$\bigcup_{i=1}^{N} X_i = X \tag{B.1}$$

$$X_i, \quad i = 1,2,...,N \quad \text{is connected} \tag{B.2}$$

$$P(X_i) = TRUE \text{ for i=1,2,...,N} \tag{B.3}$$

$$P(X_i \bigcup X_j) = \text{FALSE for } i \neq j \quad \text{where } X_i \text{ and } X_j \text{ are adjacent.} \tag{B.4}$$

This set of equations states that every pixel belongs to a single subset of the image, no pixel is left uncategorised and subsets with a common border in the image do not conform to the uniformity predicate in the same manner. They categorise methods for segmentation into three groups :-

- Characteristic feature thresholding or clustering
- Edge detection
- Region extraction.

B.1.1 Characteristic Feature Thresholding

Characteristic feature thresholding may be described by the relationship :-

$$S(x,y) = k \quad \text{if} \quad T_{k-1} \leq f(x,y) < T_k \ (k = 0,1,2,...,m) \tag{B.5}$$

where (x,y) are the spatial coordinates of a pixel; $S(x,y), f(x,y)$ are the segmented and the characteristic feature functions of (x,y) respectively. The characteristic feature function may simply be the grey level of the original image or an edge map or some other function in characteristic feature space resulting from the preprocessing stage.

These thresholding operations can be further categorised into 1) global thresholding, for which a point in the segmented image is determined only by the corresponding pixel in the input image, 2) local thresholding, for which a pixel in the segmented image is determined by the values of the pixels in a small neighbourhood of the corresponding pixel in the input image, and 3) dynamic thresholding where the output image pixel is determined by the spatial coordinates as well as the neighbourhood of the input image pixel. Clustering is a multidimensional extension of thresholding in which each class of regions forms a cluster in characteristic feature space. These clusters are then mapped back to the original image domain to produce the segmentation.

Pal et al. [1983] shows how the theory of fuzzy sets can be used to automate the selection of an optimum set of thresholds without referring directly to the image histogram. This process finds the optimum thresholds by examining the entropy of a fuzzy set. Kitler et. al. [1985] review threshold selection techniques before presenting a new method based on a simple image statistic.

B.1.2 Edge Detection

Edge detection is based on finding the grey level discontinuities in an image. Most of the information within an image lies on the boundaries between objects, and edge detection seems to be an important process in human vision.

A survey by Davis [1975] is still referenced by current workers as an important source of information on the early work on edge detection. Early work on developing edge detectors concentrated on single operators which provide approximations to first or second differentials of the image function or set of operators which model edges of different orientations. These small single operators have different properties of accuracy of localisation of edges and response to noise, based on the size of the operator mask and the various

weightings within the mask. The operators detect edges which occur over an image area of the order of the mask size. More recently attention has been concentrated on detecting edges at all scales with good spatial localisation. Accounting for noise in directional differentiation schemes involves computing the difference between the weighted sums of two, one-dimensional or two-dimensional local areas (Prewitt [1970]). In general the larger these areas are the less the effect of noise but small scale edges are consequently undetected. Similarly the effect of noise may be accounted for in computing the gradient of an image by extending the size of the region used in computing the gradient at a point. The weaknesses of linear edge detectors are that isolated bright points can lead to the detection of an edge and that the output of a local operator will be high in an extended region around the true edge point.

Marr and Hildreth [1980] consider the way in which natural image functions change over a wide range of scales and therefore conclude that no single filter can be optimal at all scales. They propose an optimal filter for smoothing an image by taking local averages of an image at various resolutions.

The purpose of filtering the image is to reduce the range of scales over which intensity changes take place, therefore the optimal filter should be band limited in the frequency domain. However, the elements of a scene over which intensity changes take place are spatially localised and therefore a point in the filtered image should be derived from a smooth average of nearby points. The optimal compromise for these two conflicting requirements is that the filter has a gaussian profile. Marr and Hildreth then go on to consider the edge detection problem as one of searching for zero crossings in the second directional derivative of an image function rather than the more common approach of searching for peaks in the first derivative. A filter which performs

both the smoothing and differentiation operations is the second derivative of the gaussian function which they term the $\nabla^2 G$ filter. The advantage of this filter over the similar Laplacian approximation operation is that the Laplacian offers no directional information. The $\nabla^2 G$ filter is similar to Wilson and Giese's [1977] difference of gaussians filter (DOG). The paper goes on to describe a search strategy for the detection of zero crossings and their directions and interpreting the functions obtained by applying the operator at various resolutions.

Canny [1983], in an attempt to derive the optimal linear shift invariant edge operator, begins with a model of a step edge in white gaussian noise and formulates the criteria that capture the desirable properties of an edge operator. These are that :-

- The detector has low probability of failing to mark edges or falsely marking non-edges.

- The marked edge should be as close as possible to the true edge position.

- There should be low probability of more than one response to a true edge.

The first two of these criteria allow formulation of an uncertainty principle. Good detection properties are achieved at the expense of localisation. Low detection error rates correspond to high signal to noise properties of the operator, which in turn corresponds to large operators. Localisation of edges is worst in general with large operators. Canny derives mathematical measures of the signal to noise and localisation properties of operators and combines them to express the uncertainty principle. The expression itself demonstrates that a difference of boxes operator is the optimal detector. The difference of

boxes operator generates a number of small maxima in addition to the maximum coincident with the true edge site. By adding a constraint corresponding to the criterion related to multiple responses he was able to derive an optimal operator for which the impulse response is a sum of damped exponential cosines. This can be approximated by the first derivative of a gaussian. This operator exhibits far less of a tendency to produce spurious maxima in the vicinity of an edge. Canny applies a method of non-maximum suppression to remove the effects of these false edge points. In order to deal with differences in image signal to noise ratio it is necessary to use multiple operator widths. The edge points found by each of the operators are combined in a stage where the output from the smallest operator with sufficiently high signal to noise properties is used to predict the outputs of the larger operators. For edge points generated by the larger masks to contribute to the final edge point map there must be sufficient evidence that the operator has found a true edge point not generated by the prediction stage. Canny shows how the initial derivation process used to derive this optimal filter can be applied to step edges in non white noise and to detecting other structure such as ridge and roof profiles.

Many edge maps consists of individual edge points which bear no interrelation to one another. Tavakoli and Rosenfeld [1982] propose a method of using least squares fitting to connected edge elements and then uses figures of merit derived from measures of compatibility of edge segments to produce links with an associated link strength. The use of link strength allows relaxation techniques to be used in obtaining the optimum edge segment linkage.

The need for quantitative comparison of edge operators is expressed by Peli and Malah [1982]. He takes known edge operators and compares their

performance on computer simulated edges using known performance measures. Some conclusions are drawn on the optimum edge operators for given situations but he states the need for better performance measures to test edge detection methods.

Burrus and Parks [1985] cover both the theory and the programming aspects of implementing Fourier transform and convolution algorithms. Eklundh [1981] tackles the problem of efficiently transposing a matrix from being stored in secondary computer memory with the smallest accessible record being an image row to one in which the columns are the smallest records accessible. This problem has to be overcome when implementing a two dimensional Fourier transform, using it's separability property (Gonzalez and Wintz [1977]) , in limited high speed memory. Two volumes edited by Huang [1981] concentrate on aspects of two dimensional transformations and filtering operations. A publication by Cracknell [1982] resulting from a postgraduate summer school at Dundee university presents programmed examples in Fortran of some basic processing routines, based on the text by Gonzalez and Wintz.

B.1.3 Region Extraction

Region extraction techniques can be subdivided into three categories :-

- region merging,
- region splitting and
- merging and splitting.

The overall strategy of these techniques is to create regions which are iteratively combined or subdivided, depending on some measure of boundary strength, until only strong boundaries remain (Brice and Fenema [1970]). An

account is given of the split and merge technique by Browning and Tanimoto [1982] who then use this technique on equal sized *tiles* of an image. This allows each tile to be held in primary storage of a minicomputer. The resultant segmentations are subjected to a linking algorithm to provide the global segmentation.

Beaulieu and Goldberg [1983] survey segmentation by region extraction techniques and go on to present an algorithm which uses stepwise optimisation methods on hierarchical representations of images. Each iteration of the algorithm merges the two regions which add the smallest value to a cost function . This cost function represents the loss of information caused by the given partition of the image.

In contrast to the sequential programming techniques employed for implementing the processes described so far Nazif and Levine [1984] incorporate the knowledge used in segmentation processes into an expert low level segmentation system. This new approach allows the image data to drive the segmentation process by triggering rules which then invoke appropriate procedures when rules are matched to the data. The system creates a basic set of regions and lines from an image and then allows rules to act upon these segments which embody knowledge of how in general lines exhibit continuity and well behaved intersections and how they bound regions. Hence region and edge / line extraction processes can interact with one another and guide the segmentation process to overcome the deficiencies in the basic feature extraction processes. Higher level rules allow control of the overall processing strategy and facilitate a focus of attention scheme so that processing may be concentrated on the interesting areas of an image.

B.2 Scene Analysis

The process of scene analysis relies on obtaining some higher level description of the segmented objects within the image. This object description may then be used for pattern recognition and classification of the object by comparison with stored representations, the data may be used in some decision process or stored as information, valuable in it's own right. Kitchin and Pugh [1983] have collected together the most common object descriptors in a tutorial paper which describes their concepts and implementations. The medial axis transform (Mott-Smith [1970], Wang et. al. [1981] and Peleg and Rosenfeld [1981]) is a description of an object in terms of connected lines which resemble matchstick type drawings. These representations hold information about the general shape and size of the object. Freeman [1974] in an extensive treatise of line drawing images, described a method of coding lines based on the direction of movement from a pixel on a line or boundary to an *eight connected* neighbour on the line or boundary. This Freeman chain code can be the basis for higher level descriptions of the objects by employing such processes as polygonal approximation of the line, or taking the Fourier (Granlund [1972],Kuhl and Giardina [1982]) or Walsh transforms (Sarvarayudu [1983]) of the chain code.

B.3 Maps and Geographic Information Systems

This section is by no means an exhaustive review of geographic information systems, but rather a collection of papers which would seem to be significant for future stages of the project. The references concentrate on a popular alternative to the regular grid matrix representation of images, the quadtree, which is being used extensively for storing map data.

However the data may be extracted from maps or aerial images, it must be

arranged in a suitable manner for convenient retrieval, logical and geometric manipulation and editing. Retrieval may involve selection of features or accessing subsets of feature data for displaying a map at different scales and resolutions.

The problems of storage and manipulation of large numbers of images held in matrix form are tackled by Chock et. al. [1984] who describe their picture data base management system (PICDMS) which can build multiple variable data bases from pictures, maps or drawings and allow manipulation of these images using simple logical commands.

Samet [1984] has also produced a survey which gives a thorough review of the quadtree and related structures. The quadtree is a representation of an image using hierarchical decomposition of an square region into subquadrants to form a tree structure of outdegree four. Each node corresponds to a subquadrant of side length 2^{-m} where m is the level of the node with reference to the root node at level 0. A binary image may be represented by colouring a node as black if the corresponding subquadrant is uniformly black, white if the subquadrant is white, or grey if the subquadrant is non-uniform. The subquadrants of a uniform region are obviously uniform, and it is therefore unnecessary to store further nodes in the quadtree corresponding to these subquadrants. A black or white node is termed a leaf.

Abel [1985] describes manipulations on quadtrees involving set operations, such as the combination of a number of data sets of the same region containing different attributes of that region, an operation to which the quadtree is well suited. Samet et. al. [1984] describes techniques for storage of quadtrees in a suitable form for editing and gives details of a working editor for their geographic information system.

Tamminen [1984] describes an alternative to the common method of storage

of quadtrees which is to include pointers in the records of quadtree nodes from parent node to offspring node and vice versa. He concentrates on the data compression attributes of the quadtree representation in describing the linear quadtree. The linear quadtree representation is a to create a sequential file in which 1s represent an internal node and 0s represent a leaf which is followed by the colour of the leaf. When a region is recognised as an internal node then it's subquadrants are examined in the order top left, top right, bottom left, bottom right. If a subquadrant is recognised as an internal node then it's subquadrants are immediately examined in the same order until the individual pixel level is reached. Thus the sequence of the file defines the parent / offspring relationship.

Jones and Iyengar [1983] describe a data compression technique which preserves the information contained in an explicitly stored quadtree structure. This involves storing every four 'brothers' in a tree structure as a single record known as a metanode. The record contains a pointer to the metanode which contains the father node, the colour or grey level of each node and a pointer to the first metanode that represents offspring of the current metanode. This represents a space saving of approximately 85 percent compared with the storage for the explicitly stored structure. They present results on the time comparative time efficiency of these structures for different types of operation.

Ranade et. al. [1982] use the hierarchical structure of the quadtree to show how the shape approximation obtainable at each level in a quadtree approaches the shape of the full representation by examining the convergence of the first few moments of each representation. They propose that this type of approximation can be used in shape matching processes, at least for the case of the type of images under consideration which were taken from a data base of over a hundred aeroplane silhouettes.

The quadtree of a given shape is dramatically altered by difference in location , size and orientation within the image. Chien and Aggarwal [1983] normalise objects with respect to their centroid and principal axes and standardise the size before quadtree generation. This produces an object representation which is position rotation and size invariant and is therefore suitable for object recognition purposes.

Appendix C

References

1. Abel D.J., *Some Elemental Operations On Linear Quadtrees For Geographic Information Systems*, The computer Journal (British Computer soc.), Vol. 28, no. 1, pp. 73–77, (1985).

2. Alty J.L., Coombs M.J., *Expert Systems, concepts and examples*, Pub. N.C.C. publications (1984).

3. Amin T.J., Kastura R., *Map data processing: Recognition of lines and symbols*, Optical Engineering, Vol. 26, no. 4, pp. 354-358 (1987).

4. Ballard D.H., Brown C.H., *Computer Vision*, Pub. Prentice Hall, (1982).

5. Beaulieau J.H., Goldberg M., *Stepwise optimisation for Hierarchical Picture Segmentation*, I.E.E.E. Conf. on Comp. vis. and patt. rec. Washington D.C. (1983).

6. Bertolazzi P., Pirozzi M., *A parallel algorithm for the optimal detection of a noisy curve*, Comp. vis. Graph. & Image. Proc., Vol. 27, no. 3, pp. 380–386, (1984).

7. Black W., Clement T.P., Harris J.F., Llewllyn B., Preston G., *A general*

purpose follower for line structured data, Patt. Rec., Vol. 14, pp. 33–42, (1981).

8. Brice C.R., Fennema C.L., *Scene analysis using regions*, Artificial Intelligence, Vol. 1, pp. 205–226, (1970).

9. Browning J.D., Tanimoto S.L., *Segmentation of pictures into regions with a tile by tile method*, Patt. Rec., Vol. 15, no. 1, pp. 1–10, (1982).

10. Burrus C.S., Parks T.W., *D.F.T. / F.F.T. and convolution algorithms, Theory and implementation*, Pub. Wiley and sons, (1985).

11. Canny J.F., *Finding Edges and Lines in Digital Images*, M.I.T. Internal Report, (1983).

12. Castleman K.R., *Digital image processing*, Pub. Prentice Hall, (1979).

13. Caponetti L., Chiaradia M.T., Distante A., Veneziani M., *A track following algorithm for contour lines of digital binary maps*, In Levialdi (ed.) *Digital Image Analysis* Pub. Pitman (1982).

14. Chien C.H., Aggarwal J.K., *A normalised quadtree representation*, I.E.E.E. Conf. on comp vis. & patt rec. Washington D.C. (1983).

15. Chock M., Cardenas A.F., Klinger A., *Data base structure and manipulation capabilities of a picture data base management system (PICDMS)*, I.E.E.E. Trans on P.A.M.I., Vol. 6, no. 4, pp. 484–492, (1984).

16. Clocksin W.F., Mellish C.S., *Programming in Prolog*, 2nd ed., Pub. Springer-Verlag (1984).

17. Cocklin M., Gourlay A., Jackson P., Kaye G., Miessler M., Kerr I., Lams P., *An image processing system for digital chest X ray pictures*, Computer programs in biomedicine, Vol. 19, pp. 3–11, (1984).

18. Cracknel A.P., *Computer programs for Image processing of remote sensing data*, Pub. Univ. of Dundee, (1982).

19. Davis L.S., *A survey of edge detection Techniques*, Comp. Graph. & Im. Proc., Vol. 4, no. 3, pp. 248–270, (1975).

20. Davis R., *Meta Knowledge : Reasoning about control*, Artificial intelligence, Vol. 15, pp. 179-222. (1980).

21. Duda R.O., Hart P.E., *Use of the Hough transform to detect lines and curves in pictures*, Comm. Ass. Comput. Mach., Vol. 15, pp. 11–15, (1972).

22. Eklundh J.O., *Efficient Matrix transposition*, In Huang T. S. (ed.) *Topics in applied physics*, Vol.43, *Two dimensional digital signal processing II, Transforms and median filters*, Pub. Springer Verlag (1981).

23. Faugeras O.D. (ed.), *Fundamentals in computer vision*, Pub. Cambridge Univ. Press, (1983).

24. Freeman H., *Computer processing of line drawing images*, Comput. Surv., Vol. 6, no. 1, pp. 57–97, (1974).

25. Freeman H., *Boundary encoding and processing*, In Lipkin B., Rosenfeld A., *Picture processing and Psychopictorics*, Pub. Academic Press, (1970) pp. 241–266.

26. Freeman H., *On the encoding of arbitrary geometric configurations*, In Aggarwal, *Computer methods in image analysis*, Pub. I.E.E.E., (1977).

27. Fu K.S., Mui J.K., *A survey on image segmentation*, Patt. Rec., Vol. 13, pp. 3–16, (1981).

28. Fulford M.C., *The FASTRAK Automatic digitising system*, Patt. rec., Vol. 14, no. 1-6, pp. 65–74, (1981).

29. Gonzalez R.C., Wintz P., *Digital image processing*, Pub. Addison Wesley, (1977).

30. Goodson K.J. and Lewis P.H., *Feature Extraction from Line Drawing Images*, Lecture notes in Computer Science 301, Pattern Recognition: Proceedings of the 4th International Conference, Cambridge U.K., Pub. Springer-Verlag pp. 216–221, (1988) .

31. Granlund G.H., *Fourier preprocessing for hand print character recognition*, I.E.E.E. Trans Comput., Vol. C-21, pp. 195–201, (1972).

32. Groch W., *Extraction of line shaped objects from aerial images using a specular operator to analyse the profiles of functions*, Comp. Vis. Graph. & Im. Proc., Vol. 18, no. 4, pp. 347–358, (1982).

33. Haralick R.M., *Image segmentation survey*, In Faugeras O.D., *Fundamentals in computer vision*, Pub. Cambridge Univ. Press, (1983).

34. Haralick R.M., Watson L.T., Laffey T.J., *The Topographic Primal Sketch*, Int. J. Robotics research, Vol. 2, pp. 50 - 72 (1983).

35. Hashim A.A., Clements P.E., *Computer vision in a manufacturing process*, Proc. 4th Int. Conf. on Robot Vision & sensory control, (1984).

36. Huang T.S., *Two dimensional digital signal processing I, Linear filters*, In *Topics in applied physics*, Vol. 42, Pub. Springer Verlag, (1981).

37. Huang T.S., *Two dimensional digital signal processing II, Transforms and median filters*, In *Topics in applied physics*, Vol. 43, Pub. Springer Verlag, (1981).

38. Jones L.P., Iyengar S., *Virtual Quadtrees*, Proc. I.E.E.E. conf on Comp. Vis. and Patt. Rec, Washington D.C., pp.133-135, (1983).

39. Kitchin P.W., Pugh A., *Processing of binary images*, in Pugh A., *Robot vision*, Pub. I.F.S. (Publications) ltd. & Springer Verlag, Berlin, (1983).

40. Kittler J., Illingworth J., Foglein J., *Threshold selection based on a simple image statistic*, Comp. vis. graph. and im. proc., Vol. 30, no. 2, pp. 125–147, (1985).

41. Kuhl F.P., Giardina C.R., *Elliptic fourier features of a closed contour*, Comp. vis. Graph and Im. proc., Vol. 18, no. 3, pp. 236–258, (1982).

42. Lee H.C., Fu K.S., *Using the fast fourier transform to determine digital straight line chain codes*, Comp. Vis. Graph and Im. Proc., Vol. 18, no. 4, pp. 359–386, (1982).

43. Levine M.D., Nazif A.H., *Rule based image segmentation: A dynamic control strategy approach*, Comp. vis. Graph. & im. proc., Vol. 32, pp. 104-126 (1985).

44. Marr D., Hildreth E., *Theory of edge detection*, Proc. Royal soc. London, Vol. B-207, pp. 187–217 (1980).

45. McKeown D.M., Harvey W.A., McDermott J., *Rule based interpretation of Aerial Imagery*, I.E.E.E. trans. on P.A.M.I., Vol. 7, no. 5,pp. 570-585 (1985).

46. Merrill R.D., *Representations of contours and regions for efficient computer search*, Commun. A.C.M., Vol. 16, no. 2, pp. 69–82, (1973).

47. Middleton A.J., Taylor M.J., *The storage and manipulation of digital maps for command control and communication applications*, Personal communication (1985).

48. Miller S.W., Iyengar S., *Representations of regions of map data for efficient comparison and retrieval,* Proc. I.E. E.E. Conf. comput. vis. and patt. rec. Washington D.C., (1983).

49. Mott-Smith J., *Medial axis transformations,* In Lipkin B. and Rosenfeld A., *Picture processing and psychopictorics,* Pub. Academic press, (1970).

50. Nadler M., *Survey: Document segmentation and coding techniques,* Comp. Vis. Graph. and Im. Proc., Vol. 28, no. 2, pp. 240–262, (1984).

51. Nazif A.M., Levine M.D., *Low level image segmentation, an expert system,* Proc. I.E.E.E. Trans on P.A.M.I., Vol. 6, no. 5, pp. - , (1984).

52. Nevatia R., Babu K.R., *Linear feature extraction and description,* Comp. graph. and im. proc., Vol. 13, pp. 257–269, (1980).

53. Pal S.K., King R.A., Hashim A.A., *Automatic grey level thresholding through index of fuzziness and entropy,* Pattern recogn. lett., Vol. 1, pp. 141–146, (1983).

54. Peleg S., Rosenfeld A., *A min-max medial axis transform,* I.E.E.E. trans on P.A.M.I., Vol. 3, no. 2, pp.208–210, (1981)

55. Peli T., Malah D., *A study of edge detection algorithms,* Comp. vis. graph. and Im. proc., Vol. 20, no. 1, pp 1-21, (1982).

56. Pferd W., Ramachandran K., *computer aided automatic digitising of engineering drawings,* Proc. I.E.E.E. Comp. soft. Applications j., pp. 630-635, (1978).

57. Pratt W.K., *Digital image processing,* Pub. Wiley, New York, (1982).

58. Prewitt J.M.S., *Object enhancement and extraction,* in Lipkin B., Rosenfeld

A., *Picture processing and psychopictorics*, Pub. Academic press, pp. 75-149, (1970).

59. Pugh A. (ed.), *Robot vision, International trends in manufacturing technology*, Pub. I.F.S. publications Ltd. and Springer Verlag, (1983).

60. Pugh A. (ed.), *Proc 4th Int. conf. on Robot vision and sensory control*, Pub. I.F.S. Publications, (1984).

61. Ramachandran K., *A coding method for vector representation of engineering drawings*, Proc. I.E.E.E., Vol. 68, pp. 813-817, (1980).

62. Ranade S., Rosenfeld A., Samet H., *Shape approximation using quadtrees*, Patt. Recogn., Vol. 15, no. 1, pp. 31-40, (1982).

63. Rosenfeld A., *Survey: Picture Processing 1986*, Comp. Vis. Graphics and Im. Proc., Vol. 38, pp. 147-225, (1987).

64. Rosenfeld A., Kak A.C., *Digital Picture Processing*, Pub. Academic Press, (1982).

65. Saghri J.A., Freeman H., *Analysis of the precision of generalised chain codes for the representation of planar curves*, I.E.E.E. trans on P.A.M.I., Vol. 3, no. 5, pp. 533-539, (1981).

66. Samet H., Webber R.E., *Using quadtrees to represent polygonal maps*, Proc. I.E.E.E. conf on comp. vis. and Patt. recogn., Washington D.C., (1983).

67. Samet H., Rosenfeld A., Shaffer C., Webber R.E., *Quadtree region representation in cartography: experimental results*, Proc. I.E.E.E. conf on comp. vis. and Patt. recogn., Washington D.C., (1983).

68. Samet H., Rosenfeld A., Shaffer C., Webber R.E., *A geographic information system using quadtrees*, Patt. Recogn., Vol. 17, no. 6, pp. 647-656, (1984).

69. Samet H., *The quadtree and related hierarchical data structures*, Comp. Surv., Vol. 16, no. 2, pp. 187-260, (1984).

70. Sarvarayudu G.P.R., Sethi I.K., *Walsh descriptors for polygonal curves*, Patt. Recogn., Vol. 16, no. 3, pp. 327-336 (1983).

71. Schowengerdt R.A., *Techniques for image processing and classification in remote sensing*, Pub. Academic press, (1983).

72. Sell P.S., *Expert Systems - A practical introduction*, Pub. Macmillan, (1985).

73. Smith R.W., *Computer processing of line images: A survey*, Patt. Rec., Vol. 20, no. 1, pp. 7-15 (1987).

74. Stansfield S.A., *ANGY: A rule based expert system for automatic segmentation of coronary vessels from digital subtracted angiograms*, I.E.E.E. trans. on P.A.M.I., Vol. 8, no. 2, pp. 188-199 (1986).

75. Suzuki S., Kosugi M., Hoshino T., *Automatic line drawing recognition large scale maps*, Opt. Eng., Vol. 26, no. 7, pp. 642-649 (1987).

76. Tamminen M., *Encoding pixel trees*, Comp. vis. graph. and im. proc., Vol. 28, pp. 44-57, (1984).

77. Tavakoli M., Rosenfeld A., *Edge segment linking based on grey level and geometrical compatibilities*, Patt. recogn., Vol. 15, no. 5, pp. 369-377, (1982).

78. Thompson, Thompson, *Inside an expert system*, Byte, April (1985).

79. Watson L.T, Arvind K., Ehrich R.W., Haralick R.M., *Extraction of lines and regions from grey tone line drawing images*, Patt. Recogn., Vol. 17, no. 5, pp. 493-507, (1984).

80. Tsuji T.J., Nakano H., *Knowledge based identification of artery branches in cine-angiograms.*, Proc. int. j. conf. on Artificial Intelligence, (1981)

81. Wang S., Wu A., Rosenfeld A., *Image approximation from grey scale medial axes*, I.E.E.E. trans on P.A.M.I., Vol. 3, no. 2, pp. 687-697 (1981).

82. Wilson H.R., Giese S.C., *Threshold visibility of frequency gradient patterns*, Vision research, Vol. 17, pp. 1177-1190 (1977).

83. Woetzel G., *A fast and economic scan to line conversion algorithm*, Patt. recogn., Vol. 12, pp. 125-129, (1978).

84. Wojcik Z.M., *An approach to the recognition of contours and line shaped objects*, Comp. vis. graph. and im. proc., Vol 25, no. 2, pp. 184-204, (1984).

85. Yachida M., Ikeda M., Tsuji S., *A knowledge directed line finder for analysis of complex scenes*, Int. joint conf. on Artificial Intelligence, (1979).