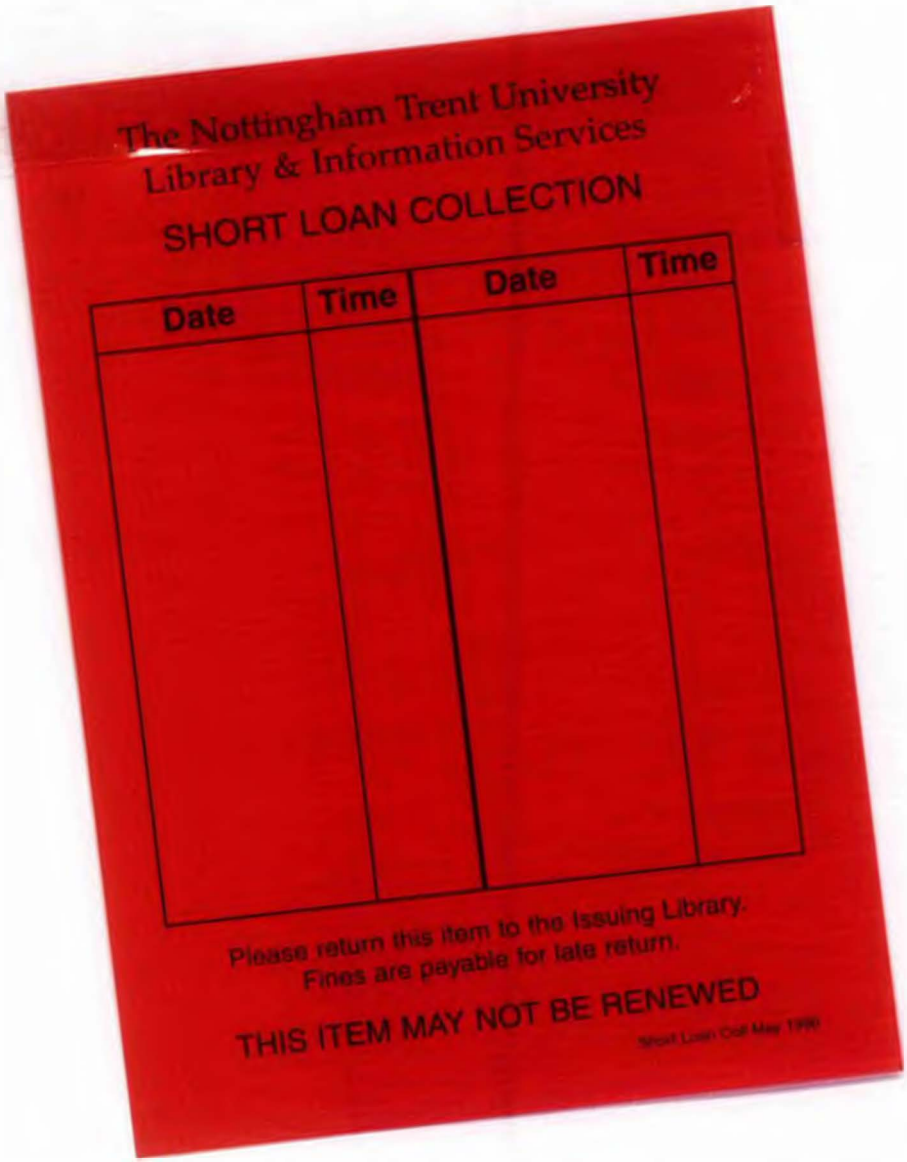

The Nottingham Trent University
Library & Information Services
SHORT LOAN COLLECTION
Date
Time
Date
Time
Please return this item to the Issuing Library.
Fines are payable for late return.
THIS ITEM MAY NOT BE RENEWED
Short Loan Coll May 1999

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

CPHD6

CHARACTER RECOGNITION IN UNCONSTRAINED ENVIRONMENTS

J.R. COWELL B.Sc, M.Phil.

This thesis is submitted to the Council for National Academic Awards in partial fulfilment of the requirements for degree of Doctor of Philosophy.

Nottingham Polytechnic,

Department of Computing.

October 1990.

CHARACTER RECOGNITION IN UNCONSTRAINED ENVIRONMENTS

J.R. Cowell

ABSTRACT

A multi-stage algorithm is devised for the recognition of alpha-numeric characters in unconstrained environments. The images used for examining the performance of the new algorithms are digitized pictures of vehicles in a wide variety of typical environments. The alpha-numeric characters read are on the vehicle licence plates.

The recognition process begins with a new algorithm for finding the licence plate region in the image. The extraction of the individual characters is by a novel region growing technique. As a precursor to the identification of the characters a new thinning algorithm is used, which is both faster than conventional methods and yields skeletons which conform more closely to the forms which are produced by humans than by traditional thinning techniques. A syntactic approach is used for the final stage of the recognition process. The selection of primitives is novel and it is based upon nodes where the character strokes intersect. Neither the size nor the orientation of the strokes affect the recognition process. The pattern grammar developed yields the same string for a wide range of patterns representing alpha-numeric characters irrespective of their position, size and orientation.

The thesis is illustrated throughout by the detection and reading of vehicle licence plates in unconstrained environments; however, the algorithms and underlying techniques are appropriate to a wide range of applications.

CHARACTER RECOGNITION IN UNCONSTRAINED ENVIRONMENTS

J.R. Cowell

ABSTRACT

A multi-stage algorithm is devised for the recognition of alpha-numeric characters in unconstrained environments. The images used for examining the performance of the new algorithms are digitized pictures of vehicles in a wide variety of typical environments. The alpha-numeric characters read are on the vehicle licence plates.

The recognition process begins with a new algorithm for finding the licence plate region in the image. The extraction of the individual characters is by a novel region growing technique. As a precursor to the identification of the characters a new thinning algorithm is used, which is both faster than conventional methods and yields skeletons which conform more closely to the forms which are produced by humans than by traditional thinning techniques. A syntactic approach is used for the final stage of the recognition process. The selection of primitives is novel and it is based upon nodes where the character strokes intersect. Neither the size nor the orientation of the strokes affect the recognition process. The pattern grammar developed yields the same string for a wide range of patterns representing alpha-numeric characters irrespective of their position, size and orientation.

The thesis is illustrated throughout by the detection and reading of vehicle licence plates in unconstrained environments; however, the algorithms and underlying techniques are appropriate to a wide range of applications.

ACKNOWLEDGMENTS

The author would like to thank Dr. D. Al-Dabass for his guidance and support throughout the duration of the research work. I would also like to thank my company, GPT for the use of the computing facilities.

CONTENTS

FIGURES

Chapter 2. CURRENT TECHNIQUES

Chapter 3. A NEW MULTI-STAGE ALGORITHM

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

Traffic congestion is becoming an increasing problem as traffic volumes continue to increase. In the U.K. the present system of motor taxation allows unlimited use of the road network, irrespective of the amount of road usage. The introduction of a system which charged for road usage was raised by the Smeed Report [Smeed 1964] and includes varying the charges at different times of the day and charging higher rates for more congested routes. It argues that this approach would deter travel at peak hours on very busy roads and therefore alleviate congestion; furthermore it ensures that road users are responsible for the cost of their road usage, including the public costs of building and maintaining the road infrastructure and the social costs of accidents and pollution.

Most systems used for road-use pricing require the vehicle to have equipment which identifies it to an inductive loop detector in the road. Experiments were performed in New York in the late 1970's [Foote 1981] but the results indicated that the system was not able to recognise vehicles reliably enough. The Hong Kong electronic road-pricing scheme, [Clancy, Dawson et al. 1985] was a larger scale experiment. It was found to perform well but the cost of maintaining the inductive loops and the associated communications equipment was too high for a commercial system to be installed. Most notable among the more recent work in this area has been the HELP (heavy-vehicle electronic licence plate) programme in the USA [Davies and Ayland 1989].

A common way of charging for road usage is to have tolls on major routes. This can cause serious congestion. To alleviate the problem the use of unstaffed tolls have become increasingly common in particular in the USA [Hills and Blythe 1989]. The system requires the driver to throw coins into a basket which has

automatic coin validation machinery. The system is often backed up by video equipment which takes a picture of any vehicle which does not pay the fee. In Bergen, Norway, there is a toll system where some vehicles are not required to stop and pay at the toll, but may drive through 'non-stop' lanes providing that they have a pass which is prominently displayed in the windscreen. These lanes are monitored by cameras to detect offenders.

The first commercial automatic tolling system was introduced at the end of 1987 in the Alesund tunnel in Norway [Gosch 1988] which uses a micro wave transponder mounted within the vehicle.

In Lyon, France in 1988 an experimental video image analysis system was tested, however, the error rate was found to be unacceptable for pay-toll applications.

A system capable of reading the licence plate of vehicles as they pass has several advantages over other systems of vehicle identification. Vehicles do not need to be fitted with any special equipment and photographic evidence is available if there is any dispute over payment of the toll. Drivers may attempt to avoid paying tolls by disabling their transducers, therefore most systems of this type also have a back-up system of video cameras to photograph vehicles if the transducer information is invalid or absent. Currently these back-up systems are simply viewed by a human operator. A system capable of reading the licence plate would therefore be of great value even it was not the primary method of vehicle identification.

The reduced cost of computer hardware and in particular image processing equipment [Taylor 1988] greatly improves the cost effectiveness of optical recognition systems. The introduction of stereo vision systems offers the potential for improving the performance of these systems [Al-Dabass 1981,1985]. In addition the development of new algorithms for the recognition of characters

in unconstrained environments will continue to improve the viability of vision processing systems.

1.1 The Pattern Recognition Problem

Since the early 1960's much research has been directed at understanding digitized images [Suen 1986]. As yet there are no techniques which will enable an unconstrained image to be presented to a computer system and a description of the image in terms of the objects it represents to be output.

The aim of a recognition system is to identify objects in an image. Humans perform this process with such ease that it is initially surprising that this problem is so complex. A major difficulty is the quantity of data present in a single image. A resolution of 512×512 pixels, with each pixel having an 8-bit intensity values is typical. The processing of such large quantities of data is time consuming even with multi-processor systems; however, this does not imply that given sufficient computing resources, the problem can be completely solved. It is difficult to produce a set of criteria which defines an object such that it can be readily distinguished from its surroundings, if the object size, position and orientation are variable and also the background is unconstrained. If an image is monochrome and its intensity is similar to its background, it can be very difficult to partition the scene. The intensity variations caused by shadows or by features which are internal to an object can easily yield greater intensity differences than the edges of the object itself. If an object has to be identified in a variety of images its pixel intensity will vary due to lighting conditions.

The partitioning of objects solely on the basis of their pixel intensity does not yield good results except in very controlled environments. This has led to the development of a wide range of locally adaptive thresholding algorithms [Giuliano 1977, Marr and Hildreth 1983, Palumbo 1983], which classify pixels not only on the basis of their own intensity but also the intensity of their

neighbours. There is a recurring conflict in thresholding algorithms of this type. If the algorithm is to be sufficiently sensitive to extract small edges, it may identify too much detail and obscure the objects it is trying to find. Conversely less sensitivity may neglect important features. To cope with this, most algorithms have constants which allow the algorithms to be tuned to the specific image.

1.2 Character Recognition Systems

Although general purpose pattern recognition systems are not available, there are some areas of notable success. Optical character recognition systems have been commercially available for over ten years and are becoming increasingly common. The cost of hardware has fallen and the performance of such systems has increased as the input devices and recognition algorithms improve [Jones 1989, Eager 1990]. Such systems achieve their results by highly constraining the input data. Input to an OCR system is usually required to have clear black letters of a known font on a white background with lighting conditions, size and orientation carefully controlled. This simplifies the problem of extracting the characters from the background.

After isolating individual characters, they must be interpreted. This is performed readily by many OCR systems when the input image is highly controlled. The system knows the size and orientation of the characters, which must have sharp edges and conform closely to the idealized shape for that particular font. OCRs are usually unable to read less constrained text, for example FAXs. Most character recognition systems use a simple comparison between the character to be identified and a series of templates, which represent idealized forms of the character set. This works well for clear highly constrained images but may require substantial effort to 'normalize' the character for size, position and orientation, prior to the comparison. Many OCR systems do not perform normalization.

Recognition of hand written script is only possible if the system is trained to recognise a particular person's writing or if individuals are trained to write in a standard manner [Himmel 1976].

1.3 Thinning Algorithms

A major difficulty with recognizing characters is that patterns with widely differing shapes and attributes are classified by the human observer as representing the same character. Foremost among these attributes, is the thickness of the character strokes. A wide range of thinning algorithms is available which produce skeletal forms of patterns, for example, Rutovitz [1966], Hilditch [1969], Yokoi [1973], and Tamura and Mori [1978]. It is surprising that different algorithms produce different skeletons. This is partly explained by the fact that some methods produce '4-connected' skeletons and some '8-connected skeletons' [Yokoi 1973]. Two pixels are said to be '4-connected' if they are orthogonal. 8-connected pixels are a superset of '4-connected', and includes adjacent but non-orthogonal pixels. However, this does not account for all the differences between skeletons, since a variety of thinning criteria is used. There are also several problems with most of the algorithms. In particular they exhibit a high degree of sensitivity to minor variations in pattern boundary. A single pixel difference between patterns can lead to very different skeletons. Some algorithms produce a flaring at the end of strokes. In addition the relative thickness of the strokes which make up a pattern affects the output. The output is invariably very different from that which is produced by a human observer. This limits the usefulness of these algorithms.

1.4 Syntactic Pattern Recognition

An area of research which has been neglected in recent years is the syntactic approach to pattern recognition. In this approach a 'pattern grammar' is

developed which defines primitives, usually in terms of lines or patterns of known size and orientation. The earliest work in this field was done by Freeman [1961], who expressed the boundary of a pattern in terms of a small set of concatenated primitives. However, serious problems have been encountered since the primitives chosen have a specified orientation and length. This leads to difficulties if the pattern to be recognized has an unknown size and orientation. Difficulties are also encountered when the primitives have to be combined in ways apart from concatenation. The use of high level pattern grammars allows the primitives to be combined in ways apart from concatenation [Fu 1986].

The syntactic approach has been used successfully for several applications including the identification of chromosomes [Ledley 1964], the identification of roads from satellites [Keng and Fu 1986], and the recognition of the alphanumeric character set [Ali and Pavlidis 1977].

A syntactic approach is used in this thesis for identification of the skeletal forms of the characters extracted from vehicle licence plates.

CHAPTER 2

CURRENT TECHNIQUES

CHAPTER 2

CURRENT TECHNIQUES

Several areas of work are relevent to the recognition of characters in unconstrained environments. The areas considered are:

- OCRs. These are widely available and have a high level of speed and accuracy; however, they only operate in highly constrained environments.
- Shape description. A wide variety of algorithms are available for describing the boundary of patterns.
- Edge detection. Edges often indicate the boundaries of objects.
- Image Segmentation. This section is concerned with the partitioning of the image into objects, and in particular includes image binarization.
- Thinning. This is the process of obtaining skeletal representations of patterns and is most useful in extracting structural information about them.
- Syntactic pattern recognition. This is a collection of techniques which allow a pattern to be expressed as a collection of spatially related primitives.

In addition, a recognition system which has been implemented using existing techniques is described.

2.1 OCRs

The idea of a machine that can read text can be traced back to the 1960's [Suen 1986]. Initially it was thought that the problem would be readily solved by designing a system to recognize characters based upon simple descriptions of character archetypes. However, many unanticipated difficulties were found and

the ideal of a machine capable of recognizing both characters printed in any font and handwritten text has not yet been realised. Character recognition systems which are able to read clear, printed text have been an area of notable success and such systems are becoming more readily available.

Over a thousand type faces are in common usage and can be readily read by humans; many of them have significantly differently shaped characters. To cope with this difficulty, a recognition system which has to deal with a range of type styles needs a set of descriptors for each font which is it is to read. Commercial OCRs are now available but these only work successfully when the image presented to the system is highly constrained, that is, if the orientation of the text, its size, and font are known. To improve their performance, standard fonts have been developed such as OCRA by ANSI and OCRB by ECMA. These allow a recognition accuracy as high as 99.99% at speeds of over 100 characters per second [Suen 1986]. The reading of handwritten script is still a topic for research as is the recognition of characters in variable environments where the font, size and orientation of the characters is not known. While the development of OCRs has been a notable success, their performance is subject to important limitations. The patterns corresponding to the characters are extracted from the input image. Typically a character size of 8 pixels wide by 10 pixels high is used. After identifying the individual digitized characters, the image is smoothed in order to eliminate random noise and voids. In some cases the image is normalized in size and orientation. The recognition process usually takes one of two forms. The first is the comparison of the image against templates of idealized characters. The second is the extraction of features which are compared to a database. The operation of a typical OCR is described in detail by Suen [1982].

Many manufacturers such as Panasonic, Hewlett Packard and Logitech produce systems which are capable of recognizing a wide range of fonts. In recent years

the cost of commercial OCRs has greatly reduced. Jones[1989] and Eager[1990] report on OCR software which runs on standard 80286 and 80386 based PCs. The systems works with a wide range of fonts and achieve success rates of over 99% on clear text. The time taken to read a single sheet of A4 text is under a minute using an AT PC computer. It should be noted that such systems are not effective on text which is not very clear and they are unable to read typical FAX messages.

Handprinted Script: The recognition of handprinted characters presents a greater problem due to the wider range of character shapes. The human observer interprets about 5% of characters incorrectly when reading hand written text if the context of the character is not given [Suen and Mori 1982].

The set of shapes which a human readily identifies as representing a particular character differ widely from each other. This means that a single description of an individual character which describes the entire set of shapes which we would recognize as representing that character cannot be developed. In addition there are human perceptual attributes which affect our interpretation of ambiguous characters. Blesser and Shilman [1973] address this problem by considering the existence of three sets of attributes for each character. They are:

- Physical attributes, that is, the geometric parts which describe the character.
- Perceptual attributes, the parts perceived as being present by the observer.
- Functional attributes.

The distinction between them is illustrated diagrammatically in figure 2.1:

The first character is open in a functional, perceptual and physical way. The second character has a small gap, the pattern is functionally closed, that is, it is more often reported as representing 'O' than 'C'. However, it is apparent to the

observer that there is a gap. The third pattern has a small gap which is usually not perceived, that is, it is functionally and perceptually closed but physically open. The fourth pattern is closed in all three senses.

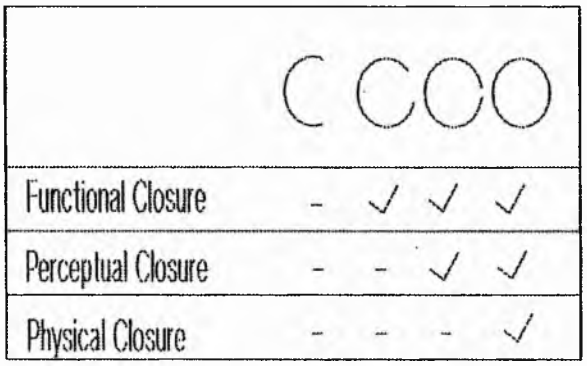


Figure 2.1 Closure Types

In the case of ambiguity the only 'correct' interpretation of a pattern is the one agreed by human observers. In practice, humans base their decision on the context of the pattern as well as its shape.

The most straightforward technique for the recognition of hand written script is simply to compare the patterns against templates of idealized forms of the character set. Most systems of this type require the user to submit a sample of handwriting [Burr 1980]. In these circumstances the performance is good. However, it is an inadequate method if the system is used as a script recognition system for many different individuals.

In addition to simple template comparison, other methods have been developed for reading handwritten script. Two of the most popular sets of techniques are:

- The contour description approach.
- The boundary approximation approach.

These techniques are variants of the general purpose techniques which are used for pattern description and are discussed in section 2.2 on shape description.

Contour Description: These methods operate by following the boundary of the shape and expressing it in terms of a series of primitives, each of which has a defined direction. Among the earliest propounders of this work were Freeman[1961] and Eden[1962]. More recent work on the application of the

contour description approach has been carried out by Toussaint[1970] and Fu[1980a]. This set of techniques is not limited to the recognition of the Latin character set and has been successfully used by Badie and Shimura[1980] to recognize Arabic cursive script. The Arab character set has a much higher proportion of curves than straight lines compared with the Latin set. This makes the recognition of Arab script more difficult . The transition of a straight line into a curve, for example, where the vertical stroke of 'D' meets the curve part of the character, is more easily identified than the partitioning of the letter 'S' into sections. In the technique suggested by Badie and Shimura[1980], the contour following starts at an end of the shape. The direction of the curve is taken as the direction of the line between the end and successive points on the contour. The direction is one of eight lines each at forty five degrees to each other as shown in figure 2.2.

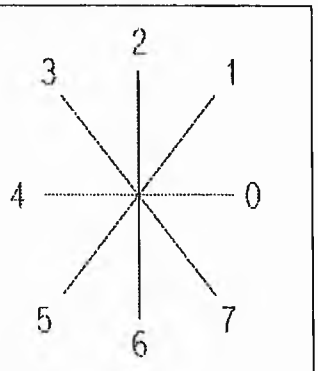


Figure 2.2 Contour Primitives

The character in figure 2.3 has its characteristic points marked.

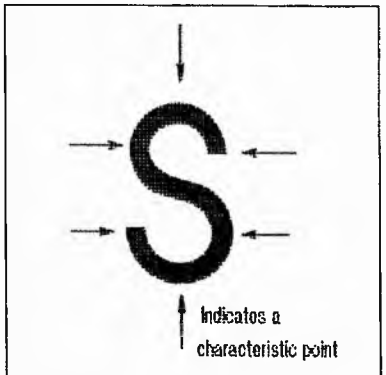


Figure 2.3 Transition Points

The point at which a transition from one of the eight directions to another is a characteristic point on the contour. When a new characteristic point is found, the process is repeated until the end of the shape is reached. Shapes which consist of a single continuous series of strokes can be expressed as a sequence of branches of known direction. Shapes which have intersecting strokes also require a connectivity matrix which specifies the spatial relationship between the lines. On the test data used, a recognition rate of over 90% was achieved. The characters are represented by a 16×16 matrix of pixels, which although not

reported is an indication of a problem with this method. The usual pixel matrix is 10×8. However, this does not provide adequate resolution for curve partitioning. Since the representation of the character is composed of discrete pixels rather than being a continuous curve, problems can arise. If, for example, a single vertical line is considered, this can be represented in a variety of ways due to rounding errors in the generation of the character. Each of the four lines below could all be digitized representations of the same vertical line.

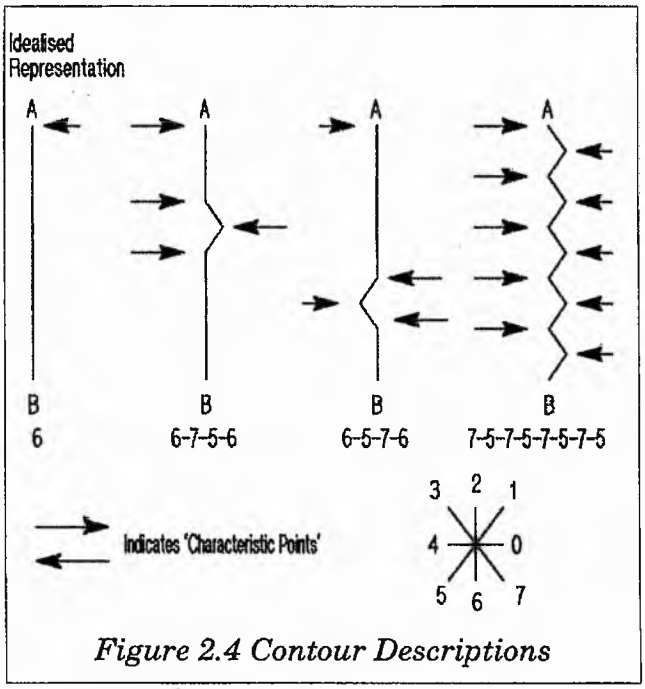


Figure 2.4 Contour Descriptions

Point 'A' is the starting point.

The four widely different representations may result. In the second and third lines, the effect of the single offset pixel is very different due to its position relative to the starting pixel 'A'. There are other difficulties with this approach. It is best applied to skeletal representations of characters. However, the term skeleton needs careful definition and is dealt with at length in section 2.5. It is possible to derive different skeletal representations of the same pattern which can yield different characteristic points and therefore different interpretations of the same character.

Boundary Description: A widely used alternative to contour description is boundary description by polygonal approximation. An ordered list of the boundary points is produced and then points are broken into a series of curves using the polygonal approximation approach first described by Pavlidis and Horowitz[1974]. The approximation technique makes the extraction of many features about the shape straightforward, including the number of holes, the

number of concave arcs on the external boundary and a variable describing holes; and so on. An exhaustive list and full explanation is given by Pavlidis[1975]. The identification of the characters is done by matching the extracted topological information about the shape against known attributes for each character. A similar approach is used by Yamamoto and Mori[1978] who describe the boundary in terms of a series of concave and convex arcs.

Centroid Lines: A variety of less widely used techniques has been developed which do not fit into either of the above categories. Naito[1978] suggests the use of centroid lines, where the character is divided into thin horizontal sections and the centroid for each section is calculated. The group of centroid points obtained is characteristic of the character. Naito and Hagita[1983] implemented an OCR for Kanji characters using three characteristic features. Firstly the stroke density function is produced by moving a horizontal line vertically down the character and counting the number of strokes which intersect it. Secondly the direction contributivity(sic) density function gives the direction of each stroke. Finally the peripheral direction contributivity function is derived by scanning horizontally across the character and calculating the component of the strokes which are intersected in eight 45 degree directions. The technique achieved a high degree of success. The system was firstly presented with characters by unknown writers and secondly with a number of characters by a specified author. The recognition rate was significantly higher for character written by known author.

2.1.1 Performance

The performance of handwritten OCRs can be improved in several ways. The problem of finding the text can be simplified by having either preprinted marks on the paper or by requiring the user to enter each character into a separate box. The performance of these systems in identifying the characters can be

improved in two ways; firstly by personalizing them, that is training them to recognize a particular writer and secondly by training writers to draw their characters in a more regular and consistent manner.

Large scale surveys conducted in the 1970's on the performance of commercial OCRs illustrate some of their limitations. Himmel[1976] gives details of the performance of OCRs on documents produced by the Netherlands PTT and the Australian Post Office. The characters read in the survey were handprinted on documents which had defined areas for each character; this highly constrained their size and orientation. There were 3000 characters in the analysis, 90.0% were correctly read, 4.5% were rejected, and 5.5% were incorrectly identified. It was estimated that half of the rejected characters could be successfully identified by improvements in the OCR algorithms. Improvements were obtained by using pens with a more consistent performance. Ball points which produced ink blobs caused many errors.

The most significant improvements in performance are obtained by training the users who wrote the characters which were to be read. Himmel[1976] reports on a study conducted using two groups of employees from the U.S Social Security Administration; one group trained in writing legible characters, the other group untrained. The trained group's characters were read with a success rate varying from 98.5% to 99.5% depending on the recognition used. The untrained group's success rate varied between 82% and 90%. This clearly illustrates the importance of controlling the characters which are submitted to the OCR if a high success rate is to be obtained. The benefits of training are further demonstrated by a system developed by D'Amato[1982]. The recognition system he produced for reading handwritten characters generates a compact description of the character boundary in terms of eight geometric features. The encoding is compared to a database of encodings which represent known characters. The system was tested using about 15,000 unconstrained handprinted characters

and gave a correct recognition rate of 73%. On a sample of characters produced by trained persons a figure of 96% was obtained.

Pavlidis[1983] considered the effects of character distortion on the recognition rate of an OCR system. The system used was able to recognize nine different fonts and obtained a recognition rate of 99% on characters written in these fonts, this fell to 98% and 95% for two character sets which the OCR system had not been specifically setup to recognize. The effects of distorting the characters in the known fonts by about 5%, firstly in the horizontal and secondly in the vertical direction had little effect on the recognition rate which fell by under 1%.

These figures compare badly with the extremely high accuracies obtained for OCRs reading the special OCRA and OCRB fonts. An accuracy of 99.99% is achievable [Suen 1986].

This section has concentrated on the algorithms used to recognize digitized characters. However, it should be noted that the scanner and the algorithms used to digitize the image are of equal importance. A scan density of 500-1000 lines per inch is commonly used for image digitization; a lower density may result in insufficient resolution. The digitized characters are in binary format, that is a black character on a white background; the interface between the background and foreground is a slope whose intensity varies from black to white. Therefore the selection of an appropriate threshold for conversion to the binary form is most important and is dealt with at length in a section 2.3.

2.1.2 Comments

The development of OCRs has been an area of success in the field of pattern recognition. Commercial OCRs are becoming increasingly common as their price falls and their performance improves. Significant advances have been made in

the area of reading hand printed script and good results can be obtained if operators receive training and pre-printed forms are used to direct the OCR to the character. However, there are significant limitations; for example, the character must be well defined and of known size and orientation. In practical terms this is straightforward to achieve as more text is produced by word processor and the price of high quality printers is now low.

2.2 Shape Description

When a recognition system is presented with a digitized image it must separate the image into components and compare each component, or a set of component attributes against a database of the set of patterns the system can recognize. The simplest case consists of a single two dimensional object which has binary pixel intensity, that is, each pixel is designated as either black or white. If the orientation and size of the object are fully controlled, it can be identified by comparing it to a series of templates which are known objects. However, even in this simple case it may be more efficient to compare object boundaries which can be stored as a list of orthogonal co-ordinates. Such a list is a representation of the object but allows the object itself to be recreated. The comparison between object representations and a database of a set of representations of known objects is potentially a far more powerful technique than the manipulation of the objects themselves for object identification. The development of techniques for storing a representation of an object without actually storing the object itself has been a recurring theme over the past thirty years of pattern recognition. It is closely related to problems of finding the closest match to an object, as exact matches are rare due to the process of digitizing the images. This always produces variations in the intensity values of pixels and differences in edge sharpness if there is even a slight variation in lighting conditions or camera position. Two dimensional shapes are the simplest to represent, particularly regular geometric patterns. For example, a

square can be completely represented by simply storing its side length, although a co-ordinate frame has to be setup in order to define its orientation. The problem quickly becomes more complicated for the generalized case of an irregularly shaped object. A range of techniques has been developed to deal with this. The techniques considered in the next section are reversible, that is the actual shape itself can be recovered from the representation of the shape.

2.2.1 Chain Codes and Shape Numbers

Freeman [1961] was among the earliest researchers to suggest breaking the outline of two dimensional shapes into small sections and representing each section by one of a small set of primitives. The shape itself is stored as the concatenation of these primitives. The smaller the primitives, the more accurate is the representation and the longer the string. The primitives used by Freeman are four lines each at right angles to the other. A major problem with this technique is that it is orientation dependent, that is different chains can be obtained for the same shape in different orientations. There is also the problem of determining if two shapes are similar when their Freeman chains are of different lengths.

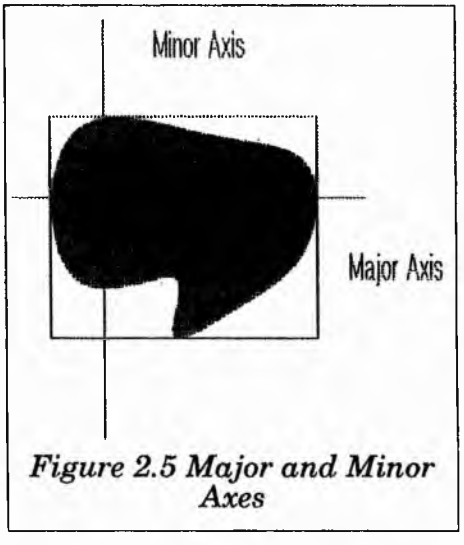


Figure 2.5 Major and Minor Axes

These problems have been addressed by Bribiesca[1979] who devised a technique for the derivation of a unique shape number. A 'basic rectangle' is derived for the shape and the direction of the four primitives are along the major and minor axes of the rectangle. The basic rectangle is the smallest area rectangle which wholly encloses the pattern. The major and minor axes are respectively parallel to the long and short sides of the rectangle.

By allowing the user to specify the order of the shape number, that is the number of digits in its encoding, the problem of comparing different length shape numbers can be overcome. However, in order to derive a shape number which is unique, for a pattern, a few further rules must be followed; for example, moving around the shape in a clockwise direction and selecting a starting point which gives the smallest possible shape number. A similar system is used by Badii[1983] who describes a chain code based upon the topological attributes of the pattern perimeter, which is invariant under rotation, translation and scaling.

2.2.2 Quadtrees

Chain codes and shape numbers concentrate on expressing the boundary information of objects; More recent techniques have concentrated on expressing the area of the object by hierarchical data structures.

A popular technique for the compact representation of two dimensional shapes is to use quadtrees.

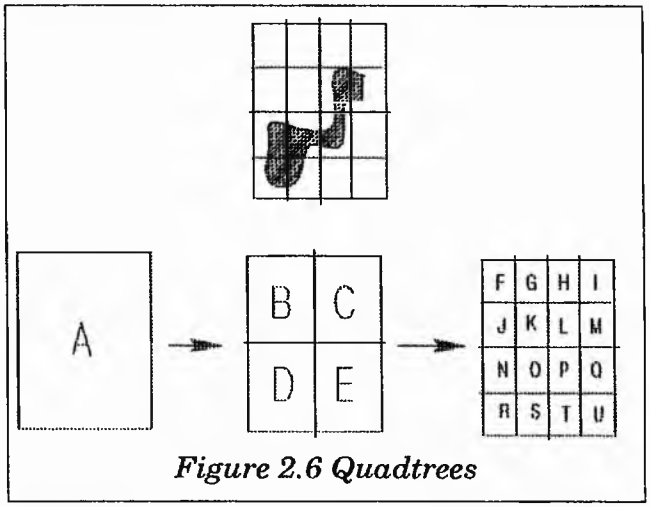


Figure 2.6 Quadtrees

Early work on this was done by Sidhu[1972] and Tanimoto[1975]. A quadtree is generated by dividing the image region into quadrants. If a quadrant contains a portion of the object, it is further divided into sub-quadrants. The process can be repeated, resulting in a hierarchical data structure representing the shape.

There are problems similar to ones arising from the use of shape numbers, in particular, the quadtree representation is heavily influenced by the position of the object in the image, its relative size and its orientation.

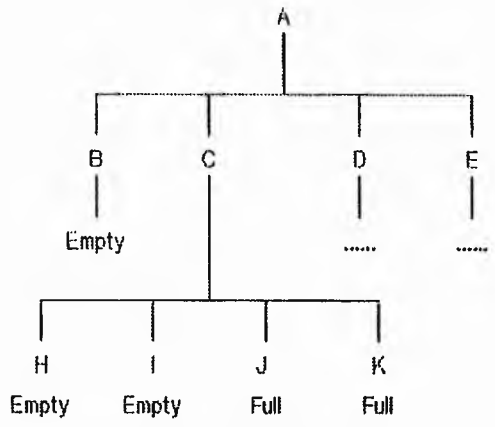


Figure 2.7 A Data Structure for Quadtrees

Normalized Quadtrees: To deal with the problem of the position of the object in an image, Li, Grosky and Jain[1981] suggested a normalized quadtree form. However, the problems of rotation and size were not resolved.

A normalized quadtree approach which deals with these problems was put forward by Chien and Agrawal[1983]. Prior to generating the quadtree some pre-processing takes place. A co-ordinate frame must be defined and the object must be scaled. This is done by using two properties of shapes which are independent of location and orientation, the centroid and the principal axes. The object is rotated about its centroid until its principal axes are horizontal and vertical, relative to a defined co-ordinate frame. It is then enclosed in a minimal square whose sides are parallel to these axes. The bottom left corner of the square is defined as the origin of their co- ordinate frame. The image within the square is now scaled to a standard size. This normalized form is independent of size, object position within the image, and orientation. The processes of re-orientation and scaling are common for representational methods of this type; however, they introduce problems of their own. The rotation process is usually achieved by multiplying the orthogonal co-ordinates of every pixel of the object by the direction cosines of the new frame with respect to the old one. This causes a distortion in the image as well as being time consuming

for large areas. The scaling process also causes distortion. If the ratio between the size of the normalized image and the original is a power of two, the distortion is minimized. If it is not, extrapolation must be used; for example, to scale a square of side 1.5L into a square of side L.

A system was implemented to examine experimentally the effects of normalization on a range of images of alpha-numeric characters. This is reported in section 2.7. Two methods of normalization were used:

In the first only the outline of the object is considered for maximum speed. In some circumstances a connected outline will not be produced after normalization and it is necessary to extrapolate between pixels when holes are found.

The second method rotates the entire image. A different problem is found here. An object which is solid, that is contained no holes prior to rotation, may have several small holes after normalization due to rounding errors in the normalization calculations.

The normalization process distorts the object, deleting valid information from the image and introducing invalid information.

2.2.3 Rectangular Codes

A novel variation is the use of sets of elementary shapes to describe the object area. Aoki[1979] suggested that this could be done by using circles, triangles and rectangles. The circle requires the least information of these shapes to define, only a centre and radius, while rectangles require the X and Y co-ordinate of one vertex, the width and the height. Circles are difficult since a set of non-overlapping circles always leaves gaps between them. The generation of a set of rectangles is non-deterministic, that is there is more than one set of rectangles which describes a shape. In practice this is not a problem as long as the same algorithm is used for encoding all the objects considered. The use of

rectangles has been explored by Kim and Agrawal[1983] who present a heuristic algorithm for constructing a set of rectangles which they call a rectangular code. This set describes the object completely. Only a single pass through the image is needed to generate the code. Starting in the top left corner the image is scanned horizontally and rectangles are grown vertically downwards; when a part of the region being scanned borders two growing rectangles, a simple set of heuristic criteria is used to determine which rectangle is extended and which is terminated. An example of an image partitioned in this way is shown 2.8.

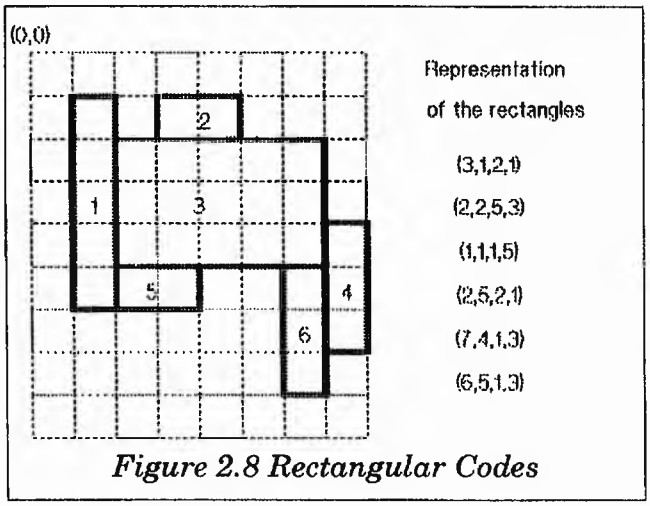


Figure 2.8 Rectangular Codes

Each rectangle is identified by four pieces of information shown in brackets in the figure; the two co-ordinates of the top left corner, the width and the height. Kim and Agrawal suggest that a compacting of this information can be achieved by grouping rectangles with similar properties and recommend grouping where either the height or width of the rectangles is similar. The information can be encoded by noting the four parameters which denote the first rectangle in the list, the number of rectangles in the group, the difference in the position of two consecutive rectangles in the group and an indication of the parameter which is the same.

Scaling by a power of two can easily be achieved with this representation. This technique is not ideal as, there is the usual problem of orientation. Rotation by multiples of ninety degrees is easily resolved since partitioning of the object will be the same, although this is not the case for rotation by other amounts. Even a shape which can be represented by a single rectangle in one orientation requires many rectangles if twisted by forty five degrees. The usual solution to

this problem is to use information invariant to position to define a co-ordinate frame and origin. This can be achieved in the same way as for quadtrees, that is by using the centroid and the principle axes.

2.2.4 Symmetric Axis Transform

A different approach to shape description has been developed by Blum[1973] which is designed to represent the shapes which occur naturally in biology. The work describes the symmetric axis transform (SAT). The SAT description of an object considers the set of maximal circles which fits within the object but not inside any other circle in the object. The shape description is in two parts:

- The symmetric axis. This is the locus of the centres of the maximal circles.
- The radius function. This is the radius at each point along the symmetric axis.

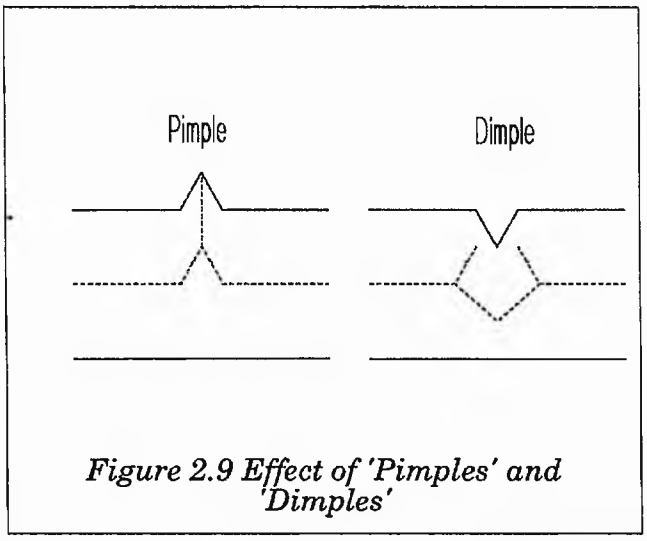


Figure 2.9 Effect of 'Pimples' and 'Dimples'

This early work was extended by Blum and Nagel[1978]. By partitioning the object at points where the symmetric axis divides, the object can be divided into 'simplified segments'. The relationship between them can be expressed by a directed graph and an associated data structure giving details of the properties of the segments. This approach is significantly different from the other shape descriptions considered so far. It is orientation independent and suited to expressing objects where the structural content is high. There are a number of difficulties which are not found in other techniques. The SAT description is greatly affected by slight bumps or

depressions in the object boundary which can lead to the creation of additional segments. An example of this is given in figure 2.9, where a slight difference in the border of a pattern leads to widely differing skeletons. The skeletons are shown as dotted lines.

A useful technique for extending the applications of SAT would be the use of high order pattern languages which are suitable for expressing shapes which have a high structural content.

2.2.5 Comments

This section has described a range of methods for describing shape. The Freeman chain codes and shape numbers describe the boundary of shapes while quadtrees and SAT deal with the area within the shapes. All apart from SAT suffer from the problems of scaling and orientation, while SAT cannot deal with shapes which have holes and is very sensitive to 'pimples and dimples' on the surface. It is clear that although the problem of shape description is a fundamental one, it is far from completely resolved. In three dimensions all of the problems of shape description are far more acute. In practical systems where it is necessary to recognize objects, the environment is often highly constrained and the system is simply required to differentiate between a few items whose size and orientation are known and where lighting conditions are controlled.

2.3 Edge Detection

The detection of edges is of great importance in partitioning scenes into discrete objects. In natural scenes the boundaries of objects tend to be characterized by discontinuities in the intensity of the brightness of the image. These discontinuities in the grey scale are edges, and due to their fundamental importance, a great number of algorithms have been developed for their

detection. All edge finding algorithms, by examining the properties of pixels over an area, search in a wide variety of forms for these discontinuities; in order to provide a fuller description of an edge, many algorithms derive both the magnitude and direction of any edges detected.

There are several difficulties in the detection of edges in real world images. Objects tend not to consist of collections of adjacent pixels of equal intensity with very sharp edges. Slight variations in lighting and surface imperfections, particularly for coarsely textured surfaces, result in a variation of pixel intensity within an object. Successful edge detection algorithms should therefore be able to detect both very sharp or 'step' edges and also edges which are broader, called 'roof' or 'ramp' edges. The larger the area which is considered by the algorithm, the better is the roof detection performance, while the smaller the area the better is step edge performance. It is important to be able to differentiate between a step edge and the local variation within an object caused by texture variation.

Since edge detection is of fundamental importance to low level image analysis problems, there is a great variety of algorithms available. Surveys are available by Davis[1974], and Rosenfeld and Kak[1982] and more recently by Kashyap[1986].

A useful categorisation of the types of edge detection algorithm is given by Ballard[1982], who divides them into three categories:

- Grey scale gradient based operators.
- Template matching operators.
- Parametric edge models.

2.3.1 Grey Scale Gradient Based Methods

First Derivative Methods: The earliest edge detectors developed are gradient based, such as the operator suggested by Roberts[1965]; for an image function f(x), the gradient magnitude s(x), and the direction d(x), are given by:

$$s(x) = \sqrt{(\delta_1^2 + \delta_2^2)}$$

$$d(x) = \tan^{-1}(\delta_1/\delta_2)$$

where

$\delta_1 = f(x + n, y) - f(x,y)$

$\delta_2 = f(x, y + n) - f(x,y)$

n is a small number, normally one.

The Roberts method is widely used. Its main advantage is that it is very fast. Its main disadvantage is that it operates on a small area and may therefore be confused by a noisy image; it is essentially based on the first derivative of the grey scales and therefore may respond erratically on a ramp intensity profile. This second problem has led to the development of second derivative methods. However, some improvement can be gained by using an absorption edge detector[Kittler 1983].

Second Derivative Methods: At a step edge the second derivative is zero, with a positive and negative peak on either side. The detection of edges is achieved by finding zero crossings of the second order derivatives. A common idea in the low level processing of images is the concept of image transforms, in essence an image may be transformed, then filtered or manipulated, that is convolved and subject to the inverse transform. Marr and Hildreth[1980] make use of this idea by searching for zero crossings in:

$f(x) = D2[G(r)*I(x,y)]$

D2 is the second derivative of intensity in the appropriate direction.

G(r) is the Fourier transform of the Gaussian distribution.

* is the convolution operator.

Convolutions are relatively expensive computationally and their number can be reduced by introducing an orientation independent second order differential operator. The Laplacian V2 is the only operator which has this property. The loci of the zero crossings can be found by searching for the zero values of the convolution V2G*I. Haralick[1984] compares the performance of the Marr-Hildreth with an edge detection method based upon the zero crossings of the second derivative and claims an improvement in performance.

2.3.2 Template Matching Methods

The use of edge templates is a popular method. A variety of templates representing ideal edges in various directions are matched against the image. The mask which gives the closest match indicates the magnitude and direction of the edge. This has been successfully used by Tamura and Mori[1978]. A major problem reported by them is that large masks are needed to reduce the effect of noise on images; 5×5 is regarded as a minimum. However, the larger the mask the more likely the output is to be confused if more than one edge is within the template area.

2.3.3 Parametric Edge Models

The final major category considered is the operators which fit local image intensities with parametric edge models; a brief overview is given by Nevatia[1980]. One of the earlier models which fits this category is the Heuckel operator. This assumes that edges are step functions and describes edges in terms of a straight line and two image intensities on either side of the edge.

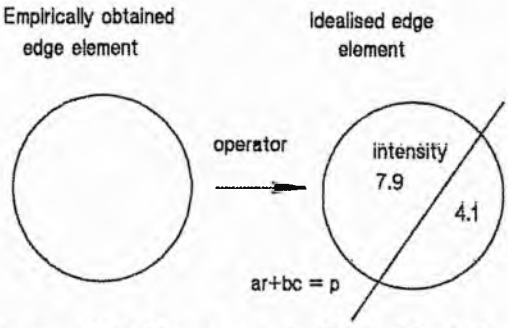


Figure 2.10 Parametric Edge Models

A set of pixels which most nearly approximates to a circle is selected and the equation of the edge and the intensities of the image of both sides are calculated as described by Heuckel[1971]. There are a number of difficulties with the Heuckel operator. The most apparent is that edges are assumed to be straight lines; this is often not the case and this operator is not suitable for detecting corners. The model also assumes that only one edge is present in a circle; it is therefore easily confused by multiple edges, particularly with larger discs. This is a serious problem [Canny 1986]. A parametric edge model is developed by Tretiak[1979] in which the edge is specified by a curve described by a pair of parametric equations, and the edge detection problem is resolved by minimising a cost function over the set of curves.

2.3.4 Comments

The role of edge detection is to partition an image into sections, which ideally delimit objects. In practice this is very difficult to achieve. The more complex methods such as the parametric edge detector do not necessarily yield better results. When choosing between the range of methods available it is important to consider the application for which it is to be used. Generally, models which look at a small area are able to detect fine edges but are very sensitive to noise in the image. The reverse is true for models which examine a large area. It is therefore important to carefully balance these conflicting requirements.

2.4 Image Segmentation

Another low level operation used in image understanding is to segment the image into regions based upon the properties of the pixels in the image. Although this is a straightforward process, it is extremely difficult is to ensure that these regions correspond to physical objects within the scene. The problem is complex since it does not have a unique solution [Fischler 1983] and therefore a purely mathematical criterion is insufficient to duplicate human performance. Segmentation can be based either solely on the intensity of individual pixels, or can include the intensity of local pixels. It is essential to consider regions when performing a texture analysis of the image or taking into account the effect of changes in shade. It is sometimes possible to perform the image segmentation recursively using information gained in earlier segmentations in order to improve the performance as suggested by Ohlander, Price and Reddy[1978]. A popular set of techniques uses region growing where small sets of uniform intensity are used as the basis of regions. Neighbouring pixels are then grouped into the region based on their properties. In many images there is likely to be a close correspondence between the region interfaces and the edges obtained by edge detection algorithms. Milgram[1979] uses a variety of threshold values to segment images, comparing each one with an edge map of the image. The segmentation which gives the closest correspondence with the edge map is taken as the optimum. A useful discussion of pixel classification techniques is given by Rosenfeld and Kak[1982]. A survey of region growing techniques is given by Zucker[1976]. Pavlidis[1977] gives a detailed report on segmentation techniques and includes a variation on the usual techniques of segmentation originally suggested by Pavlidis and Horowitz[1973] which uses a split and merge technique. The image is split into regions which can be either merged on the basis of their properties, or split if they are insufficiently homogeneous. Relaxation techniques have been applied to image segmentation by Zucker[1977] and Boyle[1988] with some success.

In the general case of an unconstrained image which is segmented into sections the results are usually not good, as there is often an incomplete or conflicting subdivision of the image into regions which do not correspond to objects. The performance of segmentation algorithms can be greatly improved by two methods. The first is to increase the amount of information in the image either by introducing colour, or by using stereo information. The second commonly used technique is to tailor the segmentation process to the specific application by the use of a-priori information. This can take the form of constraining the input image either by reducing its complexity or by ensuring that the objects to be recognized stand out from their surroundings by giving them or their backgrounds special colours. Image segmentation techniques are often supplemented by the use of heuristics, based upon experimental data collected about the types of objects likely to be found in the images analysed.

A recent implementation of an image analysis and segmentation technique used for identifying rectangular address blocks on mail items technique is given by Srihari[1987]. It is essential in such a system that one of the segments detected corresponds to the address block. To verify that the region has been correctly identified, a tool-box approach is used, where a variety of checks are invoked to test the validity of earlier tests and so increase the likelihood of success. The tools include colour thresholding, intensity thresholding, texture discriminators and shape analysis which measures how rectangular a region is. The author reports a success rate of 86% for letters. However, even in this relatively controlled environment, the processing time varies between seven and twenty minutes using single processor systems such as a Sperry-7000, VAX-785 and SUN-3 computers.

2.4.1 Binarization and Thresholding

A commonly used method of image segmentation is to binarize the image, that is, rather than assigning a pixel an intensity value, each pixel is considered as

either 'black' or 'white'. Binarization is particularly useful when dealing with images which consist of two discrete parts, for example black letters on white paper. However, even in cases of this type when the document is digitized, the edges are usually ramp edges rather than step edges, and the selection of a threshold value for binarization which gives as clear a picture of the characters as possible is essential.

A useful survey of document binarization algorithms is given by Palumbo, Swaminathan and Srihari[1986]. Kittler and Illingworth[1985] include a discussion of different categories of algorithms. There are two type of binarization, global and locally adaptive

a. Global Thresholding: In global thresholding a single threshold value is chosen for the whole image. This can be useful for certain special cases where a-priori knowledge allows a threshold to be chosen. Some images have a strongly bi-modal distribution of pixel intensities. This allows the pixels to be categorized into two distinct groups by the selection of a threshold in between the two concentrations of intensities. An alternative form of the global thresholding method is to divide the image into sections and to determine a global threshold for each section. This does not yield good partitioning of the image since, if a section overlaps an object, there may be no clear representation of the object, particularly if the thresholds chosen for the adjacent regions are widely different. To some extent this can be partly resolved by dividing the image in more than one way and comparing the output obtained by thresholding the overlapping sections. However, this is difficult to implement and greatly increases computational time. In addition to using grey scale variations, a range of techniques for the selection of a global threshold has been used, such as second order derivatives by Deravi[1983] and grey tone spatial dependency matrices as used by Haralick[1973]; both of these methods use information concerning the spatial relationship between intensity levels.

b. Locally Adaptive: In locally adaptive thresholding the conversion of a pixel to either black or white is dependent not only on its own intensity, but also on the intensity of neighbouring pixels. A variety of locally adaptive thresholding techniques has been reviewed by Palumbo[1986].

The three methods given below are by no means an exhaustive list of adaptive thresholding techniques but represent typical examples of their type.

i. Running Average Thresholding: An efficient single pass adaptive thresholding algorithm was developed by White and Rohrer[1983]. For a scan line represented by the array l(n), the horizontal running average y(n) can be expressed as the sum of the previous running average value and a fraction of the difference between the current grey pixel value and the previous average value. That is, y(n) for each row up to column n is given by the previous running average and a weighting function:

$$y(n) = y(n-1) + f(l(n) - y(n-1))$$

Here y(n-1) represents the weighted running average for the current scan line up to and including the previous point. The weighting function has a value between zero and the value of its argument. The vertical running average is given by:

$$z(n) = z(n-L) + g(y(n) - z(n-L))$$

Here z(n-L) represents the vertical running average calculated using the previous scan line of image data. Finally the decision as to whether a pixel is black or white is given by:

if $h(l(n-N)) > z(n)$ then BLACK else WHITE

h(pixel) is a biasing function and N is a constant.

ii. Contrast Measure Thresholding: An adaptive thresholding algorithm was developed by Giuliano[1977] which is now a U.S. patent. Five neighbouring regions influence the threshold value used for a particular pixel. The regions

consist of a 3×3 template around the pixel under consideration and four diagonally opposite 3×3 regions.

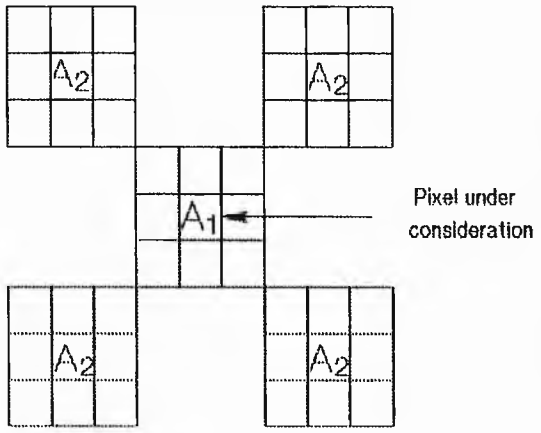


Figure 2.11The Giuliano Algorithm

Two static thresholds t_1 and t_2 are chosen. If the pixel intensity is less than t_1 it is changed to black. This simply reduces computational effort for pixels towards the edge of the grey scale spectrum. The subset of pixels in A_2, whose intensity exceeds t_2 is found, they are denoted by A_{2t}. If the average intensity of pixels in A_{2t} (weighted by the parameters t_3, t_4, t_5) exceeds the average intensity of pixels in A1 then the pixel is considered black. Formally the algorithm is:

if ($l(x,y) < t_1$) then Z(x,y) := black else

begin

if ($t_3 \times a_2 + t_5$) > ($t_4 \times a_1$) then Z(x,y) := black

else Z(x,y) := white

end

a_1 := Average of 9 pixels in area A_1

A_{2t}:= {(x,y)l(x,y) in A_2 and $l(x,y) > t_2$}

a_2 := Average of pixels in A_{2t}

An example of a practical implementation of this algorithm is used by Srihari[1987] for the identification of address blocks on mail items.

iii. Edge Based Thresholding: Marr and Hildreth[1983] suggest a binarization technique based upon the Marr-Hildreth operator[1980] which is used for edge detection. The method convolves the image with the Laplacian of a

Gaussian operator. The zero crossing points of the function, mark intensity changes in the image. Since the positive part of the function represents the side of the edge where the intensity is high and the negative part of the function represents the side where the intensity is low, the function can be used as a thresholding criterion. The positive values are assigned to white and the negative to black.

A limitation of the Marr-Hildreth approach is that it is very time consuming if a large operator size is chosen. Palumbo[1986] indicates that for a 31×31 operator running on a 512×512 image, $512^2 \times 31^2$ additions and multiplications are required and would take about 200 minutes CPU time on a VAX 11/780. This compares to about 2 minutes CPU time for the contrast measure binarization. The running average method is faster still. A further problem with second derivative binarization is that thick black regions get converted into black regions with small white holes inside. This problem also occurs with the contrast measure method.

2.4.2 Comments

Running average thresholding is the fastest and yields good results, but in noisy environments is inferior to the other two methods which are computationally far more expensive. In a test reported by Palumbo[1986] on binarization of a mail item represented by a 512×512 pixel array, the CPU time on a VAX 11/780 was 200 minutes for the edge based method and two minutes for the contrast method. The time taken for the running average method was not given. However, it was reported to be the fastest of the three.

The most straightforward method to implement is the running average method; it is also computationally the cheapest; unfortunately it tends to perform poorly with noisy images. The performance can be improved by having a pre-processing smoothing stage which reduces the amount of noise in the image.

The choice of method is dependent on the type of images used. There is no 'best' binarization algorithm; each of the methods available has its merits and drawbacks. The method chosen depends on the application. Generally the adaptive techniques perform better than global binarization; however, in some cases such as documents with dark letters on a light background in a controlled environment, global binarization with a carefully chosen threshold is most suitable.

2.5 Thinning

2.5.1 Thinning Criteria

A major difficulty in recognizing and giving a meaning to a pattern is often not that insufficient information is available for recognition, but that there is too much detail. In the recognition of an alpha-numeric character much of the information present is not relevant to the analysis. The set of shapes which represents a character have a variety of shapes but all contain similar structural information which allows their identification by a human observer. The thickness of characters is of little importance and can obscure the essential structural information. A technique which has been widely used to remove thickness information but to preserve structural detail is 'thinning' or 'skeletizing'. For simple shapes which are thin, that is long in relation to their width, there is an intuitive awareness of what constitutes a valid skeleton of the shape. To formally define which criteria are used and to provide an algorithm is less straightforward. Blum[1973, 1978] makes use of the idea of maximal circles. A maximal circle fits within the shape touching but not crossing its boundary at a minimum of two points. The locus of the centres of the maximal circles provides a thinned formed of the shape. Calabi[1968] uses analogy of a fire-line propagating from the edges towards the inside of the shape at constant speed. The points where at least two fire-lines meet constitute a skeleton. The

fire-line analogy is taken up by Arcelli[1981] who describes a parallel implementation of such an algorithm.

Many thinning algorithms use a set of templates which are applied to each border pixel of the image in turn, where a border pixel is one which is within the shape but touches at least one pixel outside it. A set of templates are centred on it, and surrounding pixels are matched against it. If a match is obtained then the pixel is deleted. A typical template set is given by Arcelli and Cordella[1975]. The templates must be applied repeatedly to the contour pixels until they can be applied to the whole image without deleting any pixels. Pixels left are skeletal. Depending upon the template set used, a few other simple criteria may need to be applied to prevent the shape being thinned to a single pixel. If for example a contour pixel is not deleted, that is, it is considered skeletal on one pass, it should not be deleted in subsequent passes.

Many of the earlier thinning algorithms were developed according to a set of heuristic criteria and did not include proofs that they would always work to produce connected skeletons. However, Several researchers have done this, such as Rosenfeld and Thurston[1971] and Arcelli[1975], so verifying earlier work on the conventional 'border following' methods. Many thinning algorithms have been developed by, for example, Rutovitz[1966], Hilditch[1969], and Yokoi[1973]. It is worthwhile comparing the output of popular thinning algorithms. A survey was carried out by Tamura[1978] which illustrates the differences between Rutovitz[1966], Hilditch[1969], Deutsch[1972], Tamura and Mori[1978], and Yokoi[1973]. The results show that the differences are substantial. This is initially surprising, since they are all aiming for the same result. However, on closer inspection images represented by rectangular arrays of pixels cannot always be thinned to produce a set of pixels which matches the maximal circles criteria. In a simple case of a vertical line four pixels wide, the theoretical skeleton is between the two central row pixels. Clearly a heuristic

decision must be made to decide which set of pixels is chosen to be skeletal. The position is more confused for complex shapes. In addition, while all the algorithms mentioned produce connected skeletons, the manner in which they are connected is different.

Four and Eight Connectedness: The concept of 4-connectedness and 8-connectedness is fundamental in explaining the differences in these algorithms. The eight neighbours of the element E_0 can be denoted as follows;

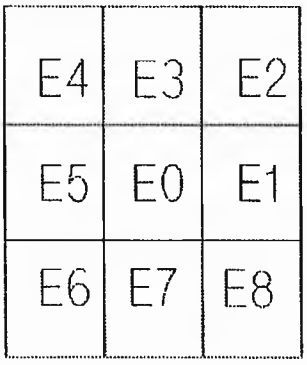


Figure 2.12 4 and 8 Connectivity

Elements E_1, E_3, E_5 and E_7 have 4-connectedness with E_0. All of the neighbours to E_0 are 8-connected. A 4-curve only contains points which have 4-connectivity. Rutovitz[1966] makes use of these concepts to define the crossing number X as:

$$X = \sum_{k=1}^{8} |E_k - E_{k+1}|$$

Here the value 0 is used for the background and 1 for the foreground. This gives a measure of the number of background-foreground chains in the 8 neighbours of E.

Yokoi[1973] defines measures of 4 and 8-connectivity in terms of connectivity numbers as:

$$N_{c4} = \sum_{k \in s1} (E_k - E_k E_{k+1} E_{k+2}) \text{ (4-connectivity case)}$$

$$N_{c8} = \sum_{k \in s1} (\underline{E}_k - \underline{E}_k \underline{E}_{k+1} \underline{E}_{k+2}) \text{ (8-connectivity case)}$$

$\underline{E}_k$ means $(1 - E_k)$

Where s1 is a set of 4-neighbours.

The properties of a point are characterized by the values of X, E_{c4} and E_{c8}. This has been investigated in detail by Yokoi[1973] and by Rosenfeld[1971].

2.5.2 Problems with Thinning Algorithms

Most thinning algorithms attempt to apply the maximal circles criteria by moving around the perimeter of the shape and considering each pixel in turn. However, it is not guarantied that this criterion is met using this approach, even when errors introduced as a result of the shape being composed of indivisible pixels are ignored. What is guarantied by all the commonly used algorithms is that an 8 or 4-connected skeleton is produced which lies wholly within the boundary of the original solid shape. On closer examination it is found that even if the maximal circles criterion was met fully, the skeleton produced is not always as anticipated by the human user. Consider the simple case of three thick lines intersection at right angles in the form of a 'T'.

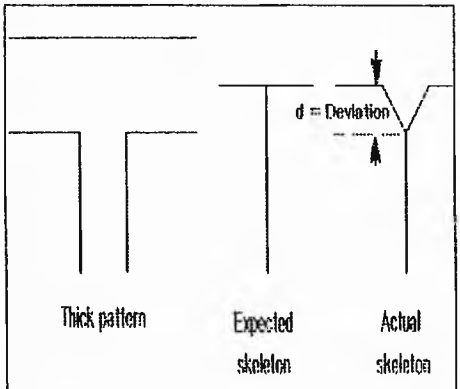


Figure 2.13 Expected and Actual Skeletons

The expected shape is quite different from the actual shape. The maximum deviation of the thinned line from the anticipated shape is shown in figure 2.14.

The deviation is directly proportional to the width W. In an attempt to overcome the distortions which are produced particularly in thick patterns where strokes intersect, Murthy[1974] suggests an extension of the usual 3×3 thinning templates. The technique searches for stroke intersection points or stroke end points by applying a series of non-rectangular templates to the image and then searching for vertical, horizontal, left and right inclined strokes from each of the intersection points. Murthy also gives a comparison between his algorithm and those used by Stefanelli and Rosenfeld[1971] and Deutsch[1972]. For the character 'K', represented by the pattern given below, two quite different representations are given by the Stefanelli and the Deutsch algorithms. These differences would be more pronounced if the original character was less regular.

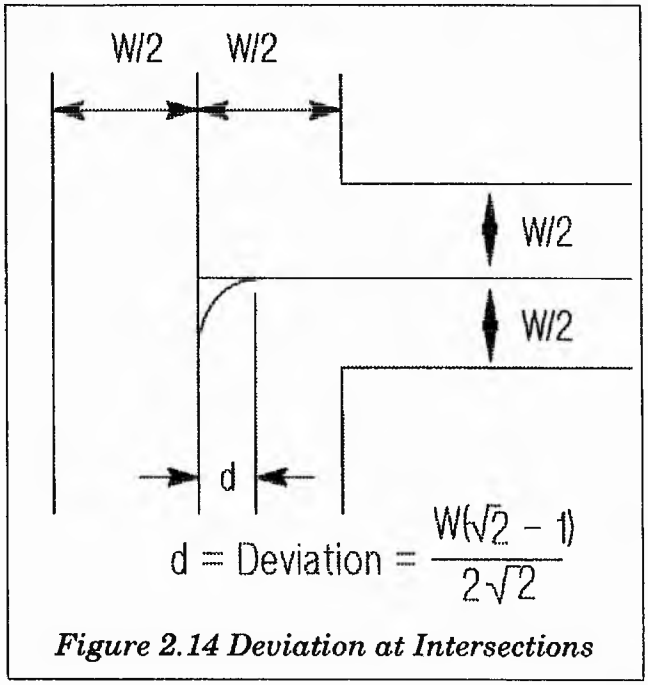


Figure 2.14 Deviation at Intersections

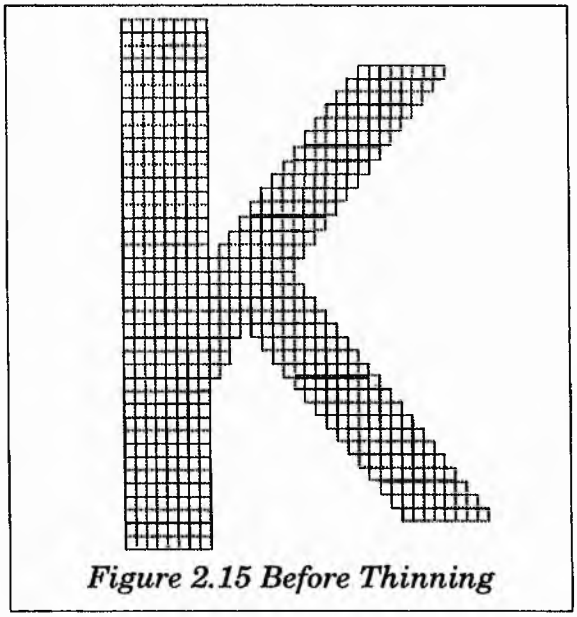


Figure 2.15 Before Thinning

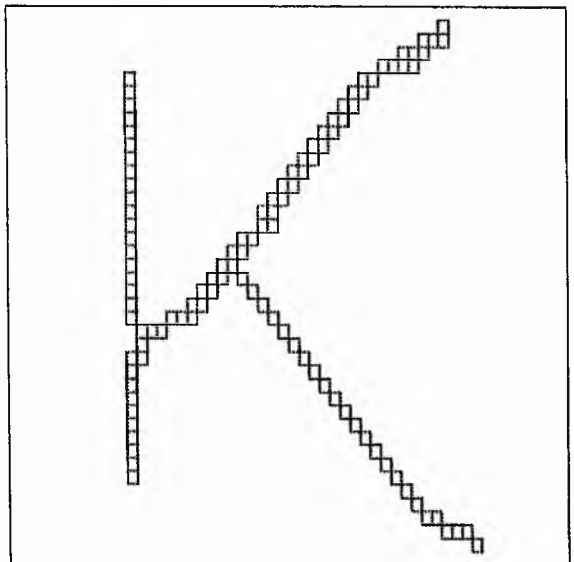


Figure 2.16 The Stefanelli Algorithm

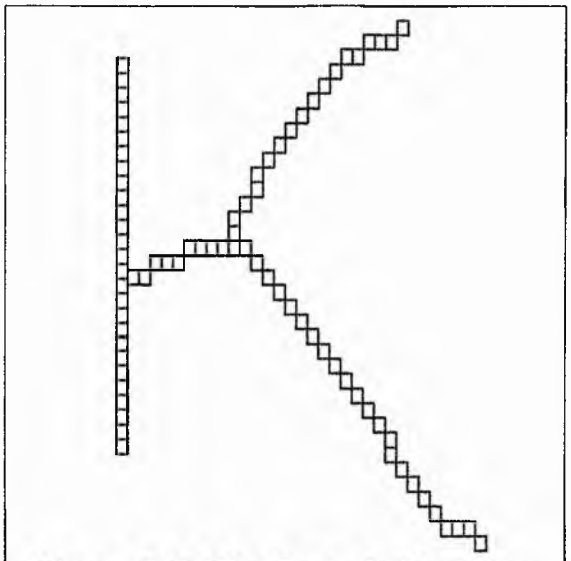


Figure 2.17 The Deutsch Algorithm

As can be seen from the above diagrams serious distortion results at the intersections and strokes are shortened.

With most thinning algorithms there is a serious problem which is their high sensitivity to irregularities in the surface of the shape. In extreme cases a slight change can cause a dramatic change in the skeleton as reported by Pavlidis[1982]. An example is given in figure 2.18, where a slight surface imperfection may lead to widely differing representations.

This problem is particularly pronounced when the shape contains thick strokes, that is strokes which are wide in relation to their length.

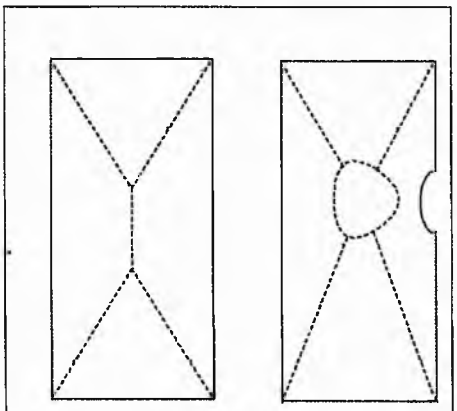


Figure 2.18 Effects of Imperfections

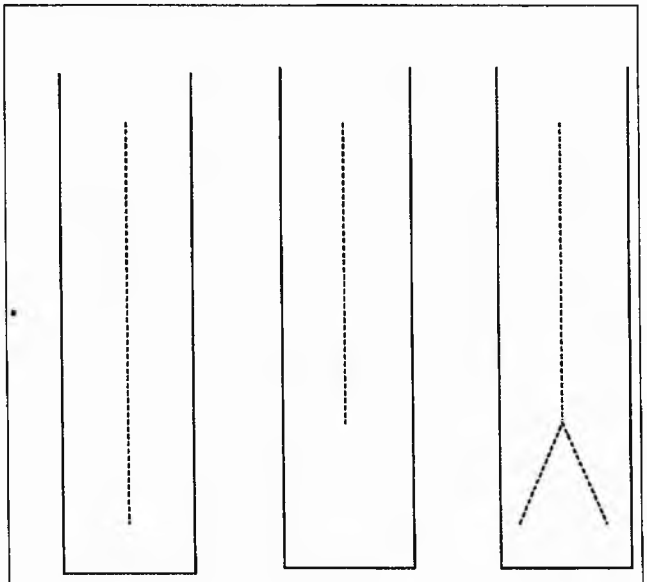


Figure 2.19 Parallel Sided Stroke Thinning

The point at which the skeleton ends varies for different implementations. There are three possibilities for a simple parallel sided stroke as shown in figure 2.19

The thinned version of the shape

is also dependent on the thickness of the original image.

2.5.3 Comments

The purpose of a thinning algorithm is straightforward. It is to produce a connected skeletal representation, which preserves the structural information of the original shape. However, the choice of algorithms and the different skeletons produced indicates that the selection of a thinning algorithm is not straightforward. Care must be taken to consider the sensitivity to irregularities in the border, the thickness of the strokes to be thinned, whether 4 or 8 connectedness is needed, and the preferred type of representation at intersections. The output of thinning algorithms is not always as anticipated by the human observer.

The use of an idealized skeleton is considered by Cox[1982] since a skeleton retains the essential structural information of a character, while removing details such as stroke thickness which are often a hindrance to recognition.

2.6 Syntactic Pattern Recognition

The problem of shape recognition is closely related to shape identification. If a pattern can be succinctly described in a compacted form in terms of either its outline, its area, or features, it can be compared to a database of known patterns described in the same manner and so identified. Fu[1980b] places the many techniques used for scene or pattern recognition into two categories: the decision-theoretic or discriminant and the syntactic or structural. The decision theoretic extracts a selection of features which characterizes the pattern and is used in the recognition algorithm. The syntactic approach views patterns as combinations of components. Both the components and the structural relationship between them can be expressed in terms of a sentence in a language specified by a grammar. Syntactic methods have been used for nearly

thirty years and early work was carried out by Eden[1962] and Ledley[1964, 1965]. A good overview of the work in this field is given by Fu[1986]. The syntactic approach can be used for compact image representation, since strings of primitives can be stored rather than the image itself. However, the main use is for pattern recognition. Ideally the primitives and the spatial relationships between them, should be deduced automatically by specifying the patterns to be recognized and applying algorithms capable of recognizing primitives and how they are connected. This is not currently possible and this process must be done manually. There are two stages to the development of a syntactic recognition system, firstly the selection of primitives and secondly the design of the grammar which relates the primitives to each other. As yet there are no general purpose pattern primitives which can be used for the representation of all patterns; selection must be made on the basis of the patterns to be encoded. The selection of the grammar is also data dependent. The selection of primitives and the grammar must be carefully considered since a simple set of primitives may require a complex set of rules to indicate how they are connected, while complex primitives tend to require a simpler grammar. Primitives should represent basic components of the pattern. For two dimensional patterns which may be described by boundary lines, primitives consisting of lines of specified size, shape and orientation are suitable choices. The notation used by many researchers to express pattern grammars was suggested by Aho and Ullman[1972], who expresses pattern grammars by the four-tuple

$$G = (V_n, V_t, P, S)$$

- G is the pattern grammar.
- V_n is the set of non-terminals.
- V_t is the set of terminals.
- P is the set of productions
- S is the starting symbol.

The terminals or variables are the lowest level primitives. The non-terminal variables are expressed in terms of the terminals and other non-terminals. The productions are a set of rewrite rules that give the relationship between the variables.

The syntactic approach has been successfully used for a number of applications. One of the earliest and most widely reported is the syntactic representation of chromosomes by Ledley[1964]. This was later to be extended by Lee and Fu[1972]. Keng and Fu[1976] used a syntactic approach for the identification of roads from satellite pictures. Ali and Pavlidis[1977] developed a syntactic representation of the alphanumeric character set.

2.6.1 Comparing Strings

There are inherent problems with the syntactic approach and despite some successful applications, syntactic methods are not widely used. Until recent years syntactic recognition was thought to be most useful for abstract or artificial patterns. When dealing with real world images, the strings generated which represent patterns in the image are unlikely to match exactly the patterns which are stored in the database. To resolve this serious difficulty there has been a number of developments. When a pattern string has been developed the simplest type of recognition process is template matching. When an exact match occurs recognition has been achieved; however, if an exact match is not found, action must be taken to determine which of a set of strings has the closest match. Levenshtein[1966] considered this problem although he did not attempt to apply it to image recognition. He suggests that the closeness between two strings was indicated by the number of error transforms needed to convert one string to another. He suggests three types of errors:

- Deletions. A missing item.
- Insertions. An extra superfluous item.

- Substitutions. An incorrect item, caused for example by the transposition of two items.

More recently error-correcting parsers to measure the distance between strings have been used by Thomason and Gonzalez[1975], Lu and Fu[1978], and Aho and Peterson[1972].

2.6.2 Polygonal Approximations

A complementary technique which has extended the range of applications of syntactic methods is the use of polygonal or functional approximation as a means of eliminating much of the superfluous noise found along the edges of shapes. Pavlidis[1982] describes a scheme for the description of closed boundaries which uses polygonal approximations for the representation of portions of the outline. The direction of each of the parts is specified as belonging to one of eight 45 degree sectors. Although the output strings are not completely orientation independent, the specified orientations relative to each other are consistent. When this technique was used on a sample of digits, each boundary encoding was slightly different. However, certain characteristics were present in all the descriptions which allowed the digits to be identified. Unfortunately, details of the data used to test the algorithm are not available for a more critical assessment. A syntactic approach which uses polygonal approximations was also used by Ali and Pavlidis[1977] for the recognition of handwritten numerals. An important advantage of polygonal approximation is that it overcomes one of the major problems of syntactic pattern recognition. The boundary of patterns often has a lot of noise which is not needed for recognition and which causes the representational strings to be long. By dividing the boundary and expressing each section in terms of a simple primitive curve or straight line, the strings are simplified and the recognition process made easier. This approach is therefore only useful where the pattern considered has a particular meaning, that is it represents, for example, either a digit, a character,

or an identifiable object rather than a shape which must be preserved. On a test sample of 1320 handwritten numerals a successful recognition rate of about 93% was obtained. One regular expression was not adequate to express all of the patterns which represented a particular digit. Even a simplest digit '1' required 3 expressions while '9' required 8.

2.6.3 High Dimensional Pattern Grammars

When a pattern grammar is used to express a pattern in terms of its boundary the only relational operator which is needed between its primitive is the concatenation operator. However, there are many patterns which cannot be expressed using only this operator, and a new set of grammars is required. These are known collectively as high-dimensional pattern grammars. A two dimensional grammar known as a web-grammar was suggested by Pfaltz and Rosenfeld[1969]. A sentence in a web grammar is a directed graph with symbols at the nodes or vertices. They show that this class of grammars can be used to express tree structures, two-terminal series-parallel networks and 'triangular' networks. A special case of a web grammar is identified when there is only a single terminal symbol. This type of web grammar is known as a graph grammar.

2.6.4 Comments

The syntactic approach has significant advantages in pattern representation and identification. Expressing complex patterns in terms of a small set of simple primitives with rules for describing the spatial relationship between them is very attractive. It allows patterns to be compressed into a highly compacted form and allows a quantitative measure of the similarity between pattern strings to be made by the use of metrics. The difficulty of the method is in choosing the

optimum set of primitives and the best grammar to state how the primitives are combined.

2.7 Implementation of a Character Recognition System

There are several recurring problems which need to be overcome in the development of a successful character recognition system. Rather than overcoming the difficulties of separating individual characters from adjacent characters and of resolving character orientation and size, many OCR systems simply operate in a highly constrained environment. When authors report on the behaviour of an OCR system or report on a low level operation used within the recognition process, the problems found in the implementation and the limitations of the system are not always clearly stated. For these reasons a working recognition system has been developed and is reported in this chapter, prior to the development of the new recognition system which is reported in the coming chapters. The objective here is not to extend the state of the art but to explore the problem as an essential step in the development of the novel recognition system which is described in later chapters and is the core of this thesis.

Aims: The aims of the recognition system developed are as follows:

- To separate a collection of adjacent binary characters into discrete characters.
- To normalize the characters for orientation.
- To normalize the characters for size.
- To compare the characters against a database of known characters and to identify them.

The recognition system consists of four programs. Communication between the programs is by files. The development environment is described in chapter 4.

2.7.1 Character Extraction(EXTCHR.C)

The input to this program is a binary 512×512 image file. The file EXT.IMG is output by the EXT program as described in section 4.5. This file has a group of 'black' characters on a 'white' background.

The characters are extracted by a region growing technique. The image is scanned from top to bottom until a foreground pixel that is a black pixel is found. This pixel and other black pixels which are adjacent to it are noted. The scan is repeated again from bottom to top and so on until no more adjacent black pixels are found. For example, the scan procedure for the letter 'M' proceeds as follows:

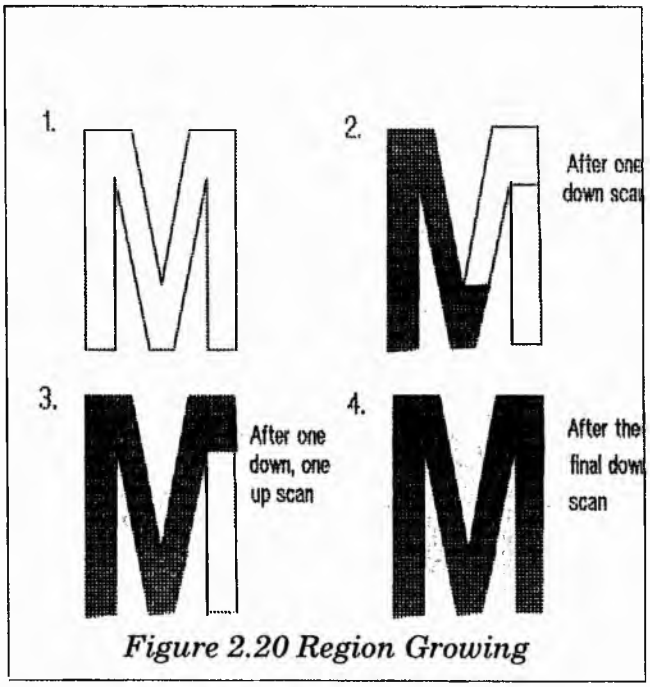


Figure 2.20 Region Growing

This process is repeated for each character. The characters are output to a series of 100×100 pixel files, called XA.SO1, XB.SO1 and so on. The number of files output is written to the file FILE.NO

2.7.2 Character Normalization(ROTATE.C, SIZE.C)

A common problem which has been considered earlier in this chapter is the normalization of the character. Before it can be compared against a database, it must be manipulated so that its size and orientation are the same as the patterns in the database against which it is compared.

The ROTATE Program: The aim of the ROTATE program is to normalize the character for orientation. In order to achieve this, a co-ordinate frame must be defined using shape properties which are independent of size and current orientation. The principal axes are used. This is achieved by simple calculation of the longest chord. This defines one axis. The other axis is the longest chord at right angles to it. The origin of the frame is taken as the intersection of these axes. The character is rotated about the origin so that the axes are horizontal and vertical.

The input to the ROTATE program is a set of binary files each containing a character as output by the EXTCHR program. The longest chord is found by determining the two pixels in the image which are the furthest apart. The other axis is found by calculating the distance between pixels along a line at right angles to the longest chord. The line which gives the maximum value is the second axis. The image is rotated about the intersection point of the axes such that the principal axis is vertical.

The process is repeated for every character extracted. The number of files is given by FILE.NO.

The outputs are a series of files called YA.SO1, YB.SO1 and so on.

The SIZE Program: The normalization process is completed by the SIZE program which scales the character to ensure that the maximum width and height along the axis is 100 pixels.

The inputs to SIZE are the set of files output by the ROTATE program. The outputs are a series of files called ZA.SO1, ZB.SO1 and so on. These files contain normalized representations of the characters.

2.7.3 Recognition(COMPARE.C)

The final stage is to compare the normalized shape matrix with a database of matrices representing known characters.

The inputs are the files output by the SIZE program. Each pattern is compared in turn to a set of 100x100 probability matrices, one for each character and an index generated. If a pixel in the input shape is white and the corresponding entry in the probability matrix is P_1, an index is incremented by P_1. If the pixel is black the index is incremented by $(1-P_1)$ since if the probability of a pixel being white is P_1, the probability of it being black is $(1-P_1)$. The probability matrix which yields the highest index identifies the closest match and therefore identifies the character.

2.7.4 Results

The results obtained for the recognition are very good; five representations of each character, excluding 'Q' since this not found on U.K. number plates, were submitted to the recognition process. Of these 175 characters, 170 were correctly identified. The incorrectly identified characters were '0','O','5','1' and '9'.

2.7.5 Generating the Probability Matrices

The database of probability matrices was developed by taking ten normalized representations of each character as produced by the method given above and by producing a probability matrix for each character which indicates the likelihood of a pixel being white. A value of P in the probability matrix indicates that in 10 representations of the character, (Px10) times the pixel is white.

In order to identify characters which are likely to be confused with each other, each of the probability matrices were compared to all the other matrices by the

program CONFUSE.C. The sum of the differences between the corresponding matrix values indicates the level of confusion between the two characters. The resulting confusion matrix is given in figure 2.21.

2.7.6 Discussion

A high value in the confusion matrix indicates a high possibility of confusion between the characters; particularly between 'O' and '0', 'B' and '9', 'I' and '1', and more surprisingly between '5' and '6'. For the recognition of characters on vehicle licence plates, the confusion between the numeric and alpha characters can be resolved since the format of the character sequence is known, typically a single alpha, three numerics and three alphas for vehicles sold after August 1983. In addition the characters on the plate all have the same orientation and size. This information can be used to develop a more reliable technique for normalization. By using these enhancements an accurate recognition system could be developed; however, there are a number of drawbacks which can only be overcome by employing fundamentally different recognition techniques.

The efficiency of the recognition system is very dependent on the quality of the characters input to it; for example, the selection of thresholds when converting to a binary representation could result in character shapes which are thinner than expected and therefore though clearly recognizable to the human observer, could not be reliably recognized by a system of this type. Characters with irregular edges would also cause problems. The failure of a recognition system to identify a character is serious, but incorrect identification is far worse since it does not permit notification to the user for him to intervene. This system incorrectly identified characters with as much certainty as it correctly identified characters. This weakness is a particular problem for this application since the characters on a plate consist of a group of random alphas, a group of random numerics and a single alpha. The more sophisticated OCR systems can readily

be linked to a spell checker which can help the recognition process in cases of ambiguity.

A	0																																
B	15	0																															
C	22	16	0																														
D	21	13	11	0																													
E	20	14	15	12	0																												
F	15	13	19	18	9	0																											
G	22	13	12	6	10	15	0																										
H	20	15	20	20	17	15	19	0																									
I	16	19	19	25	21	19	25	25	0																								
J	15	15	17	17	17	19	18	22	17	0																							
K	19	18	18	20	16	17	20	20	21	22	0																						
L	20	16	13	16	13	16	18	19	17	13	19	0																					
M	33	33	32	34	31	30	33	31	31	30	29	30	0																				
N	25	22	19	19	23	25	18	17	25	29	16	24	33	0																			
O	27	19	10	8	16	22	9	21	27	22	22	18	34	15	0																		
P	15	13	18	10	17	12	12	18	23	20	17	22	31	19	17	0																	
R	15	12	16	12	15	12	11	16	24	22	15	22	32	17	15	5	0																
S	21	14	13	13	10	14	11	21	22	17	16	17	31	20	14	17	15	0															
T	18	18	17	21	19	17	22	24	4	17	19	18	28	23	24	20	21	20	0														
U	24	14	10	8	17	23	11	16	23	14	18	13	33	17	11	16	16	15	22	0													
V	19	17	20	20	18	19	20	18	18	12	20	15	21	25	25	18	19	19	18	15	0												
W	23	30	33	38	34	29	37	26	24	26	29	28	30	31	38	33	34	33	27	33	28	0											
X	15	19	21	24	18	16	22	16	20	16	14	18	29	19	27	20	18	20	18	23	18	26	0										
Y	19	18	20	24	18	17	24	18	11	16	16	17	21	23	27	22	23	20	9	21	12	26	12	0									
Z	19	17	13	15	12	15	15	21	15	13	18	15	29	26	17	20	18	15	13	18	19	33	12	14	0								
0	27	19	10	8	16	22	9	21	27	22	22	18	34	15	0	17	15	14	24	11	25	38	27	27	17	0							
1	14	18	19	25	23	21	24	24	12	9	19	16	29	25	28	23	25	21	14	21	15	20	16	13	17	28	0						
2	21	13	12	13	9	12	11	16	20	16	19	15	31	22	14	16	12	12	19	15	17	34	18	18	11	14	21	0					
3	18	12	10	11	13	16	11	20	19	11	23	14	33	24	14	18	15	12	17	13	18	32	18	18	11	14	16	9	0				
4	9	15	21	17	19	15	18	22	18	12	18	19	32	24	24	14	15	21	18	19	17	26	18	19	20	24	15	21	17	0			
5	19	14	12	13	9	14	13	16	19	16	18	14	33	23	16	19	17	9	17	14	19	31	19	18	14	16	19	12	9	18	0		
6	19	10	11	11	8	12	10	16	20	14	18	13	33	22	15	17	14	8	18	13	18	31	19	18	13	15	20	9	7	17	5	0	
7	19	19	17	19	17	16	17	24	13	15	22	20	24	26	20	18	17	17	11	21	14	31	19	11	13	20	14	15	13	18	17	18	0
8	21	13	12	11	9	14	10	16	23	17	19	15	34	22	13	17	14	10	22	13	19	34	20	20	14	13	23	8	9	20	8	6	18	0
9	21	11	12	10	10	14	9	16	22	17	20	16	32	21	13	15	12	8	20	12	18	34	20	20	13	13	22	8	8	20	7	7	17	6	0
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	0	1	2	3	4	5	6	7	8	9

Figure 2.21 The Confusion Matrix

Two further problems are found with this recognition system. Firstly, some characters are nominally symmetrical such as 'A' and this leads to two possible normalized forms. Secondly, a new position must be calculated for each pixel of the pattern when it is reorientated in the normalization process. In some

situation this leads to the creation of holes within the pattern due to the rounding of the results of calculations to the nearest pixel.

CHAPTER 3

A NEW MULTI-STAGE ALGORITHM

CHAPTER 3

A NEW MULTI-STAGE ALGORITHM

The objective of the work is to develop algorithms suitable for the recognition of alpha-numeric characters in unconstrained environments. The images used are street scenes containing vehicles. The alpha-numeric characters read are on vehicle licence plates.

3.1 Plate Region Detection

There are a number of possible approaches to the reading of alpha-numeric characters on vehicle licence plates. Ideally a general purpose recognition system could be developed which would be capable of partitioning any image into objects which it can identify. This is not currently possible for unconstrained images. An alternative approach is to segment the entire image by a combination of edge detection and binarization techniques and to search for the alpha-numeric characters in the resulting output. However, there are serious problems with this approach. Firstly, the quantity of information in a typical 512×512 pixel image is very large and to perform edge detection, binarization and to search for small alpha-numeric characters would be very time consuming. Secondly, it is doubtful whether this approach would perform well. To identify the edges of each character, where the thickness of the strokes may only be ten pixels, would require an edge detection algorithm capable of detecting small edges. A detector of this sensitivity would contain large numbers of pseudo edges caused by variations in the shading of the picture and the texture of the objects. To binarize the image requires an adaptive thresholding algorithm to be used. This would generate many regions which did not correspond to objects. These problems indicate a recurring theme in pattern recognition; an unconstrained image does not contain insufficient information to allow

identification of an object; it contains too much information, which in this application obscures the alpha-numeric characters on the licence plate.

The first stage in reading the characters on the licence plate is to find the region containing the licence plate. In order to do this, it is not necessary to have a complete understanding of the image. However, if the image is not fully understood it is essential to use all the known information available about the scene to ensure that the plate is correctly identified. By identifying the attributes of the plate region, areas of interest can be found.

3.1.1 Characteristics of Licence Plates

Licence plates have the following features which enable them to be distinguished from the rest of the image:

- A defined shape, rectangular if viewed head on, a parallelogram if viewed from the side.
- The plate background is light in colour; white at the front, yellow at the back.
- The plate foreground is black at the front and back.
- The plate foreground is made up of discrete alpha-numeric characters.

3.1.2 Region Detection Metric

The detection of the licence plate region is achieved using a metric which checks areas of the image for the characteristics listed above.

a. A Second Difference Metric: A characteristic of the licence plate area is that there are a number of light- dark transitions. While the variation in grey scales is less than that caused by some other features such as headlamps, the vehicle licence plate is characterized by a close grouping of rapid intensity changes. This characteristic can be detected by generating an index for regions

of the image which takes into account both the intensity fluctuations and the close grouping of these fluctuations.

When deciding how to generate an index of this type it is essential to consider images typical of the situations in which the index is intended to be used. Therefore 100 digitized images of vehicles in a wide variety of environments were used as test data for this work.

The variation in intensity along horizontal lines in the images is determined. It is not necessary to do this for every line since the size of the region being sought must be at least ten pixels high otherwise the characters are too small to be read even by a human observer. Therefore every tenth line was considered. It is readily apparent that the grey scale variations found in the plate region are not the greatest found in the images. In particular, lights cause large, sharp transitions in intensity. Regular variations in intensity were found in some backgrounds for example, where there is fencing. However in these cases intensity variations tended not to be as great as in the plate region.

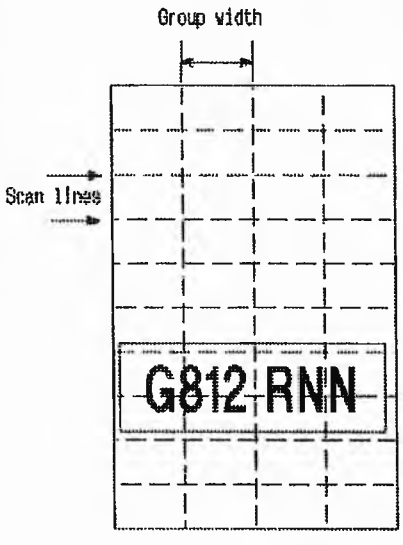


Figure 3.1 Scanning the images

The grouping of the edges generated by the collection of dark characters on a light background can be reflected in the index by summing the intensity change values over a small region of contiguous pixels along each of the scan lines.

Since for the purposes of generating the index it is unimportant if the transitions at edges are light-dark or dark-light, the modulus of the intensity changes can be taken.

The importance of the sharpness of the edges

relative to the number of edges can be emphasized by considering the second differences of the intensities of the pixels.

In summary, two indices can be generated. In both the image is scanned in a horizontal direction every tenth vertical pixel. The first is produced by summing the first differences of the grey scales for groups of adjacent pixels along the scanned lines. The second is produced by summing the second differences in the same way.

Both of the indices have the following effects:

- The distinction between light-dark and dark-light transitions is lost.
- The larger the individual transitions, the larger the index.
- The smaller the region over which the transition occurs, that is the sharper the edges, the larger is the index.
- Summing the differences over a small distance means that the licence plate region which has a number of fairly large transitions yields a higher index than a region which has a single very large transition, such as the edge of a headlamp.

By scanning across the image at regular intervals a series of indices for sections along each of the lines is obtained. A high index value is likely to indicate that the scan line crosses the plate.

In order to implement this algorithm there are two important variables to consider:

- The pitching of the scan lines; if they are set too far apart, the plate may be missed completely. If they are too close the searching is slowed down.
- The group size; if it is too large the effect of the plate may be masked by the background. If it is too small an insufficient number of the edges on the plate are considered and therefore a small index generated.

There are significant differences when using the first and second differences. It is important both to understand why there are these differences and to determine which performs best in this application.

The effect of using the first differences is to emphasize regions of transitions from light to dark. Areas of constant intensity yield zero values.

The effects of using the second differences are to further emphasize areas which have sharp intensity transitions and to give a zero value for areas where the rate of change in intensity is constant. The second difference increases the contribution of the sharpness of the edges to the index compared to the first order differences.

It is worthwhile identifying the factors which affect the size of the index and establishing the contribution of these elements.

The index is dependent on the following elements:

- The magnitude of the light-dark transitions
- The size of the light-dark transition region.
- The size of the light region.
- The size of the dark region.
- The group size for which the differences are summed.

Consider the value of the index based upon the sum of the second differences:

- L_t = size of dark --> light light--> dark transition.
- L_l = size of light region.
- L_d = size of dark region.
- N = (grey scale intensity of light region) - (intensity of light region)
- I_s = second difference index.
- I_f = first difference index.
- L = The length over which the differences are summed.

All sizes are in pixels.

For one dark --> light --> dark transition

$$I_s = 4N/L_t$$

Length of one transition = $2L_t + l_d + L_l$

If $L_l = L_d$ for length L

$$I_s = (L/(2L_t + L_d + L_l)) \times (4N/L_t)$$

$$I_s = 2LN / ((L_t^2) + L_d)$$

Similarly for the first differences index:

$$I_f = LN / (L_t + L_d)$$

The difference between the above two formulae is that the value of the index is inversely proportional to the square of the transition length for the second difference index and to the transition length for the first difference index. This has the effect of greatly increasing the value of the index for regions where there is a fast transition from dark-->light and light-->dark. The direction of the transition is not important since it is the modulus of the differences that is considered. A number of transitions within the length L also contribute to a high index.

As anticipated the second difference metric gives a far more reliable indicator of the plate region than the first difference metric.

What is the optimum value of L in terms of the plate length in order to obtain the maximum index value for the plate region?

P = the plate length

r = the ratio of (summation length / plate length)

$r = L / P$

For the range $0 < r < 0.5$, as the value of r increases, the value of the index increases in direct proportion since the whole of the length considered overlaps the plate. However for the region $r \geq 0.5$, where $r \leq 1$, as r increases the

maximum possible index increases but the probability of obtaining it diminish. The optimum value for r is obtained by maximising the expression:

f(r)=(probability of complete overlap)x(index obtainable)
+ (probability of minimum overlap)x(index obtainable)

$f(r) = (1-r)/r \times r + (2r-1)/r \times (1-r)$

$f(r) = 4 - 3r - 1/r$

$d(f(r))/dr = -3 + 1/r^2$

At the maximum value for r

$0 = 1/r^2 - 3$

$r = 1/\sqrt{3} = 0.577$

The optimum length over which the second differences are summed is 0.577 x plate length.

b. Image Binarization: A new metric is used for the detection of the licence plate region. As a result of this analysis, a series of lines are identified which can be used as the basis of the second component of the tool-box. Each of the lines is divided into sections along which the second differences are summed. The section which gives the highest index is considered first.

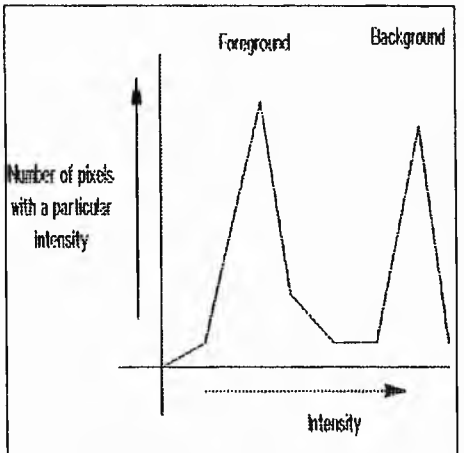


Figure 3.2 Idealised Distribution

It was anticipated at this point that an examination of the distribution of the pixel intensities would yield a bi-modal distribution if the line crosses the licence plate region. One peak in the intensity distribution corresponds to the foreground and one to the background.

The idealized distribution of grey scales assumes that the edges of each character are very sharp and each pixel is either in the

background or foreground. In fact, this is not achieved and a typical actual distribution is shown .

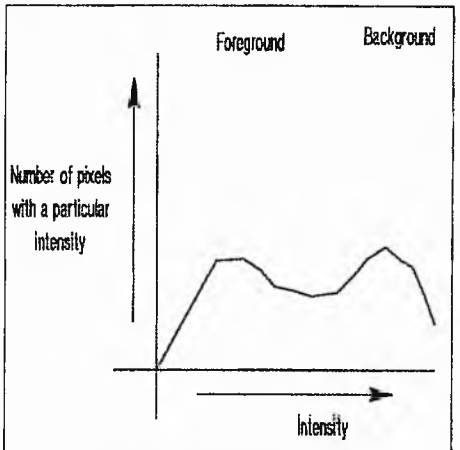


Figure 3.3 Actual Distribution

In a plate with a width of one hundred and twenty pixels such as image 3 in section 5.1.3. the characters are clearly legible. The width of each stroke of the characters is about seven pixels. The typical edge thickness is not less than three pixels and therefore there is no single intensity which corresponds to the foreground. The way in which the line, along which the distribution is being considered, cuts the strokes has a great impact on the distribution.

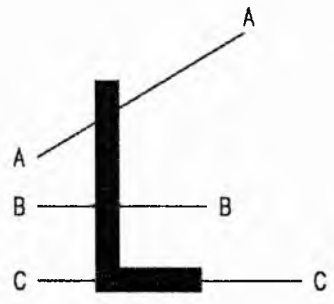


Figure 3.4 Character Intersections

The results of intersecting with a portion on the character 'L' along lines 'A', 'B' and 'C' as shown will yield different results.

In this application there are also problems with the intensity of the foreground which is not completely uniform.

There are however two common features of all the distributions. Firstly , the range between the lightest and darkest pixels is a maximum of about 250. Secondly, there are very sharp cut offs at each end of the spectrum.

The pixels fall into three categories:

- Black characters - the foreground.
- Yellow or white licence plate - the background.
- Transition pixels - between the foreground and the background.

Although the distribution of intensities does not correspond to a pattern showing two well defined peaks, it is still possible to use the information available as the basis for selection of a global binarization threshold. A value mid-way between the highest and lowest intensities is used as a global threshold. This method not only takes advantage of the sharp cut-offs exhibited by the intensity distributions for lines which intersect the plate region but also is not affected by variations in distribution.

When a global threshold has been determined it can be used for two purposes:

- To convert the image into a binary representation.
- As a basis for isolating the licence plate region.

Global thresholding is simple to apply; a pixel which has a higher intensity than the threshold is taken as 'white' all other pixels are converted to 'black'. Binarization of the image has several advantages. It is more straightforward to search for patterns in a binary image and the extraction of the patterns is simple. When dealing with an unconstrained image a global thresholding algorithm yields poor results and many edges are not detected. This is a serious drawback of global thresholding and has lead to the development of adaptive thresholding algorithms, in which the threshold chosen for different regions of the image varies. The only circumstance where global thresholding is effective is where the image consists of a single foreground and a single background intensity. This is the case for the portion of the licence plate region of the image. Although the selection of a single threshold is likely to obscure other edges in the images, it is advantageous in this situation where the objective is to isolate the plate region from the remainder of the image.

After the image has been binarized the quantity of information within the image is significantly reduced. It can be further reduced by the use of a region growing technique which isolates the plate region from the remainder of the image and simplifies subsequent analysis.

c. Region Growing: The objective of this work has not been to develop a complete understanding of the images presented, rather it has been to find areas of interest as quickly as possible and to discard regions of the image considered unimportant. The quantity of information present within a 512×512 pixel image with pixel intensities ranging from 0 to 255, is very large; the main difficulty in detecting and reading the vehicle licence plate stems not from insufficient information being available, but from too much being available. The quantity of information considered to be of interest is successively reduced by each stage of the analysis. As a result of binarization, instead of pixels having an eight bit intensity associated with them, they have one bit intensity values. The region growing process continues with this philosophy of removing superfluous information.

The whole image is not subject to the region growing scheme; instead, the process starts from the background pixels along the line which yields the highest value of the second order difference metric. All 4-connected and 8-connected background pixels are grown from them. It is important to use all the background pixels for the basis of the region growing since it is possible, that if only one of them is used, it may be fully enclosed within a region of a character such a '0' or 'O' and therefore the plate region is not fully identified.

It is better to region grow from the background licence plate pixels rather than the foreground pixels since this has the advantage of identifying the rectangular plate region. The characters on the plate are separate from each other and to region grow all of them would require a starting point for each one. This is not available at this stage.

In the region growing process used, the starting points are the licence plate background pixels. These pixels are given a temporary intensity value to distinguish them from the 'black' and 'white' pixels that constitute the binary image. For convenience, this intensity assignment is called 'grey'. The image is

scanned from top to bottom, and background 'white' pixels which touch grey pixels are changed to grey. This is repeated for successive rows of the image until no more pixels are changed. However, this is not the end of the process. The image is scanned again, this time upwards, starting from the bottom row containing grey pixels, until no more transformations occur. This pair of scans is repeated until a scan is made in which no more pixels are changed. Often only one downward scan and one upwards scan are needed however in extreme cases such as the example shown in figure 3.5, five scans are needed to convert all the necessary pixels.

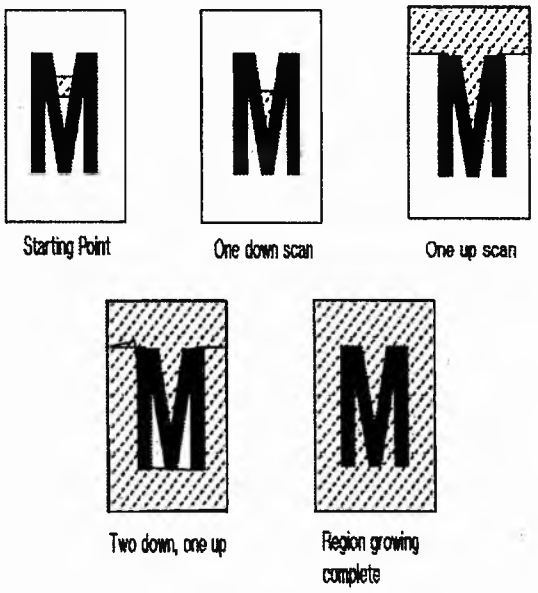


Figure 3.5 Region Growing

This process does require varying amounts of backtracking; however, this can proceed very rapidly due to the small amount of data which is being processed. A plate which is clearly legible may be only 150×50 pixels in size, less than 3% of the original image. Furthermore, these pixels are binary in intensity rather than possessing an eight bit intensity.

The region growing process has identified a set of connected pixels which enclose the characters on the plate. After the region growing process has finished, all information except for the plate and its characters can be extracted from the image and subject to further processing by converting all non-grey pixels to black and all grey pixels to white.

There is one problem with the region growing algorithm which requires a small amount of subsidiary processing. Some characters, such as 'D' and '0', contain fully enclosed background pixels. Since these pixels cannot be connected to the

pixels used to start the region growing, they are not part of the connected foreground set. A second stage is required to deal with background pixels which are fully enclosed within a connected foreground pixel set.

The region growing satisfactorily produces outlines of the characters but not the holes they may contain. The area which is developed as a result of the region growing process may contain completely enclosed background pixels. If this occurs when all the non-grown pixels are converted to an intensity different to the grown pixels, the holes disappear. As a result, the pixels which make up the character interior are not differentiated from the pixels which make up the characters. This process is illustrated by figure 3.6.

The grey pixels are shown as the hatched area. The black pixels are shown as black and the white pixels shown as white.

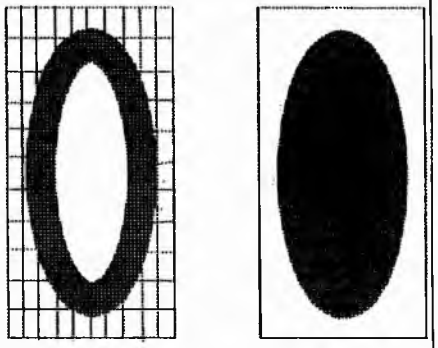


Figure 3.6 Character Extraction

The first part of the figure shows the plate background grey with the letter 'O' in black. The interior of the letter has been untouched by the plate region growing and is therefore white. The second part of the diagram shows the result of applying the simple rule that all non-grey pixels are converted to black and all grey pixels are converted to white. This erases all of the image except for the plate region and the fully enclosed characters.

3.1.3 Rectangle Recognizer

The identification of the licence plate outline increases the certainty that the region is the one being searched for. It is not necessary for the region extracted in the earlier analysis to correspond to the plate; however, it is anticipated that

at least traces of the plate edge will be found. If this is not the case then it is likely that the region under consideration is not valid. If this occurs, the second difference metric should be used again, in order to consider other regions of the original image which are likely to contain the plate.

The only portion of the original image which is considered in this analysis is the set of connected background pixels. Ideally, the outline of the region extracted corresponds to the vehicle licence plate. There are two basic shapes of plate which are in common usage. The usual pattern on vehicles is where all the characters are in a single line; a much less common form is found on commercial vehicles where the characters are in two rows, one on top of the other. In both cases two pairs of parallel edges should be detected and reported. If they are not found, there is a greatly reduced likelihood that this is the licence plate region. The direction of these vectors is cross referenced with a vector through the characters as reported in section 3.6.1

To isolate the characters and to remove the plate, region growing is started from the background outside of the plate. The grown pixels are changed to the same intensity as the plate. This leaves 'black' characters on a 'white' background.

In order to overcome the 'enclosed hole' problem, reference is made to the image first produced after the binarization process; each black pixel which makes up the character is compared to the same pixel in the binarized image. Any pixel which in the new image is black, while its corresponding pixel in the earlier image is white, is converted to white. This can be done very rapidly since each character is usually only a few hundred pixels in size.

3.2 A New Technique for Character Extraction

The image is scanned in columns, starting at the top left. When a foreground pixel is found, adjacent pixels on the same column are noted. In figure 3.7, these

pixels are the start of region 1. The process continues until other pixels are found which are not yet connected to the region 1 pixels. The process is repeated until groups of foreground pixels are collected into regions. It is straightforward to connect regions with each other. Sets of connected regions correspond to objects.

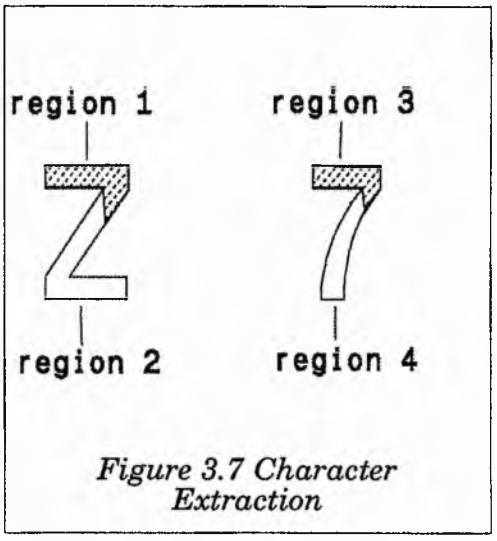


Figure 3.7 Character Extraction

Regions →	1	2	3	4
1		*		
2	*			
3				*
4			*	

* indicates connected

Figure 3.8 Region Connectivity

The scanning process forms the four regions as indicated in figure 3.7. During the scanning, a matrix is built up indicating the connectivity of the regions. For the two characters above the connectivity of the regions is given in figure 3.8. By combining regions 1 and 2, and 3 and 4, two character are produced.

3.3 Character Thinning

The recognition process begins after extracting individual characters from the original image. The characters are thinned to obtain a skeletal representation. Not all patterns are suitable for thinning; the thinned form of some patterns does not bear a strong physical relationship to the original unthinned form. Patterns which are suitable have a high structural content; they consist of interconnected members which are long and thin as shown in figure 3.9.

The 'blob' pattern has a large area in relation to its volume and is unsuitable for thinning. The behaviour of thinning algorithms tends to be unpredictable due to the sensitivity of most algorithms to slight surface irregularities. These irregularities have a much greater effect on the thinned form of 'blob' patterns than on long, thin patterns.

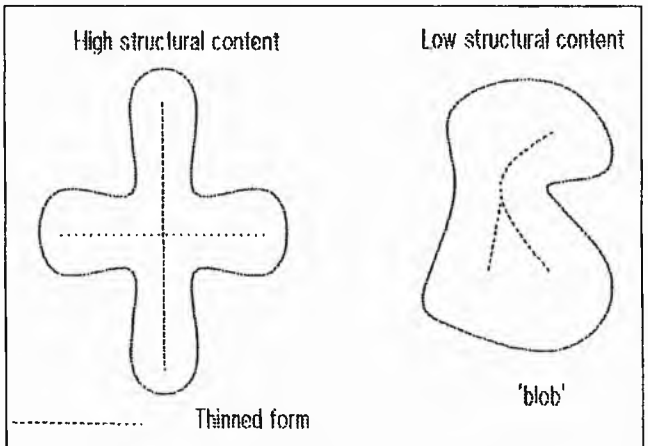


Figure 3.9 Structural Content

There are usually two reasons for thinning. The first is as an aid to a recognition process; used when the pattern has some intrinsic meaning, for example a character. The second is that it provides a compacted version of the original. Ideally in both cases the thinned form of each pattern should be a unique representation of the original. However, this is not always the case even for patterns which have a high structural content as is shown in figure 3.10. If the two shapes given are simply regarded as members of the same set of patterns the thinning is usable. In some cases, the convergence of patterns as a result of the thinning process is beneficial when the aim is to recognize the pattern rather than to store a compacted form of the original. In the case of alpha-numeric character recognition, it is most helpful that characters which are composed of strokes of different thickness yield the same thinned form.

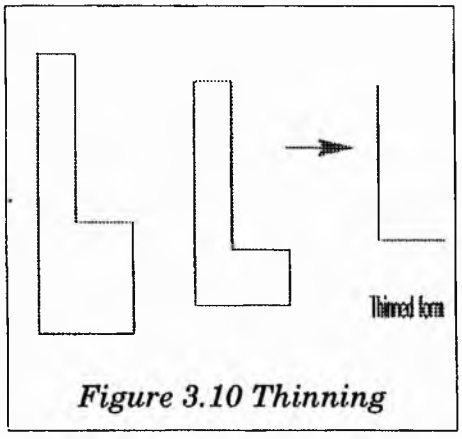


Figure 3.10 Thinning

The patterns which constitute the alpha-numeric character set are very suitable for thinning. Their structural content is high and they can be subdivided into thirty six subsets (A-Z and 0-9). The thinned form of a member of any subset contains sufficient information to differentiate it from any other subset. The

sole exceptions are the number '0' and '1' and the letters 'O' and 'I' which are very similar, but this is not a result of the thinning process. Even the best recognition system available, the human observer, makes frequent mistakes in the absence of context information. The range of patterns in each subset, that is the different forms of each character are varied. This is due to a number of reasons. There are many different font types, even after excluding the highly styled ones, such as Gothic. However, the range of fonts found on vehicle plates is comparatively small. The most important attribute of the licence plate is legibility. However, even when two patterns of the same letter from the same font are compared there are differences. This is due to either irregularities in the boundary caused by poor threshold selection when converting to a binary pattern, or to insufficient precision in the image capture and digitisation system. Thinning helps to eliminate these problems.

Since such a wide range of patterns can all represent a particular character there are occasionally ambiguous patterns. Consider the character in figure 3.11.

Figure 3.11 Character Ambiguity

The first character is clearly 'A', the second 'H', but the third is ambiguous. As a result of thinning it is interpreted as 'A'. The thinner the top line in the third representation the more the human observer is likely to interpret it as 'A' rather than 'H'. However, there is no correct interpretation, since the rules which define what it is that constitutes a particular character are heuristic and certain patterns even cause problems for humans.

It is highly advantageous to have thinning as the first process in the recognition process because:

- After thinning the character is still recognizable to the human observer indicating that, although information has been removed concerning the thickness of components of the pattern, that information has little bearing on the recognition process.
- The volume of data representing the character is reduced, so that processing can proceed more quickly.
- There is no shape boundary which in a conventional syntactic pattern recognition system is followed, broken into primitives and used to express the pattern. The pattern has been reduced to a series of lines and nodes

The thinning process is fraught with difficulties. Different algorithms tend to produce different representations particularly at the ends of strokes. For most algorithms, the thickness of the pattern alters the skeleton obtained. The criteria used to determine the skeleton are often not explicitly stated. The maximal circles criterion is often given, this is the locus of centre of the set of maximal circles for the pattern. This is hard to obtain in practice and researchers tend to use small templates compared against pixels at the boundary of the pattern, nibbling at the edges until a connected skeleton is obtained. The results obtained do not always agree with the skeletons drawn by humans, indicating that the maximal circles approach is not the criterion which individuals use. This is explained by the fact that humans examine the whole image when thinning, while the maximal circles and its implementation by using boundary templates examine only a small area. Most thinning algorithms exhibit extreme sensitivity to slight irregularities at the boundary. In addition there are problems caused by different forms which are produced by generating a 4-connected and an 8-connected skeleton.

Ideally an algorithm suitable for thinning alpha-numeric characters should have the following characteristics:

- Explicitly stated criteria used for the production of the skeleton.

- Insensitivity to boundary irregularities.
- Insensitivity to the thickness of the pattern.
- No 'flaring' of stroke ends.
- Rapid execution.

The new thinning algorithm has these characteristics.

3.3.1 A New Thinning Algorithm

Patterns which are most suitable for thinning have a high structural content. The alpha-numeric character set is a good example. The character set can be viewed as consisting of strokes some of which intersect. For example, the letter 'T' consists of two strokes at right angles to each other. Each of the patterns can be subdivided into two distinct regions.

- Stroke regions.
- Intersection regions.

A stroke region has two roughly parallel sides and is long in relation to its thickness. In the simplest case the thinned form of a stroke region is parallel to the two sides and directly between them.

An intersection region is formed by the intersection of two or more strokes. In order to find the thinned form of regions of intersections the skeletons of the stroke regions must be considered.

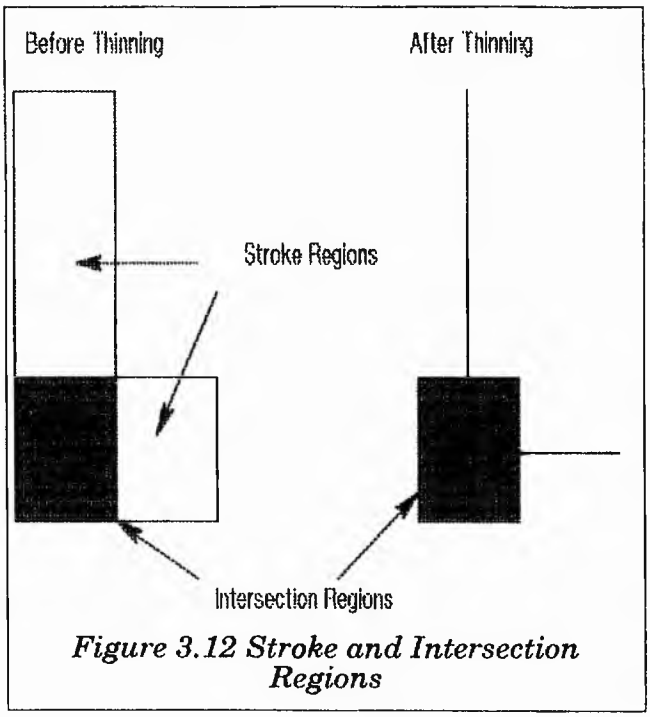


Figure 3.12 Stroke and Intersection Regions

For the simplest case of two strokes intersecting at right angles. The result of thinning the stroke regions is that two lines are obtained which are at right angles to each other, both of which enter the same region of intersection. The lines can be extended until they intersect. This gives a right angled 'L' shape.

The results for three and four stroke intersections are similar:

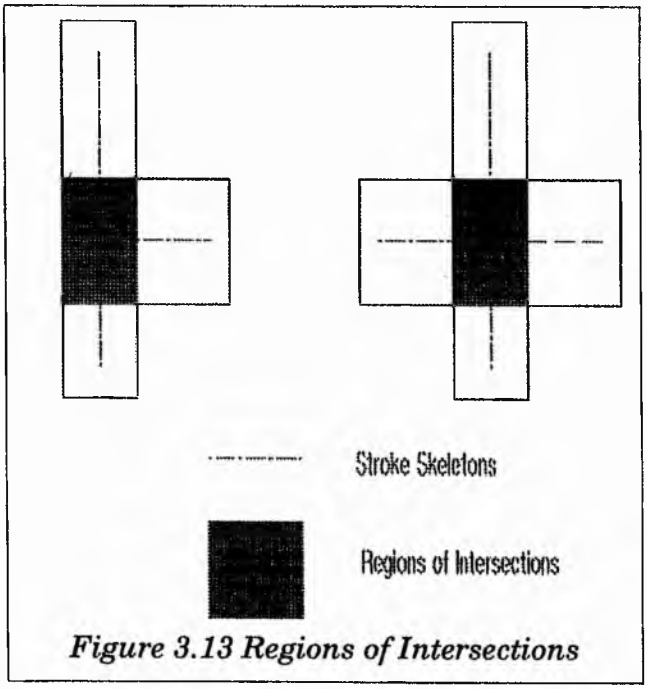


Figure 3.13 Regions of Intersections

By extending the thinned lines which result from thinning the stroke regions, into the regions of intersection, a complete skeleton can be produced. This yields skeletal representations which bear a close resemblance to those which are intuitively produced by humans. A stroke component of a pattern can be thinned by reference to its own shape, but the thinned form of a region of intersection is wholly dependent upon the strokes of which it is the intersection. In order to distinguish between the two region types the properties of each must be considered.

In the ideal case the strokes are parallel sided, and the thinned form is a line which is the locus of points which are mid-way between the two sides. These points have the property that vectors drawn from them to the nearest two edges will be at a 180 degrees to each other.

In order to execute quickly, the algorithm should be implemented as follows:

- Generate a list of edge pixels.
- For each interior pixel calculate the distance to the nearest edge pixel.
- Extend a vector at 180 degrees to the vector from the interior pixel to the nearest edge until it reaches another edge.
- Calculate the distance of the interior pixel to the second edge.
- If the two distances are the same, the pixel is an interior pixel.

An edge pixel is a foreground pixel which touches both a foreground and a background pixel. An interior pixel is a foreground pixel which touches only other foreground pixels.

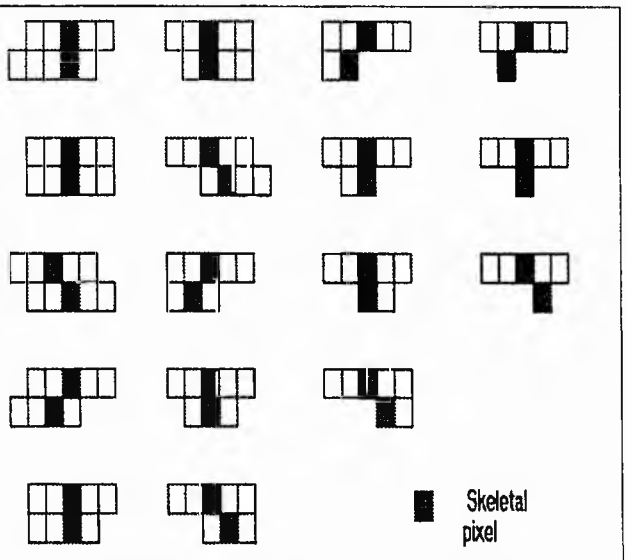


Figure 3.14 Pixel Connectivity

Applying this algorithm produces a skeleton for the stroke regions. If the stroke region is parallel sided, the skeletal pixels for the region are connected. This is not necessarily true if the sides of the stroke region are not parallel sided. However, the situations where connectivity occurs can be readily identified. Figure 3.14 considers a vertical stroke and the position and size of pixel rows which yield a connection with an adjacent row of five pixels.

A similar analysis can be performed for longer adjacent rows. The sizes and positions of rows of pixels which give connectivity with an adjacent row, of length N pixels, where N is an odd number of pixels in width is given below.

n is an integer and the row size is greater than zero

- For a row size N±2n, a displacement of one pixel to the left or right yields three positions.
- For a row size N±(2n-1), a displacement of one pixel from each of the nominally central positions (an even number of pixels has no one true central pixel, but two nominally central pixels) yields three positions.

Similarly for a row of length N pixels, where N is an odd number:

- For a row size N±2n, a displacement of one pixel to the left or right from each of the nominally central pixels yields four positions.
- For a row size N±(2n-1), a displacement of one pixel from each of the nominally central position yields five positions.

This is illustrated below in figure 3.15.

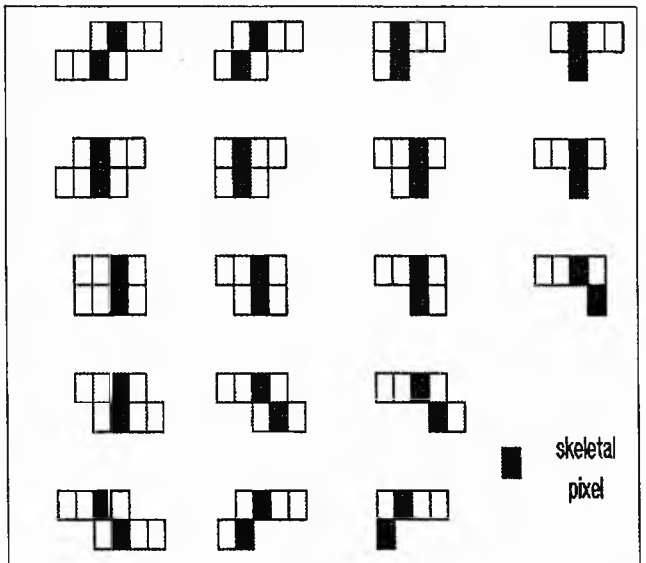


Figure 3.15 Pixel Connectivity

As a result of applying this algorithm, thinned forms of the stroke regions are generated. If the conditions given above are met, each stroke skeleton is connected.

The behaviour of the algorithm at the end of strokes is very predictable and this means than the problem of 'flaring' is dealt with satisfactorily, as in the case of a simple parallel sided stroke.

As shown in figure 3.16 there is a gap between the end of the stroke and the end of the flared components of length L;

$L = (\sqrt{2}-1)w/2$

w is the nominal stroke width.

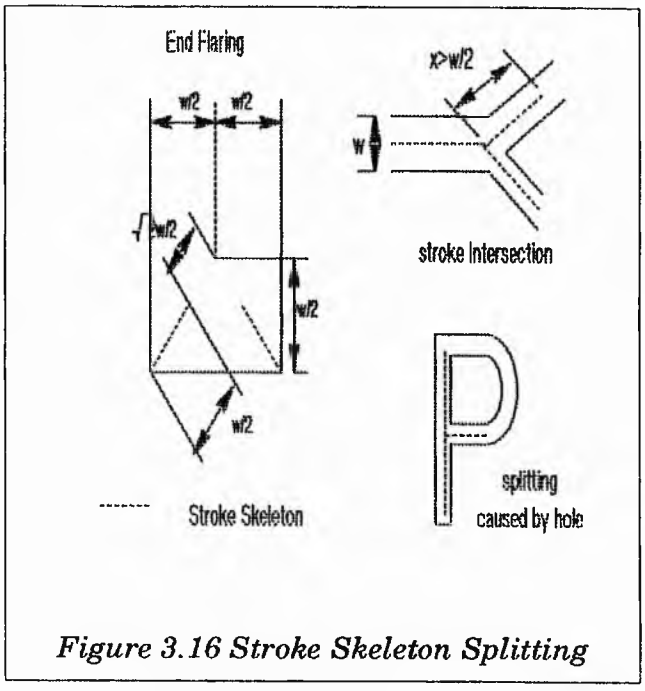


Figure 3.16 Stroke Skeleton Splitting

By ignoring connected skeletal pixels of length less than w/2 these pixels are eliminated. If there is an unusual pattern at the stroke end, a further simple check may be applied. A stroke skeleton splits only where there is a hole in the pattern or, in some unusual cases, where two strokes intersect. In summary, if a stroke skeleton splits and there is no hole between them and their length is less than w/2, end flaring has occurred and the flares can be ignored. In the ideal case the stroke skeleton ends w/2 from the end of the pattern.

The connection of the thinned strokes is determined from a consideration of the original image and the stroke skeletons.

Consider the case of a two way intersection. The extension of the stroke skeletons into the region of intersection until they cross gives a complete skeleton.

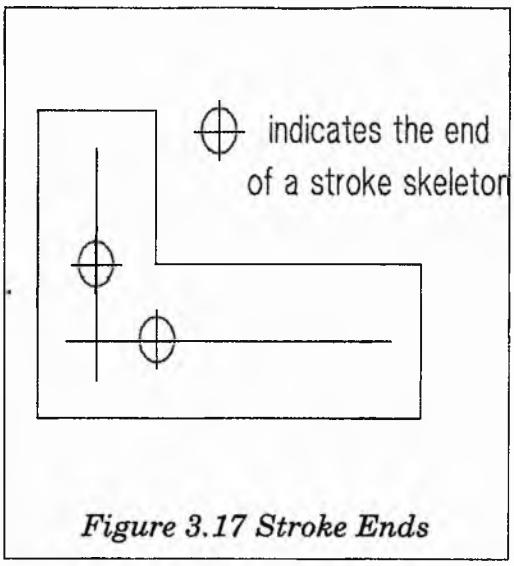


Figure 3.17 Stroke Ends

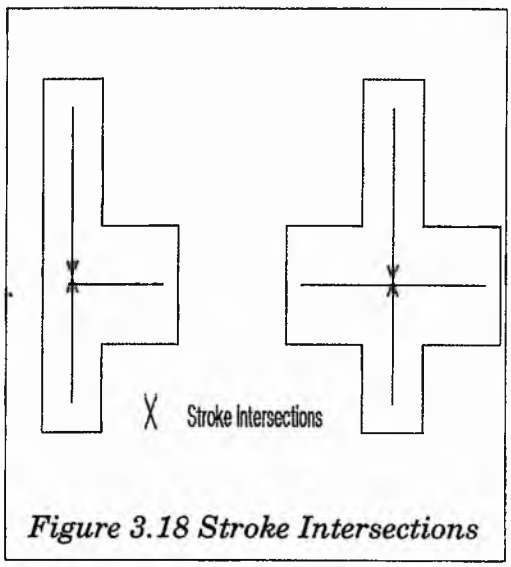


Figure 3.18 Stroke Intersections

The problem of extending into the intersection region in order to determine the skeleton is more complex for an intersection of more than two strokes. Before considering the general case there is an important special case to consider.

The crossing point which results from extending the stroke skeletons is shown in figure 3.18 figure by 'X'. This is a special case, since the extended skeletons will not always intersect at one point. A more general case is shown in figure 3.19.

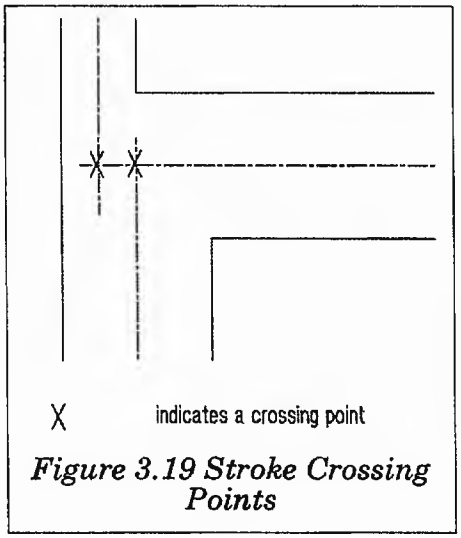


Figure 3.19 Stroke Crossing Points

In the general case of a three way intersection, there are two crossing points. Often these will be coincident, as in the case of the intersection in the letter 'T', since the two horizontal strokes are really a single stroke divided by the vertical stroke. In cases where there are two discrete crossing points, either may be chosen.

3.3.2 Advantages and Limitations

There are many advantages of this algorithm over conventional methods which use either sets of large templates or a maximal circles criterion:

- It should be faster to execute and does not require repeated passes over the pattern in order to produce a result.
- It should not be sensitive to small variations in the boundary of the pattern, which can cause large deviations in other methods.
- The problem of the flaring of ends can be satisfactorily resolved.
- The skeleton produced is the same irrespective of the thickness of the original pattern.
- The problem of unpredictable behaviour at stroke intersections does not occur.

There are drawbacks to the technique:

- A connected skeleton is not produced, and there is no guarantee that the stroke skeletons will be connected.
- The method only produces a meaningful result on patterns with a high structural content and stroke lengths which are long in relation to their width.
- A small amount of post processing is required both to remove superfluous pixels and to produce either a 4 or 8 connected skeleton as required.

The problem of the stroke skeleton not being continuous is the most significant drawback; however this can easily be dealt with by a small amount of post processing. The production of an 8-connected skeleton is easily solved by the use of simple templates. One pass is sufficient.

The limitation regarding patterns with a low structural content is not a problem, since any thinning technique is only valid on patterns of this type.

3.3.3 The Output from Pattern Thinning

Depending upon the type of use to which the output is to be used, there are two possible strategies. The first is to produce a connected skeleton and then to proceed to a conventional recognition system. Such a system may use either template comparison, or a syntactic scheme where the primitives are expressed in terms of stroke size and orientation. The second strategy is to use the connectivity information as the basis for a recognition system, strokes which enter the same region of intersection are connected. A syntactic recognition scheme based upon the intersections of nodes is now developed.

3.4 Syntactic Representation

The syntactic approach to pattern recognition expresses patterns in terms of primitives and operators which describe the spatial relationships between them. The set of primitives and the operators constitutes a pattern grammar. The usual technique is to specify primitives which have a defined size and orientation, and to use only the concatenation operator. This works well in highly constrained environments, but it is not a successful technique when the input patterns are less controlled. While most grammars have the strokes of the patterns as the primitives, there is a class of grammars which have the nodes or stroke intersections of the pattern as the primitives. These are called graph grammars.

The recognition process is by matching the string which represents the pattern against a set of strings of known meaning.

It is difficult to provide a description of an alpha-numeric character which describes the set of patterns which would be recognized by a human observer as representing a particular character. Any pattern grammar which uses as its primitives, objects which include in their descriptions a size and orientation,

generate different strings for slightly different forms of the same characters. To overcome this problem a new graph grammar has been developed which expresses the character set in terms of the intersections of strokes, irrespective of the size and orientation of those strokes.

3.4.1 Graph Grammars

One dimensional pattern grammars are only able to use the concatenation operator. For more complex spatial relationships, additional operators are needed. In graph grammars the terminal symbols are the nodes in the graph rather than the branches. Sentences generated by a graph grammar are directed graphs with symbols at their vertices. A graph grammar can be expressed in terms of the four-tuple:

$G = (V_n, V_t, P, S)$

- V_n is a set of non-terminal structures, nodes and graphs.
- V_t is a set of terminals.
- P is a set of productions or writing rules.
- S is a set of initial graphs.

3.4.2 Unconstrained Pattern Recognition

There are two main stages in the development of a graph grammar. The first is to determine what feature of the pattern constitutes a node. In simple cases this may simply be an intersection point. The second is to develop operators which express the spatial relationships between the nodes.

Three types of nodes are identified. They are illustrated in the character 'A' in figure 3.20

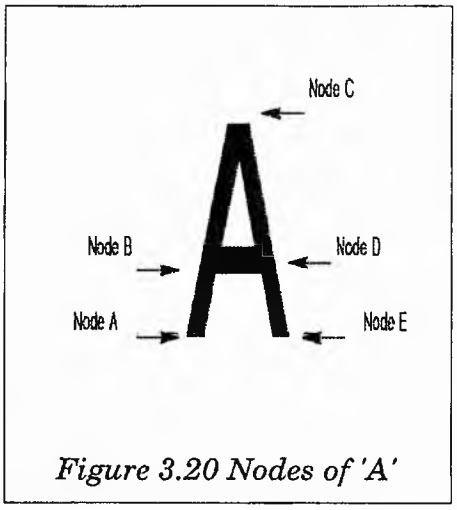


Figure 3.20 Nodes of 'A'

Nodes 'A' and 'E' are distinguished as end points. Nodes 'B' and 'D' are intersections between three lines and node 'C' is the intersection of two lines, signalled by an abrupt change in direction of greater than 90 degrees.

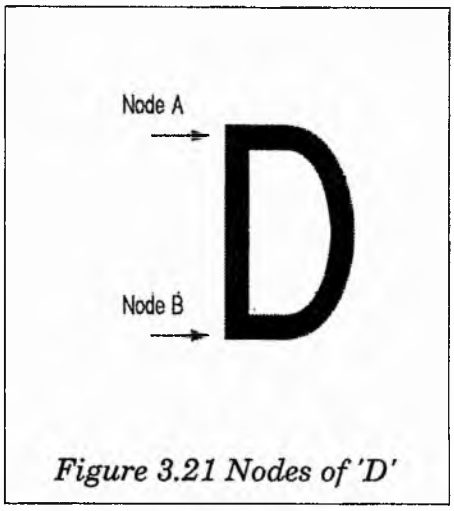


Figure 3.21 Nodes of 'D'

It is quickly apparent that the selection of nodes is inadequate. The characters '0' and 'O' have no nodes according to this scheme, since there are no end points, intersections, nor abrupt changes in direction. There are also ambiguities with characters such as 'D'. For this character nodes A and B are connected. This is the same information as for 'I' and '1'. There is also a difficulty with other characters. For example in figure 3.22, the representation 'A' is incorrect unless a new node type is introduced, since node 'C' is not found because the upper part of the character from nodes 'B' to 'D' is curved.

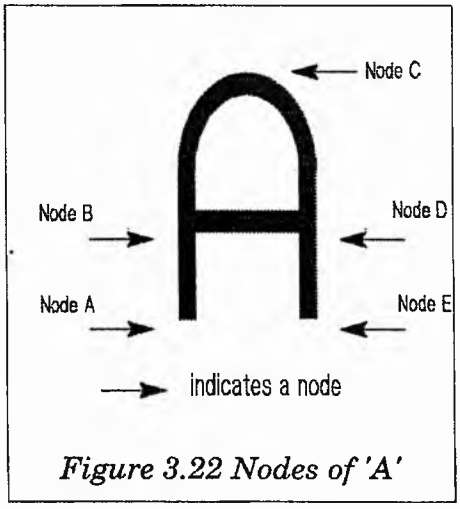


Figure 3.22 Nodes of 'A'

Clearly a new node type must be developed. It is essential to follow the curves of a pattern and to identify a change in the curve direction of ninety degrees as shown in figure 3.23.

In addition a node type for the intersection of lines as found in the character 'X' is needed. This gives four node types as shown in figure 3.24.

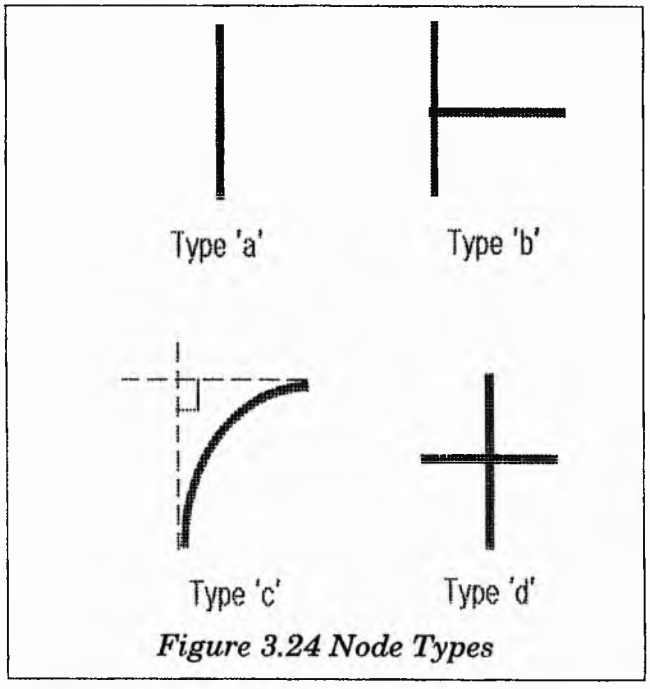


Figure 3.24 Node Types

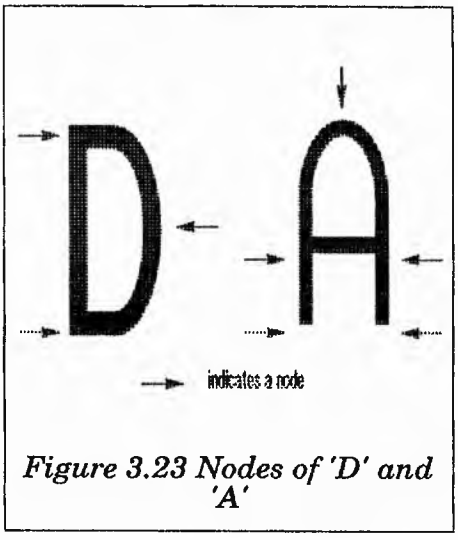


Figure 3.23 Nodes of 'D' and 'A'

It should be noted that the type 'c' node is used for abrupt changes in direction and also for curves where there has been a change in direction of greater than or equal to ninety degrees.

3.4.3 Primitive Selection

For the representation of the information contained within the connectivity matrix, a low dimensional pattern grammar is not adequate since it can only use the concatenation operator. A high dimensional web or graph grammar must be used. The most suitable grammar type is to use a graph grammar which is capable of expressing the relationships between the nodes of the alpha-numeric character set. The smaller the number of primitives in a pattern grammar the more complex is the grammar needed to express how they are related. However

the scheme proposed here has both a small selection of primitives, that is node types, and a simple grammar.

The following types of operators for expressing spatial relationships have been identified as shown in figure 3.25.

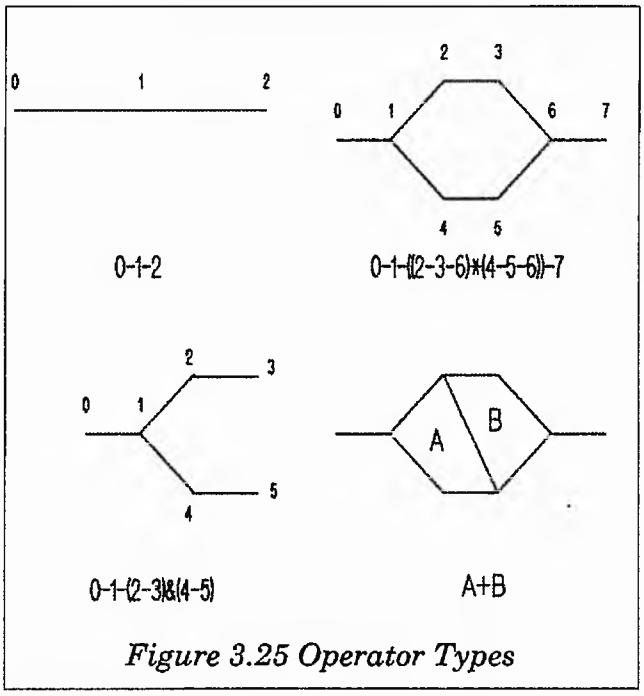


Figure 3.25 Operator Types

The numbers on the diagram indicate arbitrary node numberings.

The '-' operator is used for expressing a simple connection between nodes without branching.

The '&' operator is used when branching occurs, and the two branches do not join at a later point. It indicates two alternative paths.

The '*' operator indicates two paths which join together again after branching

The '+' operator indicates the concatenation between two patterns, that is between two loops which share a single side.

a. String Conflicts: Using these node types there are a number of conflicts when the same string represents two characters. This occurs for:

- 'C', 'U', 'N' and 'Z'.
- 'S', 'M' and 'W'.
- 'Y' and 'T'.
- '0' 'D' and 'O'.
- 'I' and '1'.
- 'J', '7', and 'L'.

- '6' and '9'.

Some of the conflicts such as between '0' and 'O' and between '1' and 'I' are inevitable because the characters are so similar. Even a human observer is often unable to differentiate between them without using context information. Fortunately, in the U.K. most licence plates follow a strict pattern of either three alphas, three numerics and an alpha or, for more recent vehicles an alpha, three numerics and three alphas. The gap between the block of three alphas and the three numerics is larger than the gap between individual characters. It is possible from this information to infer the correct interpretation of the character.

A small number of vehicles have 'personalized' plates which do not follow this pattern. A few vehicles have plates where there is a deliberate attempt to make the number '0' look like the letter 'O'. In these cases it is the intention of the plate owner to deceive the observer into confusing certain pairs of characters. Dealing with these cases is beyond the scope of this report.

The syntactic representation is not dependent upon the orientation of the character. While this has many advantages it can lead to some confusion. An upside down 'M' is very similar to a 'W', similarly '6' and '9' and 'L' and '7' can be confused. This problem can easily be solved by using context information, since it is known that all the characters on the plate have the same orientation and most characters are unambiguously represented. Indeed, in the case of a plate consisting entirely of '6's and 'L's, it is still possible to infer the correct interpretation, since even a vehicle which is cornering is unlikely to have its plate at an angle greater that 20 degrees to the co-ordinate frame of the image. Therefore in the case of '6' and '9' an adequate test is determines if the loop of the character is at the top or bottom within the image's frame.

This leaves a few character pairs which cannot be resolved by using context information. 'U' and 'V' are very similar in appearance. At the two extremes, the character 'V' has a sharp intersection at its base, while 'U' has a curve.

However, there are many intermediate forms of the two letters which are ambiguous either as a result of the font or due to a lack of clarity in the image. The same is true of the characters 'Y' and 'T'. While the extremes are easy to identify there is a range of intermediate shapes which cannot definitely be assigned to either character. In these cases the relative directions of the strokes as they enter the regions of intersection can be used to differentiate between them. Such a criterion is heuristic, but this is inevitable when a variety of fonts are used and distortions occur due to lack of resolution of the images and the effects of processing the image prior to this stage of the analysis.

In order to overcome the problems of confusion between the remaining characters such as 'J' and 'L' and 'T' and 'Y', the introduction of two further node types is required.

After determining the co-ordinate frame of the characters 'J' and 'L' the type 'c' nodes can be considered from the viewpoint of a starting node and it can be determined if the node is generated as a result of a clockwise or anti-clockwise change in orientation of the line.

To resolve the 'T' and 'Y' conflicts, the three way intersections can be classified into two types dependent upon the relative angles of strokes as they enter the regions of intersections. To implement this requires a considerable amount of extra processing and leads to an increased number of strings representing some characters.

A serious problem is the differentiation between 'D' and 'O'. The character 'O' is not used to indicate the year of a vehicle, but both letters are used in the alpha part of the plate. In this case a heuristic criteria must be used to differentiate between them. In the letter 'O' the position of the upper two nodes will be in the same vertical position as will be the two lower nodes. This is not the case for the letter 'D'

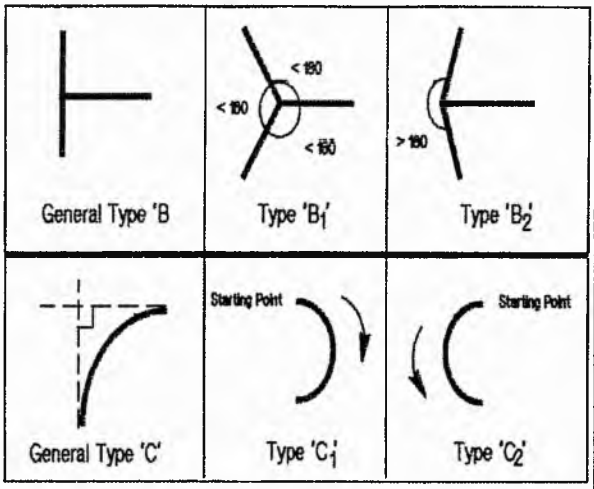


Figure 3.26 Subdivision of Type 'B' and 'C' Nodes

The new node types are shown below in figure 3.29:

The introduction of additional node types is not required by some characters since they yield a unique representation. It is only when the string produced is identified as representing more than one character that it is necessary to consider the additional node types.

3.4.4 A New Grammar for Alpha-numerics

The relationships between the nodes can be expressed formally. Since the recognition system is based upon the relationship between nodes rather than the branches connecting them, web grammars are ideal for expressing node relationships. The grammar given below has been used to express the alpha-numeric character set:

$G = (V_t, V_n, P, S)$

$V_t = \{a, b, c, d,\}$

$V_n = \{$<character>, <subweb>, <node>, <operator>$\}$

P:

<character> ⟶ <subweb><operator><subweb>

<operator> ⟶ *

⟶ +

⟶ -

⟶ &

<subweb> ⟶ <node><operator><node>

<subweb> ⟶ <node><operator><subweb>

<node> ⟶ a

⟶ b

⟶ b_1

⟶ b_2

⟶ c

⟶ c_1

⟶ c_2

⟶ d

$S = \{a \mid b \mid b_1 \mid b_2 \mid c \mid c_1 \mid c_2 \mid d\}$

The '|' operator means 'or'.

The lengths of the lines which connect the nodes are not considered, nor is the path they follow, provided that the path does not deviate by ninety degrees or more, since this would be interpreted as a further node.

The node types are given in the string representations of the patterns rather than some arbitrary node numbering.

A type 'a' node is chosen as a starting symbol. If a type 'a' is not available then either a type 'b' or 'c' is chosen. It is clear that there are possible ambiguities when there is more than one possible starting node and when the character is not symmetrical. This is shown for the character 'E' in figure 3.27.

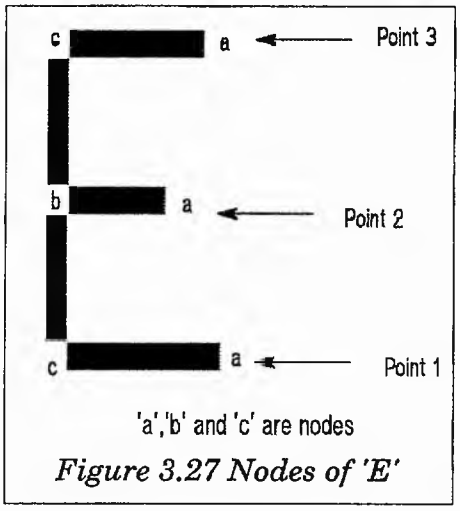


Figure 3.27 Nodes of 'E'

If the starting point is 1 or 3 the string representation is:

a-c-b-((a)&(c-a))

If the starting point is 2 the string expression is:

a-b-((c-a)&(c-a))

Clearly a recognition system must detect that these strings represent the same pattern; to determine if a string representing a pattern is an 'E' it must be compared against two strings. In the list given below, where there is more than one representation of the character, such as for 'E' all representations are shown. The letter 'Q' is not considered since this not found on licence plates in the U.K.

3.4.5 Character Strings

A a-b-((c-b)*(b))-a | a-b-((b)*(c-b))-a

B (b-b-c-c)+(b-b-c-c)

C a-c-c-a | a-c$_2$-c$_2$-a

D c-c-c-c

E a-c-b-((a)&(c-a)) | a-c-b-((c-a)&(a)) |
a-b-((c-a)&(c-a))

F a-c-b-((a)&(a)) | a-b-((c-a)&(a)) | a-b-((a)&(c-a))

G a-c-c-c-c-a | a-c$_2$-c$_2$-c$_2$-c$_2$-a

H a-b-((a)&(b-(c)&(c)))

I	a-a
J	a-c-a | a-c_1-a
K	a-b-((a)&(b-(c)&(c)))
L	a-c-a | a-c_2-a
M	a-c-c-c-a | a-c_1-c_2-c_1-a
N	a-c-c-a | a-c_1-c_2-a
O	c-c-c-c
P	a-b-((c)*(c-c)) | a-b-((c-c)*(c))
R	a-b-((b)*(c-c-b))-a | a-b-((c-c-b)*(b))-a
S	a-c-c-c-c-a | a-c_2-c_2-c_1-c_1-a
T	a-b-((a)&(a)) | a-b_2-((a)&(a))
U	a-c-c-a | a-c_2-c_2-a
V	a-c-a | a-c_2-a
W	a-c-c-c-a | a-c_2-c_1-c_2-a | a-c-c-c-c-c-a
X	a-d(a&a&a)
Y	a-b-((a)&(a))| a-b_1-((a)&(a))
Z	a-c-c-a | a-c_1-c_2-a
0	c-c-c-c
1	a-a
2	a-c-c-c-a | a-c_2-c_2-a
3	a-c-c-b-((a)&(c-c-a)) | a-c-c-b-((c-c-a)&(a))
4	a-b-((c)*(c))
5	a-c-c-c-c-a | a-c_1-c_2-c_1-c_1-a | a-c_2-c_2-c_1-a
6	a-c-b-((c)*(c-c-c)) | a-c-b-((c-c-c)*(c))
7	a-c-a | a-c_1-a
8	(b-b-c-c)+(b-b-c-c)
9	a-c-b-((c)*(c-c-c)) | a-c-b-((c-c-c)*(c)) | a-c-b-((c)*(c-c)) | a-c-b-((c-c)*(c))

3.4.6 Node Detection

There are two techniques which can be used for node detection, depending on how the skeletal forms of the characters are derived. Conventional thinning algorithms produce connected skeletons. If an algorithm produces 8-connected skeletons, node detection is as follows:

A type 'a' node has only one other pixel adjacent to the node pixel.

A type 'b' node has more than two pixels adjacent to the node pixel.

A type 'c' node is found where a line has no intersection with another line but undergoes either an abrupt change in direction, or a gradual change in direction of equal to or greater than ninety degrees relative to the direction of the line from another node.

A type 'd' node can be determined since it is the intersection of four strokes and at least one pixel where they meet is adjacent to 4 pixels.

The detection of a type 'a' node is straightforward; it is necessary only to search for a character pixel which touches only one other pixel.

The detection of a type 'b' node is also straightforward. However, determining precisely what type of 'b' it is, is more complex, since the angle of each of the intersecting lines with respect to the other two lines must be calculated. The gradient of a line can be found by calculating the 'best fit' line for the intersection point and for a small number of other points along each of the lines. This can be performed for all intersecting lines. The co-ordinate frame within which this is carried out is arbitrary but this is unimportant since the same frame is used for all intersecting lines and it is the angle between them that is significant.

The detection of type 'c' nodes can also be performed by calculating the gradient of a line for a group of points, advancing by one pixel, recalculating the gradient and comparing it to the previous value.

3.5 Alternative Node Detection Technique

In order to generate the graph representations of characteristics it is not essential to generate the connected skeletons first. If it is known that three strokes enter the same region of intersection then a type 'b' node exists at this point; similarly, when two strokes enter the same intersection region a type 'c' node exists. The detection of type 'c' nodes caused by a curve in the stroke can be detected by following each disconnected portion of the stroke from one to the other. Prior to doing this, checks must be made to ensure that if only two strokes enter the same region of intersection and are roughly parallel then a node does not exist.

3.6 Checking Procedures

Ideally, after performing the binarization process and region growing, the area considered would contain only the alpha-numeric characters, which would not be touching each other. In practice this is not the case. There are two problems which frequently arise:

- Non-characters are left.
- Adjacent characters are touching.

Small groups are sometimes produced which do not correspond to characters. These can be easily eliminated prior to the thinning and character recognition by ignoring groups below the minimum size which could be read if they were a character.

The problem of adjacent characters touching is harder to resolve. However, if a pattern is not recognized, a check is made to determine whether its width indicates that it may be two characters. The orientation of the plate is known from the calculated orientation of the plate region, and by the relative positions of the characters successfully read from the plate.

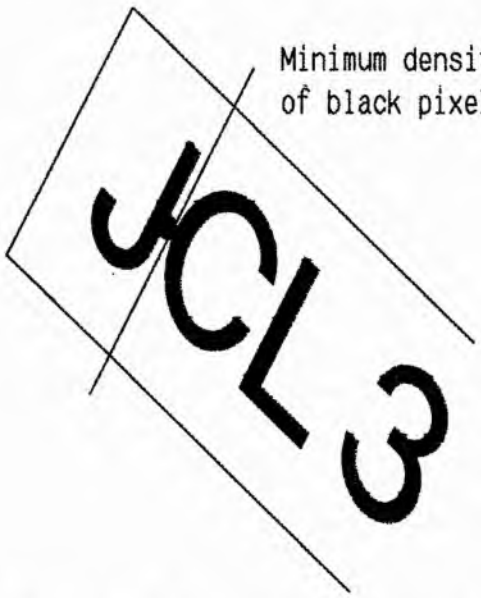


Figure 3.28 Separating Touching Characters

In order to separate the characters the density of black pixels is considered along lines at 90 degrees to the direction of the characters, and the long horizontal axis of the plate as shown in figure 3.28

At the point where the characters touch, the density of black pixels is a minimum for the central portion of the image. The characters can be separated along this axis and re-submitted to the thinning and recognition processes.

3.6.1 Extracted Object Distribution

A valuable check on the likelihood of the extracted objects being characters on a licence plate is to consider both their relative size and position. The size and orientation of the vehicle plate are known from the earlier analysis. The two orthogonal sides of the plate can be used to define a co-ordinate frame. Within this frame the height of the characters is constant, although the width varies between the widest character ('W') and the thinnest ('I'). Patterns which meet this criterion have the centre of their basic rectangle calculated. A line connecting the centres of these rectangles gives the orientation of the plate. This may be checked against the long horizontal axis of the plate calculated from the rectangle recognizer. They should be parallel.

CHAPTER 4

SOFTWARE DEVELOPMENT

CHAPTER 4

SOFTWARE DEVELOPMENT

The software consists of a set of programs which reflect the multi-stage nature of the algorithm. Data is passed between the programs using files. This means that the recognition process proceeds far more slowly than would be achieved if data remained memory resident. When the image has been binarized it is written to a file. The next program in the sequence is the region growing. This necessitates the reading of the binary file. For most of the programs the majority of the execution time is occupied by disc access. The main advantage of this approach is that it is easier to determine the behaviour of the system at each stage, by examining the input and output files. The detection of errors in the system is also more straightforward. An overview is given in figure 4.1

The programs often make reference to values, for example, the size of each row in the binary files. These references are referred to symbolically throughout the programs. They are given in the 'include file' VI.H. In this section the symbols are referred to in capitals, for example LINE_SIZE.

Several of the programs require parameters to be input. Sometimes it is preferable to be prompted for these; on other occasions, particularly if the program is being run from a batch file, it is preferable if in-line parameters are used. This is straightforward in 'C' as the number and a list of in-line parameters are given in the integer variable 'argc' and the array of addresses of the in-line parameters 'argv'. Both argc and argv are specified as input parameters to the program. If no in-line parameters are provided the value of argc is 1.

Outline structure charts as described by Jackson[1983] and Burgess[1984] are given for the main programs are in Appendix 2.

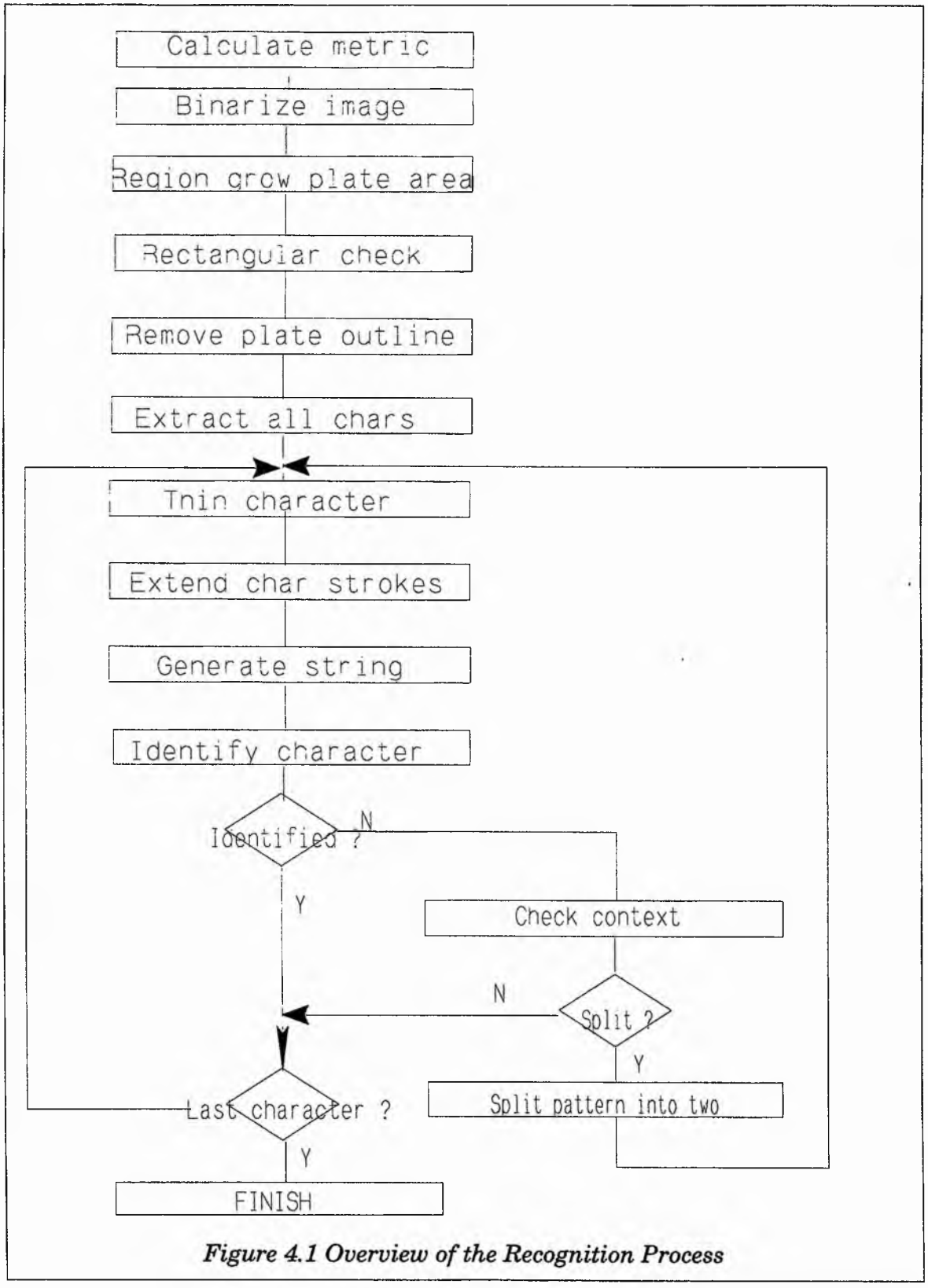


Figure 4.1 Overview of the Recognition Process

A system optimized for speed with a sophisticated user interface could be developed readily by the application of standard software design and implementation methodologies. This is outside the scope of this thesis.

The following set of programs find, extract and interpret the characters. They are intended to be run in sequence. Each program takes its input from files produced by the previous program and writes to files which are used by later programs in the chain.

4.1 Second Difference Metric(SECOND.C)

This is the first program in the suite. Its purpose is to determine the second differences of the pixel intensities of the images, which are indicative of the licence plate region.

The input to this program is a file containing a digitized 512*512 pixel image.

The first row of the image is read and the modulus of the first difference in intensity, I, between pixel n and pixel n+1 is calculated by $(I_{n+1}-I_n)$. The process is repeated to give the modulus of the second differences. The second difference values are summed for groups of adjacent pixels. The process is repeated along the line, to give LINE_SIZE/GROUP_SIZE values. The image file pointer is incremented by GAP_BETWEEN_LINES*(LINE_SIZE-1) pixels and a new line is read. The process is repeated for this line. At the left edge of the image it is not possible to derive a second difference value for the two leftmost pixels and therefore the average value for that group is used.

Two files are output. The first(SECOND.OUT) is an ASCII file containing a table of second order differences, each row of the table corresponds to one scanned row of pixels in the image file. The second(FILE.NAM) contains the name of the input image file. This is used by later stages of the processing.

4.2 Image Binarization(BINARY.C)

The aim of this program is to convert the image file to a binary representation where each pixel is either BLACK or WHITE, rather than possessing an eight-bit intensity value. It derives a global threshold which it applies to the entire image.

There are three inputs to this program. The first is a file containing the name of the image file(FILE.NAM). The second input is the image file itself. The final input is the file of second differences(SECOND.OUT).

This program reads the second difference file SECOND.OUT, output by the previous program. The highest second difference value is found. The GROUP_SIZE and the GAP_BETWEEN_LINES is known, therefore the maximum second difference value can be readily related to a group of pixels in the original image. If the maximum second difference value occurs in a position specified by row 'R' and column 'C' in SECOND.OUT. The row in the image file is given by:

$$R \times LINE\_SIZE$$

The column is given by:

$$C \times GROUP\_SIZE$$

The first row in the image is zero. The offset from the start of the image file is given by:

$$offset = (R \times LINE\_SIZE) + (C \times GROUP\_SIZE)$$

The pixels considered start at the offset calculated above and are of length GROUP_SIZE. They are read from the image file.

The maximum and minimum intensity values within the group of read pixels are found and the average is taken as the global threshold. The binarization process is straightforward, the image file is read and pixels whose intensity is less than

the threshold are converted to BLACK and all others to WHITE. A new binary file is produced called BINARY.IMG.

4.3 Region Growing(RGROW.C)

The aim of this program is to region grow from the background(WHITE) pixels of the licence plate. All connected background pixels are found and all other pixels are converted to BLACK. This program has the effect of highlighting the plate region and deleting all other information in the image.

The input is a binary image file, BINARY.IMG produced by the previous program and the second difference file SECOND.OUT

SECOND.OUT is read in order to determine the section of the image file which was used to calculate the global threshold.

The region growing process starts from all background pixels in the section of the image which was used to calculate the global threshold. All connected WHITE pixels are converted to GREY, that is any intensity which is not BLACK or WHITE. When this has been done all non-GREY pixels are converted to BLACK and the GREY pixels to WHITE. This has the effect of highlighting the licence plate background. It is seen as white on a black background. The characters on the plate are seen as completely contained groups of BLACK pixels.

This stage of the region growing has the side effect of filling the completely enclosed background areas found in characters such as '0' and '6'. However it does highlight the shape of the plate and allows a check to be made of the likelihood of this being the correct region.

The resulting image is output to a file called RGROW.IMG.

4.4 Rectangular Checking(RECT.C)

The input file is the binary image file produced by the region growing process RGROW.IMG.

The program chooses as a starting point the edge of the plate. The edge is followed in one direction until an abrupt change in direction of the edge occurs. The edge is again followed from the starting point in the opposite direction until an abrupt change occurs here. This process is repeated from the termination of the first edge. When a straight section is found this is followed. This section is terminated under the same conditions. The procedure continues until the whole of the plate perimeter has been considered. In this way a set of vectors is generated. Where possible the vectors are paired for parallelism. Ideally four vectors are generated which form two pairs. The magnitude of the two vectors in each pair should be the same. The pair with the greatest magnitude corresponds to the long horizontal side of the plate. The short vector pair correspond to the shorter vertical side. Ideally the two pairs should be roughly right angles to each other. However, this may not be the case even when the plate is clearly isolated due to the perspective of the image.

The direction of the larger of the vector pairs is taken as giving the orientation of the plate. This value is output to the file RECT.TXT.

The vector groupings are displayed as an indication of the validity of the region under consideration being a plate region. A failure to identify the plate outline tends to indicate that another region should be considered using the next most likely region indicated by the second difference metric.

4.5 Plate Removal(EXT.C)

The aim of this program is to generate an image file with the characters in BLACK on a wholly WHITE background.

The input files are RGROW.IMG and BINARY.IMG.

The program is a variant of the RGROW program. The image file RGROW.IMG has the characters in BLACK surrounded by a WHITE plate region which in turn is surrounded by BLACK pixels. This program region grows the outermost BLACK pixels and converts them to WHITE. This leaves the characters as BLACK on a wholly WHITE background. The region growing starts from a BLACK pixel in the top left of the image which is assumed not be a part of a character. All the connected BLACK background pixels are converted to WHITE. The output file is EXT.IMG

The problem of the filled holes is resolved by referring to the first binary file produced, BINARY.IMG. If a BLACK pixel is found in EXT.IMG whose corresponding pixel is WHITE in BINARY.IMG this pixel is fully enclosed in a character. It is therefore converted to WHITE.

The final output file EXT.IMG consists of an image comprising only the characters in BLACK with enclosed holes and background in WHITE.

4.6 Character Extraction(EXTRACT.C)

The image file EXT.IMG output from the EXT program has the characters on the plate as completely enclosed BLACK objects on a WHITE background. By searching for a BLACK region and growing all the connected BLACK pixels, individual characters can be isolated and each one can be output to an individual file.

The input to this program is the binary image file EXT.IMG.

The image file EXT.IMG is read into memory and scanned in columns, starting on the left of the image until a BLACK pixel is found. The connected pixels are grown into regions as described in chapter 3. More than one region may constitute a single character. If regions touch, they are connected to form one

new region. Only one pass is required to extract all the characters. In some images it was found that there are small groups of BLACK pixels which do not correspond to characters. These are typically less than ten pixels in size, therefore to eliminate them, the set is ignored if the size of the set of extracted pixels is less than MIN_CHAR_SIZE. Ignoring these small groups of pixels is valid since a pixel set which is less than several hundred pixels does not contain sufficient resolution to allow the characters to be interpreted even by a human observer.

Each of the connected set of pixels is output to a separate file. The name of the original image file is read from FILE.NAM as output by SECOND. The prefix of the name is given to each of the output files. The extension of the files is SO1, SO2 and so on. For example, the names of the files containing individual characters extracted from image file IM1.IMG is IM1.SO1, IM1.SO2 and so on.

The horizontal and vertical sizes of each character are measured to ensure that the pixel is centrally placed in the output image files. The image files produced are $100^{x}100$ pixels.

The number of files output is written to the file FILE.NO. This is used by later programs in the analysis

4.7 Context Checking(CON.C)

The role of the context checking part of the analysis is to determine the relative sizes and dimensions of the extracted characters. In the co-ordinate frame of the licence plate the vertical height of the characters is the same. The widths are similar.

Their are two advantages from this program:

- If the extracted characters are in a line, in the co-ordinate frame of the plate, this means that they are characters and a more detailed analysis can begin.
- If an extracted object is twice the width of the other items, but spatially correct, it can be deduced that the object corresponds to two characters and therefore it can be divided.

The input to this program is a set of 100×100 binary image files produced by the EXTRACT program, each of which ideally contains a single character, and the output of the RECT program RECT.TXT.

The output is a file, CONTEXT.TXT, which contains details of the characteristics of the input image files, specifically the height and width of the patterns and their position within the original 512×512 image file. From this information the orientation of the plate is calculated. These figures is compared to those output by the RECT program and the closeness is reported.

At this stage no attempt is made to modify the set of image files. If however an image file is not recognized by the later stages of the analysis, reference is made to CONTEXT.TXT in order to determine if the image file contains two characters which can be separated.

If the orientation of the plate is found to be within ± 45 degrees of the horizontal, no re-orientation of the characters is required by later stages of the recognition process. If the plate is outside of this range, re-orientation is necessary, since for example, the letter 'N' becomes 'Z' if it is rotated. In all the images considered no re-orientation is required.

4.8 Thinning (THIN.C)

This program takes 'solid' forms of the characters input and produces skeletal forms of them.

The names of the input files are derived in the following way. The file FILE.NAM, output by SECOND, is the name of the original image file. This is read to determine the prefix of the names of the solid characters. The number of files is given by the file FILE.NO, output by EXTRACT. The names of the files to be thinned is 'image_name.SOn' where n is between zero and the number of files.

The input file is read, and a list of edge pixels compiled. An edge pixel is a BLACK pixel which touches at least one WHITE pixel. Each BLACK pixel is considered in turn. Both the distance and direction to the nearest edge pixel are calculated. A vector is extended at 180 degrees to this direction and the distance to the nearest edge pixel in this direction is calculated. If the two distances are the same ±1 pixel the pixel is skeletal.

A further stage is required to ensure that an 8-connected skeleton is produced. The second stage comprises a single pass. Each remaining pixel is matched to see if it matches the patterns shown below

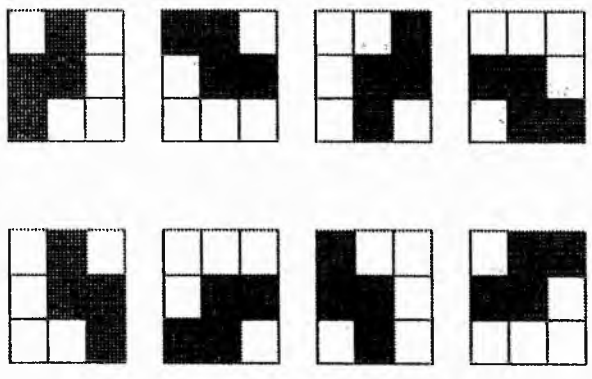


Figure 4.2 Thinning Templates

If the match is found, the central pixel is converted from BLACK to WHITE since, it is not required for 8-connectedness of the skeleton.

The output is a 100×100 binary file with the same name as the input file but with a '.SKn' extension, where n is a value between 0 and the number of files. The files contain only the stroke skeletons not the connected skeleton.

4.9 Extending Strokes (EXTEND1.C and EXTEND2.C)

The aim of these two programs is to extend the stroke skeletons into the regions of intersections. The number and name of the files which contain the output skeletons are derived in the same way as for the THIN program, except the file extensions are '.SLn' for EXTEND1 and '.SMn' for EXTEND2, where 'n' is an integer between 0 and the number of files.

This process is divided into two parts. The first program, EXTEND1, deals with the problem of two strokes which if extended, produce two lines which are roughly parallel and do not touch as shown below.

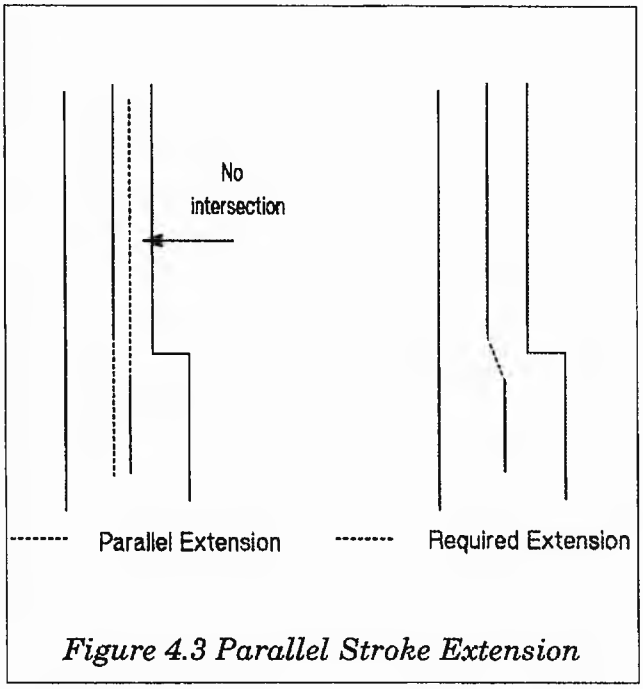


Figure 4.3 Parallel Stroke Extension

The EXTEND1 Program: The first part of EXTEND1 is to find the end points of strokes. This is straightforward, since for the 8-connected stroke skeletons, an end pixel is BLACK and connected to only one other BLACK pixel. Next, the directions of straight lines, which could be extended from each end of each stroke, are calculated. The direction of the line is based on a line fitted through the last 10 points of the stroke or the number of points in the stroke if less than 10. The process is repeated for each stroke end. Extensions which are close to 180 degrees of each other are found. For the test data used, within ±15 degrees of parallel was found to be suitable.

Pairs of stroke end points whose extensions meet this criterion are found, and the direction of a vector between them is calculated. If the direction of this

vector is close to the direction of the extended lines, these points are connected in the new skeletal. A value of within ± 20 degrees was found to be suitable.

The input files to EXTEND1 are FILE.NO and the first thinned form, which has a '.SKn' extension.

The output file is the thinned form with certain stroke ends connected if the criteria outlined above are met. It has a '.SLn' extension.

The EXTEND2 Program: This program completes the extension process. The process is very similar to the earlier program except that strokes are extended until they meet one of the following criteria:

- The edge of the 'solid' form of the shape is reached.
- Another extended stroke is met. If this is the case, both strokes are deleted after the point at which they intersect.
- A part of a stroke itself is met. The extended stroke is deleted after this point.

Two criteria may qualify as an intersection as shown below.

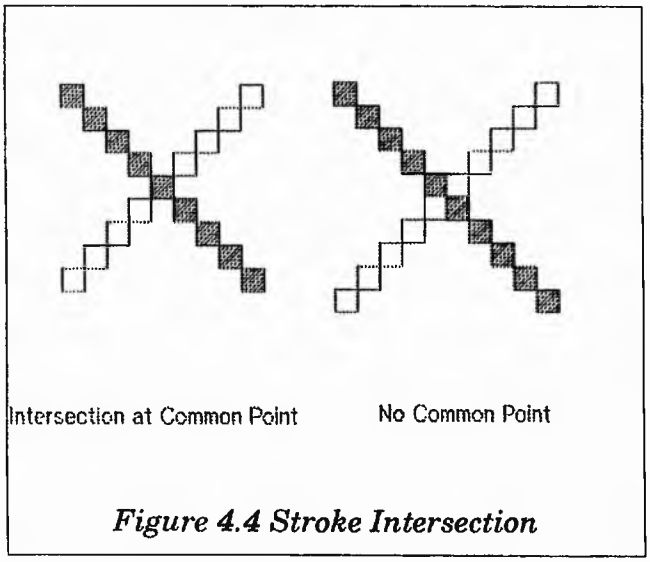


Figure 4.4 Stroke Intersection

An extended stroke which has no intersections is deleted.

The input files to EXTEND2 are FILE.NAM, FILE.NO, the solid form which has a '.SOn' extension, and the thinned form output by EXTEND1 which has a '.SLn' extension.

The output from EXTEND2 is an extended skeleton file, with a '.SMn' extension.

4.10 Syntactic Representation(SYNTAC.C)

The input to this program is FILE.NAM, FILE.NO and a file with a '.SMn' extension as output by the EXTEND2 program. The '.SMn' file is an 8-connected skeletal representation of a character.

The first stage in the analysis is to find the nodes of the character. The second stage is to determine the connectivity of the nodes.

The identification of nodes which are at the end of strokes is straightforward. End nodes are BLACK pixels which touch only one other BLACK pixel.

It is a little more complex to find the 3-way intersection nodes. A variety of intersection patterns are possible as shown in the figure below, since the extension process may result in skeletons containing pixels which are not needed for connectivity.

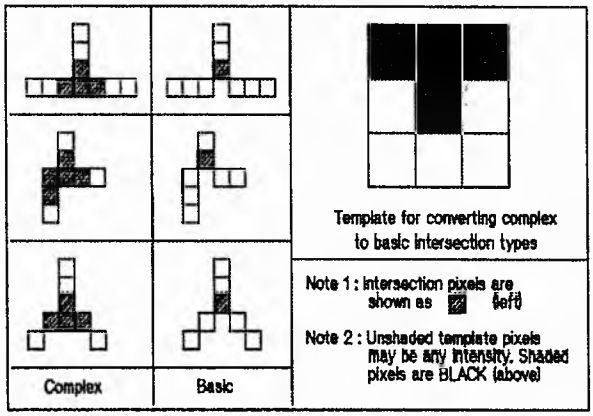


Figure 4.5 Basic and Complex Intersection Types

All of the example 'complex' configurations shown are reducible to a single 'basic' form. This form has a single intersection pixel which is connected to three other BLACK pixels.

The reduction of the more complex intersection types is straightforward. The 'T' shaped pattern of pixels are sought, using the template shown. The central pixel of the 'T' is deleted to convert from a complex to a basic form of intersection.

A second basic intersection type is also possible. This consists of an 'L' shaped group of three BLACK pixels each of which is connected to three other BLACK pixels. The two basic types are shown in figure 4.6.

After this preliminary analysis, intersection pixels are BLACK pixels which touch three or four other BLACK pixels. If the single pixel type is found, the three strokes extend from the three connected directions of the pixel. If the intersection is of the 'L' shaped type, one stroke extends from each of the three intersection pixels.

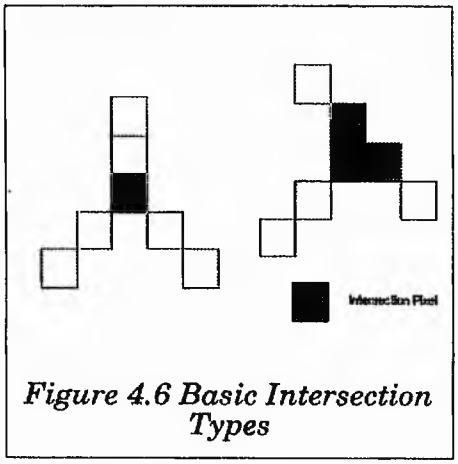


Figure 4.6 Basic Intersection Types

Apart from intersection and end pixels, all BLACK pixels touch two other BLACK pixels.

Pixel chains are followed from the end pixels until either another end pixel or an intersection pixel is found. The change in direction of this line is calculated and a variation in direction of more than 90 degrees is noted. This is a new node. In this way the manner in which the nodes of the pattern are connected is found.

The file output from this program has a '.SYn' extension. It contains the following information for each node.

- The type of node, for example, end or intersection.
- A node number, which is used for node identification.
- The co-ordinates of the node in the image file.
- The nodes to which this node is connected.

4.11 String Generation and Recognition(STRING.C)

This program takes as input the files FILE.NAM, FILE.NO, CONTEXT.TXT and the '.SYn' file giving node details as output by the SYNTAC program.

The connectivity information is used to generate a string representation of the character using the grammar defined in chapter 3.

The identification process is done by comparing the string with a series of strings which have a known interpretation. These strings are in file STRING.TXT.

A string representation is produced for the pattern. In some cases this is sufficient to uniquely identify the character; for example, the representation of the character 'E' is unique. The derived string is compared to the list in STRING.TXT. If the string corresponds to more than one character a more detailed analysis is required. The general type 'b' and 'c' nodes as described in chapter 3 are examined and converted into types b_1, b_2, b_1, etc. This requires the orientation of the characters to be taken into account; for example 'Z' is the same shape as 'N', but they are differentiated by the human observer because of the difference in their orientation. The new string is derived and again compared to STRING.TXT to see if the new string is unique. STRING may contain multiple strings for the same character. These strings may be due either to different shapes corresponding to the same character, or because the character needs sub-types of node types 'b' and 'c' for a full unique description. For example, the letter 'C' is represented by the string description '0-b-b-0' and also by '0-b_1-b_1-0'. The character 'N' is also represented by the '0-b-b-0' sequence. However, it is distinguished from the letter 'C' by the introduction of the sub-types to be represented by the string '0-b_2-b_1-0'.

There are three possible results of this final comparison:

- Successful unambiguous recognition of the character.
- Failure to identify the character.
- Incorrect recognition.

If the database of strings does not describe the pattern found the system fails to identify the pattern. If this occurs it may be due to the pattern not

corresponding to a character. A common problem is that two characters are connected, usually as a result of fixing bolts on the plate being directly between two characters; reference is made therefore to the file CONTEXT.TXT with a view to splitting the pattern into two separate patterns. This is done by the next program in the sequence SPLIT.C.

Finally a character may be incorrectly identified when a valid string representation of the character is produced which corresponds to a valid character which is not the right character. This may be caused by a poor quality image which cannot be resolved by the system; some characters cannot be identified even by the human observer.

Setting up the String Database: The database is a single file STRING.TXT. It is set-up using the idealised forms of the character set as described in the chapter 5. If a character is not recognized and the representation is not already found in the database it is incorporated into the database. In this way if the system is presented with a similar pattern in the future, it will be able to recognize it. Therefore, the performance of the system improves as it is presented with more patterns.

4.12 Character Separation (SPLIT.C)

The objective of this program is to consider patterns which have not been identified by the STRING program. The file CONTEXT.TXT is read. This contains details of the characteristics of the patterns. If the pattern width is twice that of identified patterns, it is likely that it contains two characters. This splitting is achieved by considering the number of BLACK pixels along lines which are parallel to the vertical edge of the plates within the central 50% of the pattern as shown below.

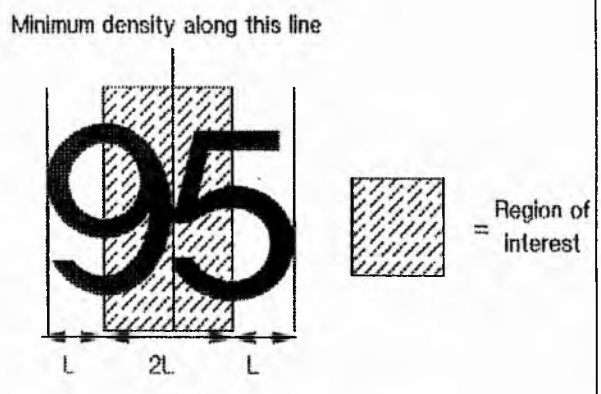


Figure 4.7 Character Splitting

The line which contains the minimum number is used to separate the two characters. It is anticipated that this line will be near the centre of the pattern, since the characters are usually the same width. However, this is not always the case; characters '1' and 'I' are always narrower than all other characters and 'W' and 'M' are usually wider. This method therefore yields better results than the method of simply splitting the pattern into two equal parts.

In order to recognise them, the two new patterns are again subject to the thinning, extension and recognition sequences.

4.13 Support Programs

The following programs perform image capture and printing.

a. Image Generation(READER.C): The configuration of the system is shown in figure 4.8.

The images used are digitized photographs. These are produced using a 'PCVISIONplus' frame grabber manufactured by Imaging Technology fitted into an IBM PC compatible computer(Imaging Technology 1987). It is a video digitizer and frame memory, capable of digitising standard RS-170/330 video input and storing the digitized images in an on-board frame memory. The digitized image is 512×512 pixels. Each pixel has an 8-bit intensity value. An image is stored in four 64K segments, each of which has a configurable base address.

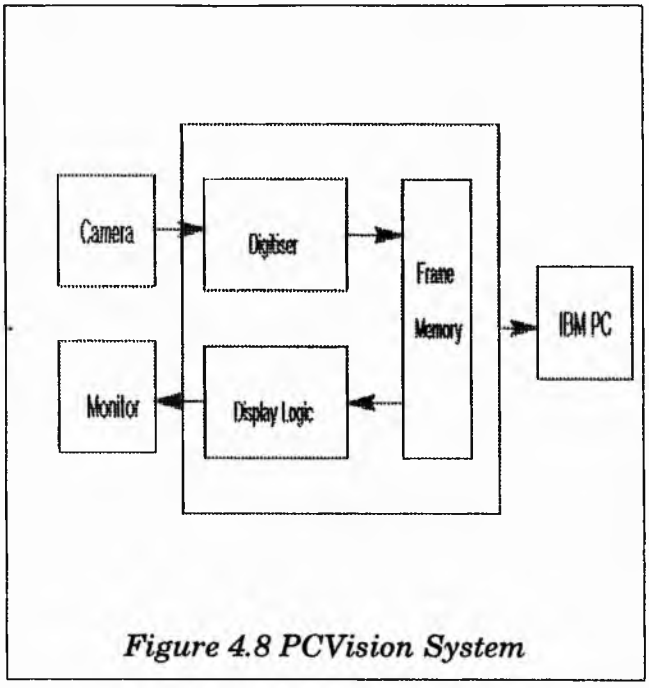


Figure 4.8 PCVision System

A simple program was developed to read the four segments and transfer the results into a 256K binary file, each byte representing the intensity value of a single pixel.

b. Image Display(SC2FI.C and FI2SC.C): Software is needed to capture images from the PCVISION monitor to file and to transfer those images from file to screen. The two programs SC2FI and FI2SC performed this function.

c. Image Printing(IMAGE2.C and IMAGE255.C): These two programs print the images shown in the chapter 5. IMAGE255 prints images with 8-bit intensity, while IMAGE2 prints binary images. Both programs use the Postscript 'IMAGE' operator to generate the images. The word processor WORD5 is able to incorporate encapsulated Postscript files into documents providing that scaling information is passed by using the 'Bounding Box' command. The images are printed by a DEC LNO3R Postscript laser printer.

4.14 Software Tools

All software is written using Microsoft 'QuickC' [Microsoft 1987] which includes comprehensive editing and debugging facilities. An extensive library of 'C' functions is also available. The compact memory model is used; a single code segment(CS), limited to 64K and multiple data segment(DS) each limited to 64K.

The majority of the development work is carried out using an IBM XT compatible computer with 640K of memory, a 32Mb disk and a monochrome EGA screen and MS-DOS V3.2 operating system. Some work is carried out using the Digital 'C' compiler on VAX computers with VMS operating system where it is found that the software is completely portable between the two systems, providing that the memory and stack size of the PC are taken into account.

CHAPTER 5

RESULTS AND DISCUSSIONS

CHAPTER 5

RESULTS AND DISCUSSIONS

In the experimental work one hundred images were considered. To illustrate the results, five images are considered in detail.

5.1. The Images and the Second Difference Metric

The images shown below are produced from the image files. They all have the same resolution and intensity variation. They are not photocopies of photographs since photocopies would not indicate the actual data on which the recognition system is operating.

Graphical representation of the second difference metric obtained for the five images under consideration is also given below for two group sizes. The peaks correspond to high values and these peaks correspond closely to the licence plate regions in the images. The full numerical results are tabulated in Appendix 2.

5.1.1. Image 1: Dark Red Vauxhall Cavalier

Figure 5.1 Image 1 512x512 Representation

The background is a typical street scene with brick buildings and sections of road and pavement. The headlights and windscreen are significantly brighter than the surrounding scene. Due to their large areas these regions are not likely to be confused with the plate region. In this scene the road is wet, but the variations in intensity caused by reflections are not sufficient to cause problems.

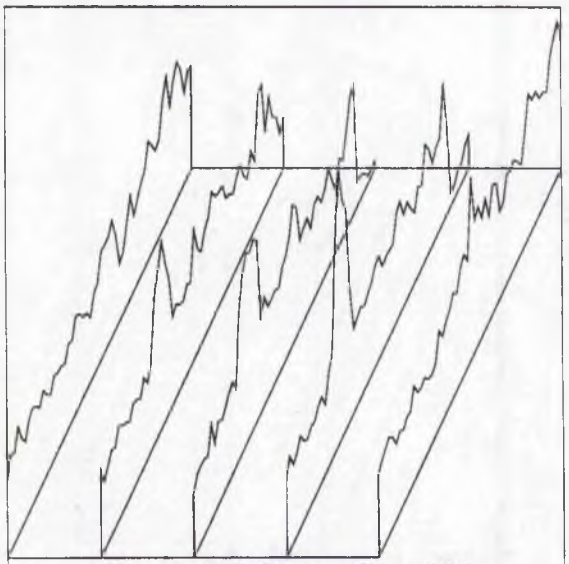


Figure 5.2 Group Size 100

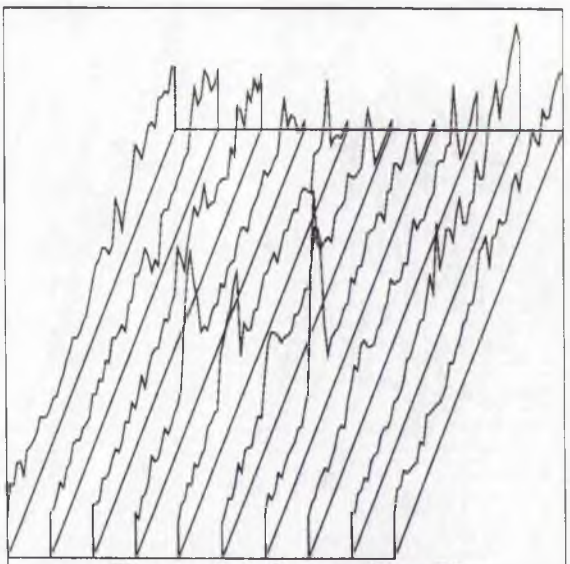


Figure 5.3 Group Size 50

5.1.2. Image 2: Light Red Ford Sierra

Figure 5.4 Image 2 512x512 Representation

The vehicle engine size is written to the left of the plate in silver letters, but the contrast between these letters and the background is too small to be a problem. The image presents two difficulties for the recognition process which are not apparent. The first is a small mounting bolt between the first '5' and the '8' which causes the numbers to be connected for some thresholds. The second is the indistinct shape of the letter 'W' which is due to an inadequate resolution of the image.

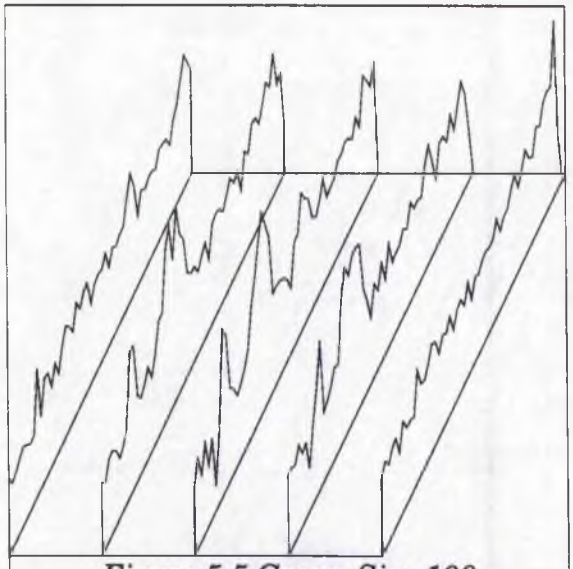


Figure 5.5 Group Size 100

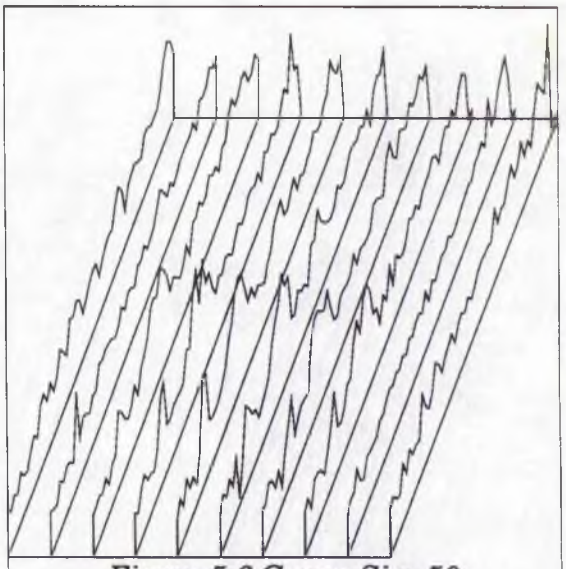


Figure 5.6 Group Size 50

5.1.3. Image 3: Brown Austin Metro

Figure 5.7 Image 3 512x512 Representation

The characters on the plate are very clear since both the car and the road are dark in contrast to the light coloration of the plate background. The rear windscreen is reflecting light and its interface with the car body forms a sharp intersection.

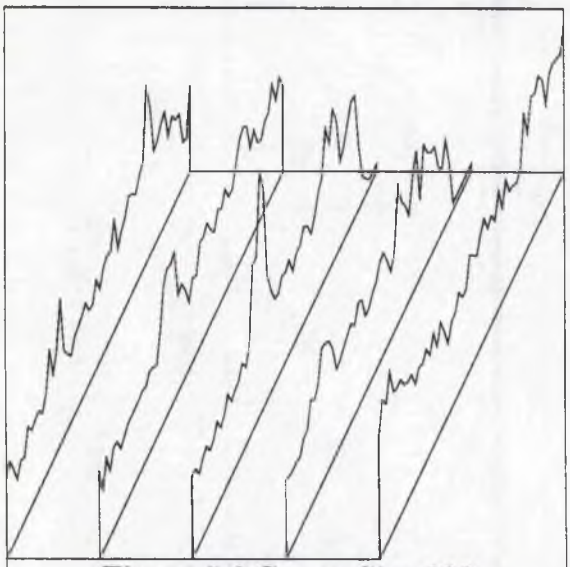


Figure 5.8 Group Size 100

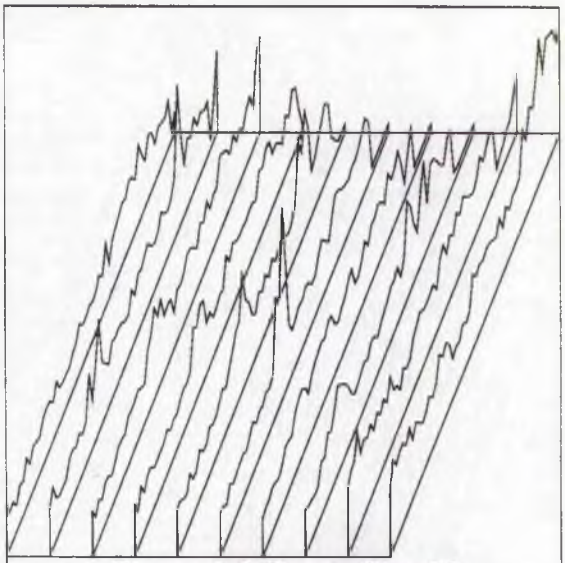


Figure 5.9 Group Size 50

5.1.4. Image 4: Light Blue Bedford Van

The main problem with this image is that the plate is dirty. This is not obvious to the human observer but it darkens the background of the plate and creates some difficulty in the recognition process. The mounting bolts do not cause a problem since although they are seen as a part of the characters for some threshold values, they do not cause characters to be joined.

Figure 5.10 Image 4 512x512 Representation

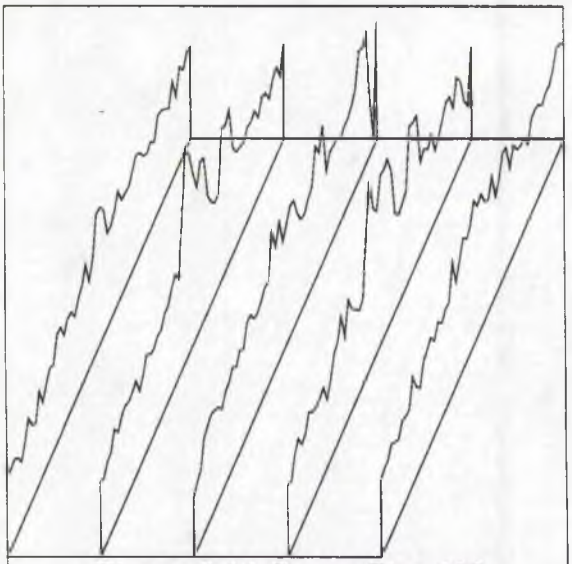


Figure 5.11 Group Size 100

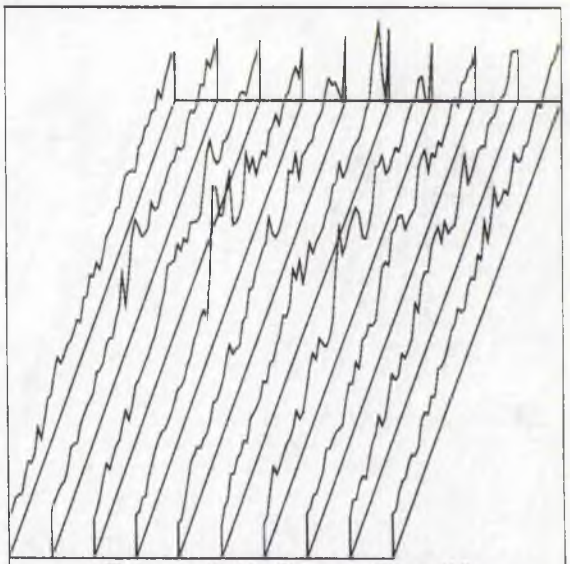


Figure 5.12 Group Size 50

5.1.5. Image 5: Red Ford Fiesta

Figure 5.13 Image 5 512x512 Representation

The main difficulty with this image is that there is insufficient contrast between the plate background and the car body. The rear of the vehicle is in shade. Humans are able to make use of colour as there is a clear colour interface between the plate and the car. However, the contrast in intensity between these two regions is not large. There is a dark line around the characters and the '3' is touching a manufacturers name on the bottom of the plate.

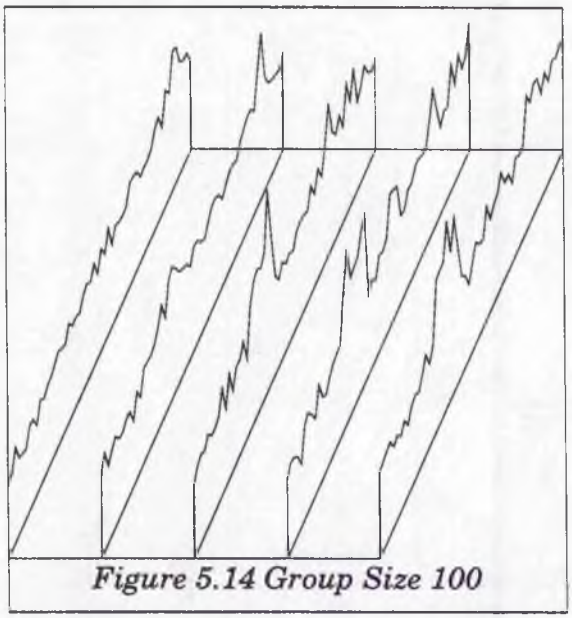


Figure 5.14 Group Size 100

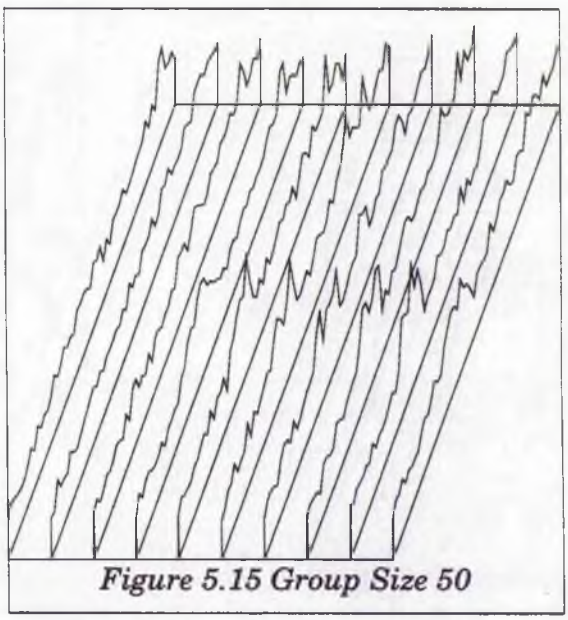


Figure 5.15 Group Size 50

5.2. Second Difference Metric

Appendix 1 tabulates the actual results for lengths of 50 and 100 pixels. The top 2% of the values obtained are highlighted. For all images there is a good clustering of the highest values around the plate region. The results obtained from using the longer summation metric when calculating the metric give a tighter grouping around the plate region. In several images, for example images 4 and 5, a length of 50 pixels gives rise to highlighted areas which are remote from the plate region. When a longer length is considered, these regions are not highlighted indicating a very localized area in which there is a large variation. In image 3 there is a small area in the top left of the image which gives metric values in the highest 2% for the shorter length but this area is not in the top 20% when the summation length is doubled. The table below shows the optimum length for the plates in order to yield the highest possible metric value. All lengths are in pixels

Image	Actual plate length	Optimum metric scan length
1	200	115
2	280	161
3	200	115
4	230	132
5	290	167

Figure 5.16 Actual and Optimum Scan Length

Optimum plate length = 0.577 × actual plate length. This is derived in section 3.1.2.

The metric calculated with a length of 100 is far closer to the optimum than the shorter length and is expected to yield a better result. However, the metric does not exhibit a great sensitivity to variations in length. In the worst case, which is image 5, half of the top 2% of values are outside the plate region in the top right of the image. The metric is not deemed to have failed if it highlights areas outside of the plate region, since its purpose is to indicate regions of the image

which can be subject to further study. The area outside the plate region which yields a high metric value is quickly rejected by later stages of the analysis which then directs effort towards other likely areas of the image.

The average value of the top 2% of values obtained for the images is in all cases significantly greater than the average value of the remaining 98%, in all cases more than twice the value and in most cases more than three times greater. These results indicate a high degree of correlation between the metric and the plate region for the images considered.

5.3. Image Binarization

Using the second difference metric a global threshold is derived as described in earlier chapters. The table below shows the value of this global threshold for two different group sizes.

Image	Threshold Values	
	Group size 100	Group size 50
1	193	143
2	168	168
3	173	170
4	137	140
5	158	155

Figure 5.17 Group Sizes and Threshold Values

Three groups of binary images are shown below. Two are produced using the global thresholds given above. The third is generated using the locally adaptive method of Giuliano(1977]. This algorithm has the effect of highlighting edges which are shown in black.

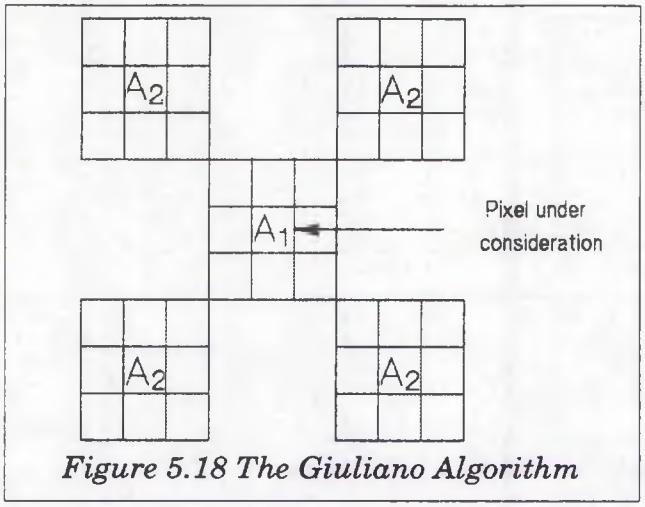


Figure 5.18 The Giuliano Algorithm

In the Giuliano five neighbouring regions influence the threshold value used for a particular pixel. The regions consist of a 3×3 template around the pixel under consideration and four diagonally opposite 3×3 regions as shown below. The algorithm is described in detail in section 2.4.1.

5.3.1. Binarized Images 1 - 5

Image 1

Global Threshold 193

Global Threshold 143

Adaptive Threshold

Image 2

Global Threshold 168

Global Threshold 168

Adaptive Threshold

Image 3

Global Threshold 173

Global Threshold 170

Adaptive Threshold

Image 4

Global Threshold 137

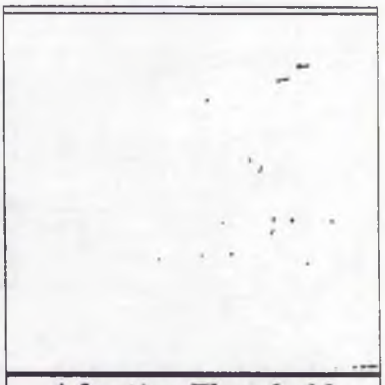
Global Threshold 140

Adaptive Threshold

Image 5

Global Threshold 158

Global Threshold 155



Adaptive Threshold

Global binarization is not suitable in situations where a-priori knowledge is not available. Its great advantage over locally adaptive methods is that it is much faster to perform. The implementation of the Giuliano algorithm produced

unexpected results. The edges of the images are emphasized and are displayed as dark areas, while both light and dark areas of the images are shown as white.

An important aspect of an adaptive algorithm is that it does not require a-priori information about the image on which it is to operate. However, although this is an adaptive algorithm it requires five constants to be assigned; each of these alters the value of the threshold calculated for a pixel. Giuliano gives no guidance as to how these parameters are to be chosen. Choosing a set of parameters which performs well with one image may result in poor performance with other images. This is borne out by the set of images in which the five parameters remain unchanged. Some produce clear outlines of the plate and the characters on it, while others show no discernible features. The purpose of the global binarization is not to produce a clear overall image, but simply to produce a clear picture of the characters on the plate region. In all cases this was achieved and both the edges of the plates and the characters on them are clearly visible. As anticipated the edges of the individual characters are not completely straight; this is due to the resolution of the camera system and the binarization process. On real digitized images it is inevitable that edges are not always 'step' but have a 'ramp intensity' profile. However, this can be dealt with satisfactorily by the later stages of the analysis which are able to cope with edges of this type.

5.4. Region Growing

The results of the first region growing process are shown below. The purpose here is to erase all detail from the images except for a white plate region with black alphanumeric characters on it. The characters are clearly identified and the remainder of the image has been deleted. Completely contained background areas such as found in the '9' and '0' are not shown. In addition, the background is shown in the same intensity as the characters. These two problems are resolved in the next stage of the process.

Image 1

Image 2

Image 3

The outline of the plate is only important in that it helps to determine if the plate region has been correctly identified. In all images except image 5 the outline of the plate is complete. However, even Image 5 has a substantial amount of the plate visible, adequate to confirm that it is a plate. A more important aspect of the images is the clarity of the characters. Ideally each character should not be touching either its neighbours or the black background of the image.

Image 4

Image 5

There are a total of 35 characters. Of these, two pairs are touching. They are the '5' and '8' in image 2, the '9' and the '5' in image 5. The connection in images 2 and 5 is caused by a mounting bolt on the plate directly between the characters. These features can be clearly seen in the representations of the original images given earlier in this chapter. In addition, there are 2 characters which are touching the background. They are the '3' and 'U' in image 5. This is caused by a combination of two factors. The plate is indented, and therefore in shadow. The vehicle is bright red which in terms of pixel intensity is similar to the yellow background of the plate. There is a black line around the characters, which reduces the thickness of the border around

the characters. The central '3' in the image has a manufacturer's name on the plate touching it, this in turn touches the edge of the plate, linking the character with the region outside the plate.

5.5. Rectangle Recognizer

The program follows the region of the plate and identifies its four sides. In images 1, 3 and 4 all four sides of the plate are clearly and unambiguously identified. The orientation of the plate calculated corresponds closely to the actual orientation.

Image 5 is less clear since the right of the plate is not found and the bottom vector is split into two regions. However, the top of the plate in this image is readily found and is parallel to the two vectors which correspond to the bottom edge. These in turn are at right angles to the left side of the plate. Although the results do not correspond to the ideal of four vectors which can be paired on the basis of their direction and magnitude, there is sufficient information to indicate that this likely to be a plate region.

Image 2 has all four sides of the plate identified but in addition, there is a small connected region below the left bottom corner. There is still sufficient information to indicate that this is a plate region since the required two pairs of vectors can be found from among the vector of the outline.

5.6. Plate Removal

This stage removes the plate background and leaves the characters as black on a white background. It also resolves the problems of the enclosed 'holes' in characters.

The results are shown below.

Image 1

Image 2

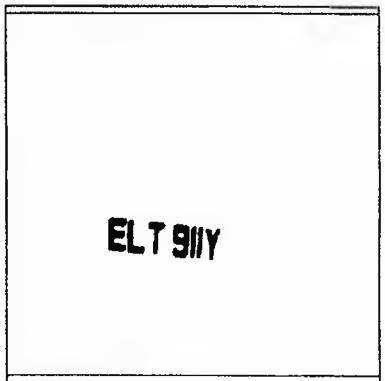


Image 3

Image 4

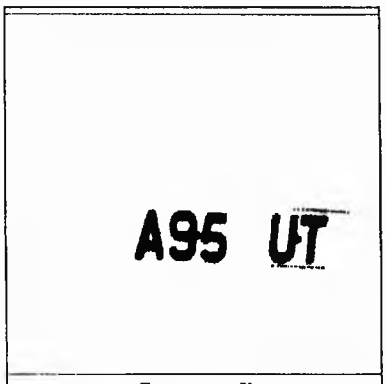


Image 5

In all cases the characters are clearly displayed. Images 4, and 5 also have several small groups of black pixels which do not correspond to characters. In image 4 these correspond to additional writing on the plate, which gives the manufacturer's name. In image 5, the marks correspond to a black outline around the characters. The detection of these shows the sensitivity of the processing so far carried out. These collections of pixels can be discarded easily at the next stage when pixel groups below a certain size are ignored.

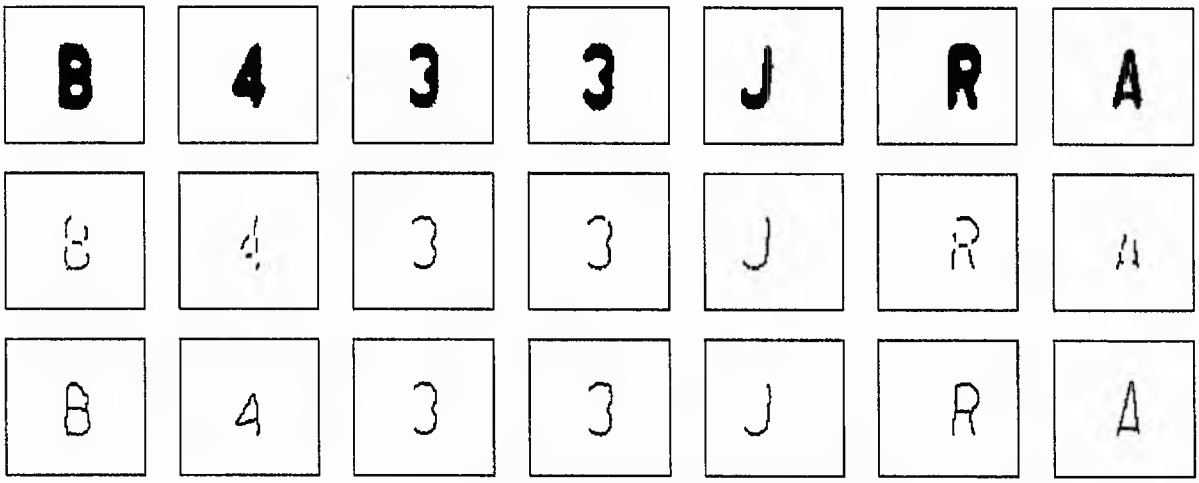
5.7. Character Extraction

The objective here is to separate the group of black characters on a white background, so that each character is in a separate file. Groups of pixels which are below a threshold size are ignored.

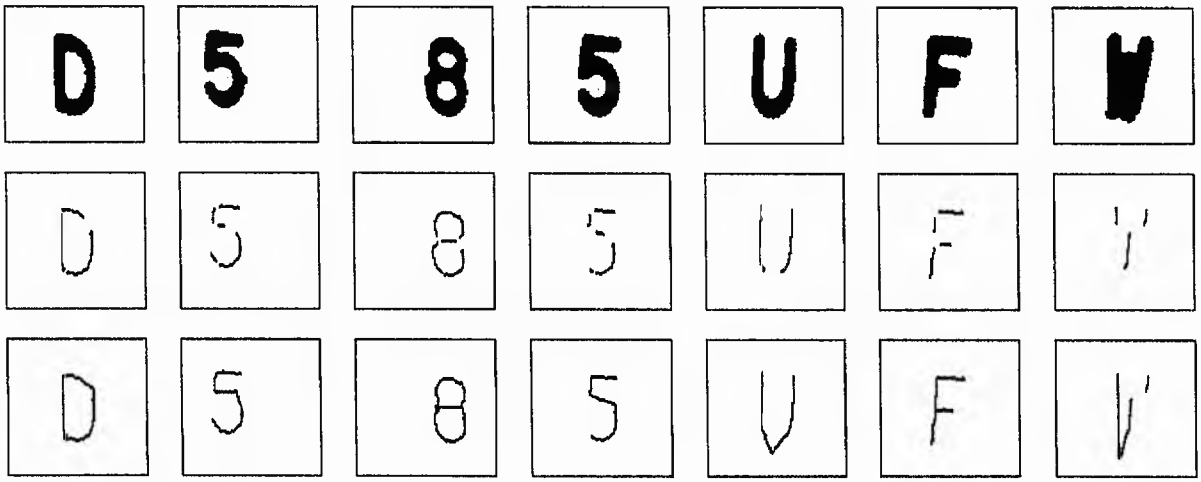
Three forms of each character are shown:

- The solid, unthinned form of the character.
- The thinned form - without extending the nodes into the strokes.
- The thinned forms - with extension of strokes into the nodes to form connected skeletons.

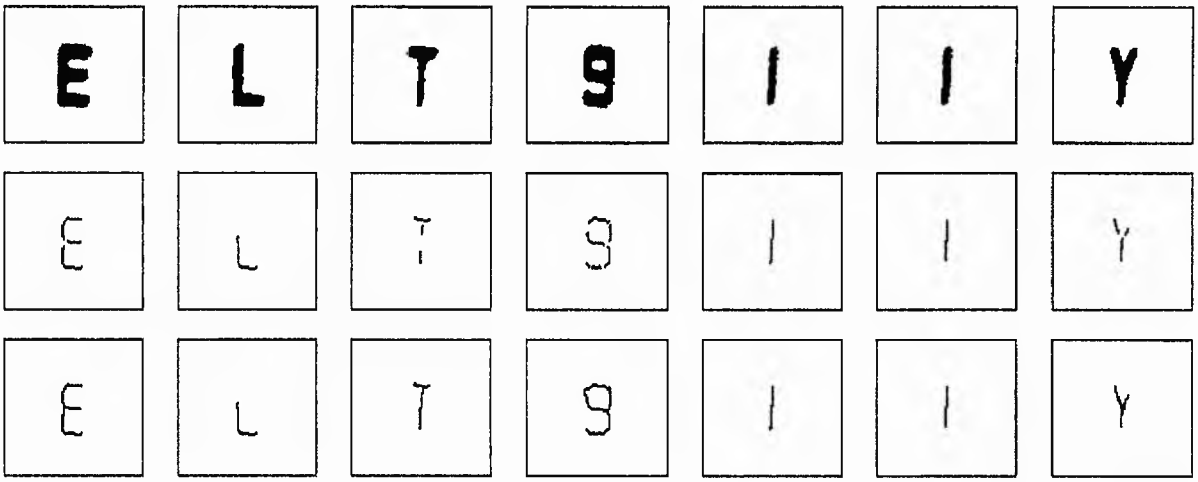
5.7.1. Image 1 - Extracted Characters



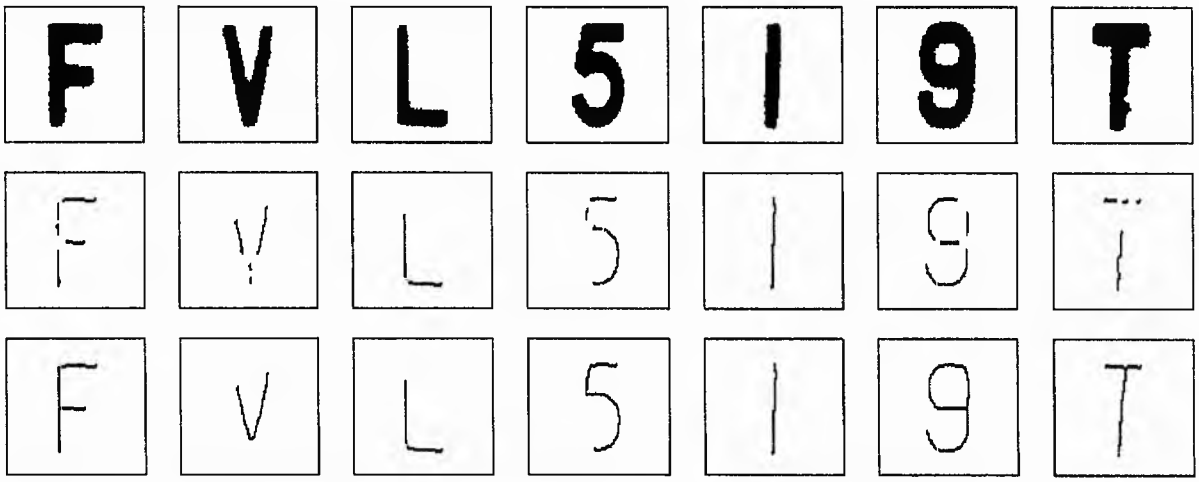
5.7.2. Image 2 - Extracted Characters



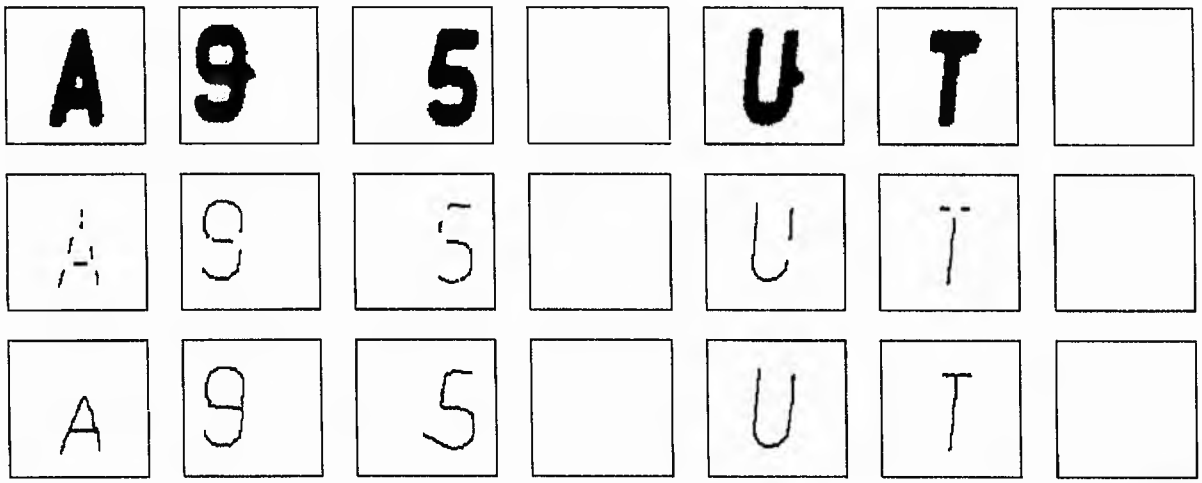
5.7.3. Image 3 - Extracted Characters



5.7.4. Image 4 - Extracted Characters



5.7.5. Image 5 - Extracted Characters



In all cases the characters are clearly recognizable to the human observer with the exception of the character 'W' which is found in image 2. This is largely due to a lack of resolution in the original image file. In cases where there is an irregularity in the character line this can usually be traced back to the form of the image file. In the 'T' in image 4 and the 'U' in image 5, there are slight defects in the characters. This is due to mounting bolts, which can be seen in the original images.

The thinning and stroke extension algorithms produce clear 8-connected skeletons. The only exception is 'W' in image 2. The unthinned form is not a clear representation of the characters and the thinned form is not recognizable. The characters '3' and 'U' in image 5 have not been detected.

5.8. Context Checking

The context checker collects information about the extracted characters which is used later. The basic rectangle is derived for each pattern and output to a file along with the position of each pattern in the original 512×512 image file. Connecting the mid points of the basic rectangles gives the horizontal axis of the plate. The calculation of an axis at right angles to it defines a co-ordinate frame for the characters in the original image. For the five images considered here the frame is within 4 degrees of the frame as measured from the images.

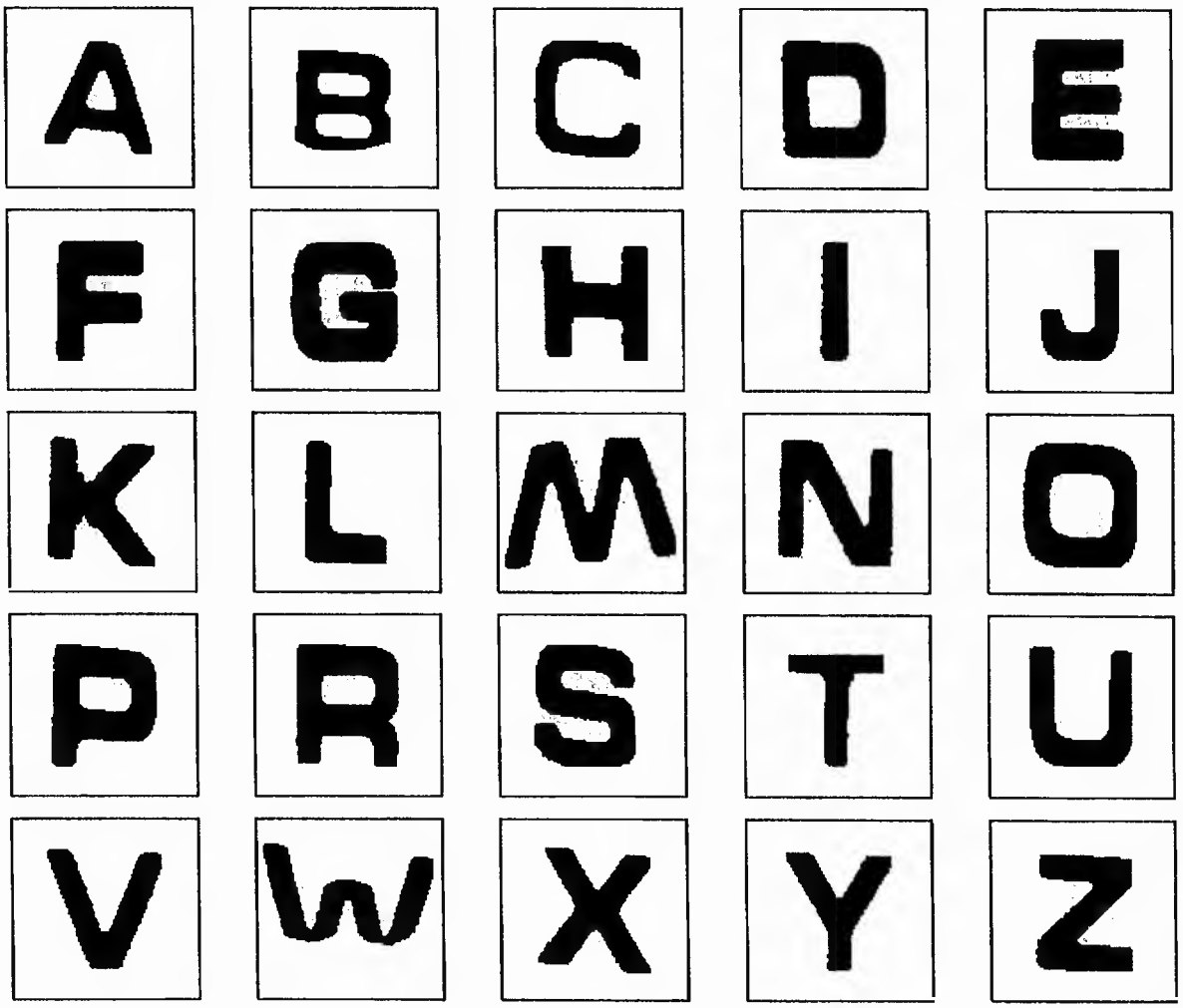
The context checking confirms whether or not that the extracted objects are spatially distributed in a format which correspond to that expected for a row of characters on a licence plate. It also enables a pair of touching characters to be separated.

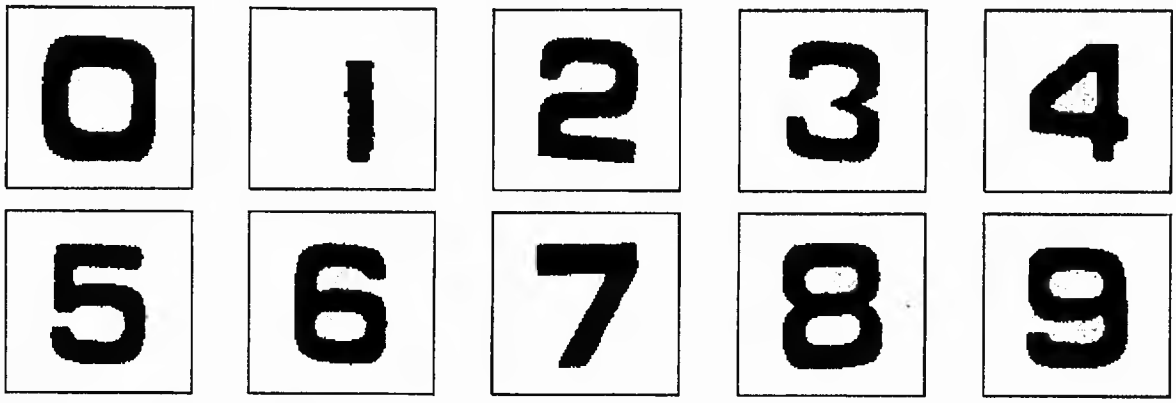
5.9. Stroke Skeleton Generation

After the individual characters are extracted, they are thinned. The complete alphanumeric set of thin and thinned characters is shown below. The characters are generated by digitizing images of characters, rather than by using an idealized form with parallel sides and regular features, which are very special forms of the characters.

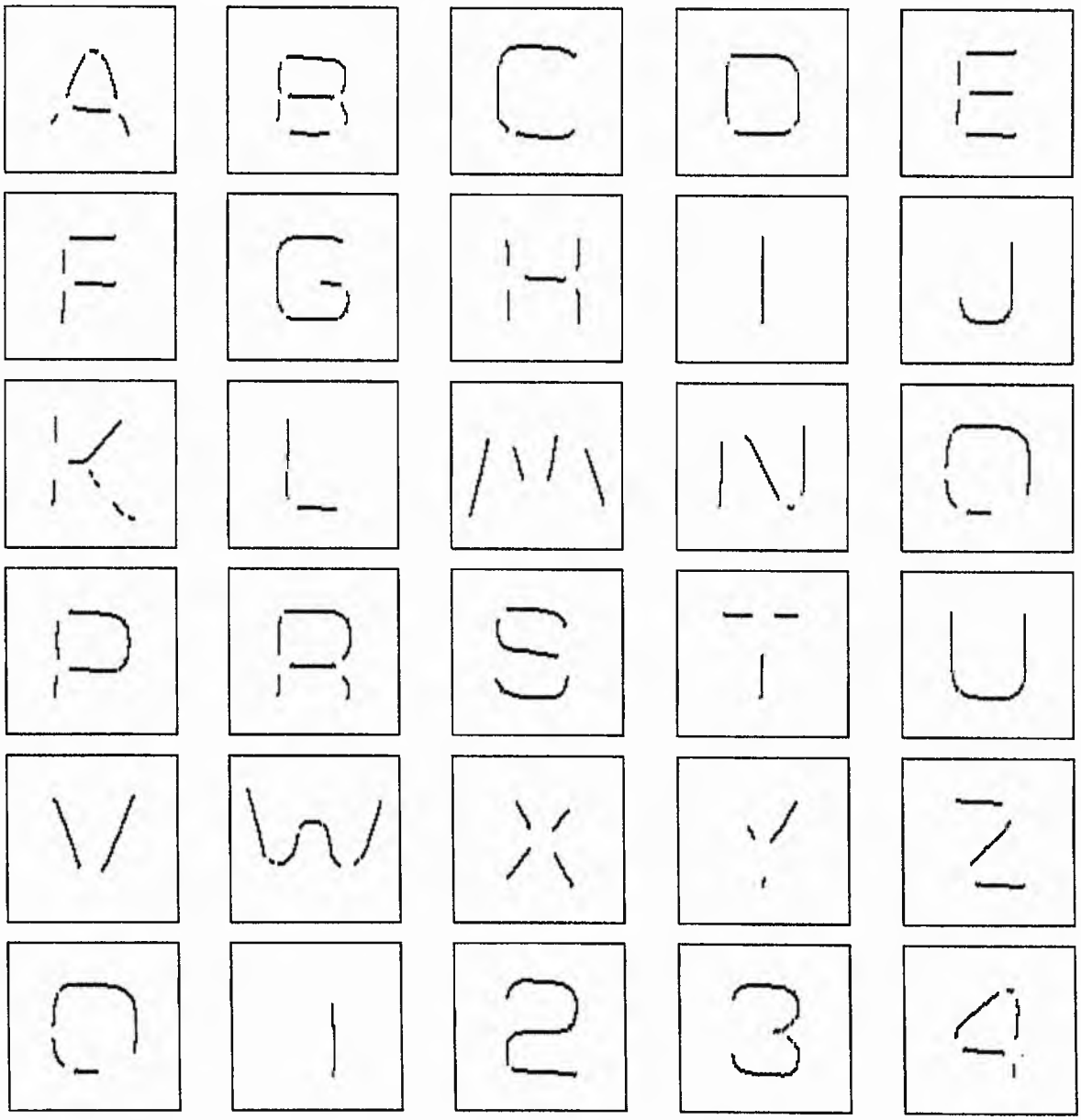
Three forms of each character are shown:

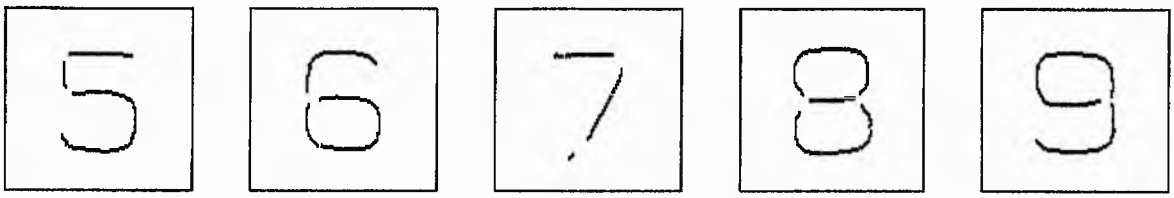
5.9.1. Unthinned Characters



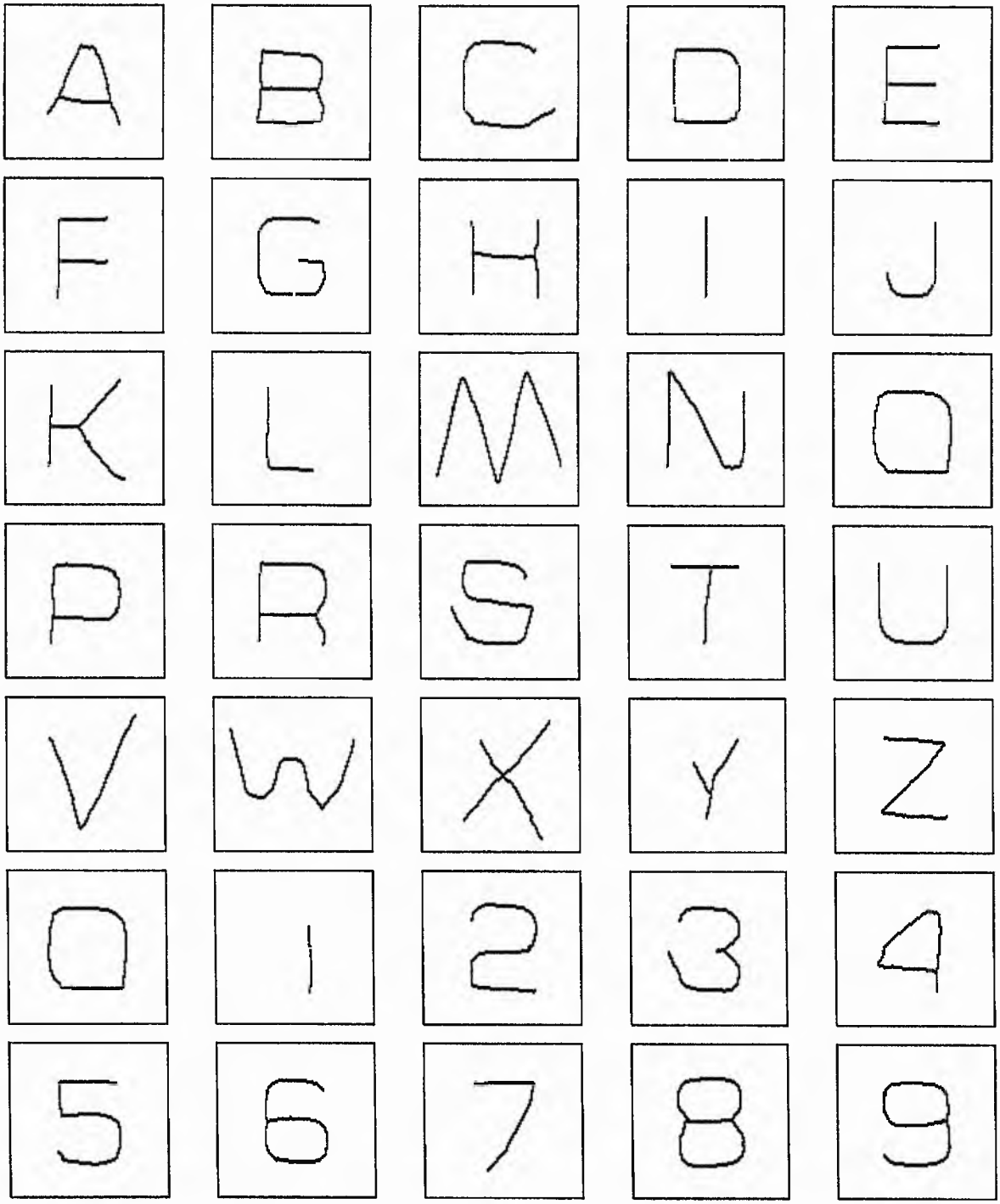


5.9.2. Thinned Forms - Without Stroke Extension





5.9.3. Thinned Forms - With Stroke Extension



In all cases the final skeletal forms are clear representations of the original characters despite a far from regular border. The skeletons are clearly legible to a human observer.

5.10. Syntactic Representation of Characters

After thinning, string representations of the characters are generated. The nodes for an idealized alphanumeric character set and the representation of each member expressed as a pattern string are shown below. In a few cases, for example 'E', there is more than one representation.

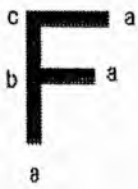
a-b-((c-b)(b))-a*
|
a-b-((b)(c-b))-a*

(b-b-c-c)+(b-b-c-c)

a-c-c-a
|
a-c_2-c_2-a

c-c-c-c

a-c-b-((a)&(c-a))
|
a-c-b-(((c-a)&(a))
|
a-b-((c-a)&(c-a))

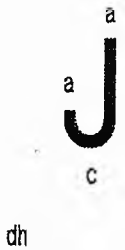


a-c-b-((a)&(a)) |
a-b-(c-a)&(a))
|
a-b-((a)&(c-a))

a-c-c-c-c-a
|
a-c_2-c_2-c_2-c_2-a

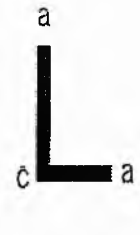
a-b-((a)&(b-(c)&(c)))

a-a



a-c-a
|
a-c$_1$-a

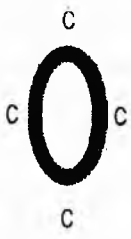
a-b-((a)&(b-(c)&(c)))



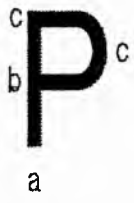
a-c-a
|
a-c$_2$-a

a-c-c-c-a
|
a-c$_1$-c$_2$-c$_1$-a

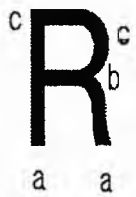
a-c-c-a
|
a-c$_1$-c$_2$-a



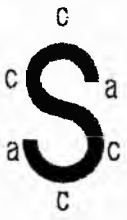
c-c-c-c



a-b-((c)(c-c))*
|
a-b-((c-c)(c))*



a-b-((b)(c-c-b))-a*
|
a-b-(c-c-b)(b))-a*



a-c-c-c-c-c-a
|
a-c$_2$-c$_2$-c$_1$-c$_1$-a

a-b-((a)&(a))
|
a-b$_2$-((a)&(a))

a-c-c-a
|
a-c$_2$-c$_2$-a

a-c-a
|
a-c$_2$-a

a-c-c-c-c-a
|
a-c$_2$-c$_1$-c$_2$-a

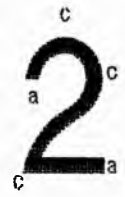
a-d(a&a&a)

a-b-((a)&(a))
|
a-b$_1$-((a)&(a))

a-c-c-a
|
a-c$_1$-c$_2$-a

c-c-c-c

a-a



a-c-c-c-a
|
a-c$_1$-c$_2$-c$_2$-a

a-c-c-b-((a)&(c-c-a))
|
a-c-c-b-((c-c-a)&(a))

a-b-((c)(c))*

a-c-c-c-c-a
|
a-c$_1$-c$_2$-c$_1$-c$_1$-a

a-c-b-((c)(c-c-c))*
|
a-c-b-(c-c-c)(c))*

a-c-a
|
a-c$_1$-a

(b-b-c-c)+(b-b-c-c)

a-c-b((c)(c-c-c))*
|
a-c-b-((c-c-c)-c

5.11. Final Recognition Results

The strings generated for the extracted characters are given below. In cases where a string is generated which is different from the idealised string, it can be included in the database providing that it does not clash with an existing string.

	String Representations	Identified Character	Actual Character
Image 1	(b-b-c-c)+(b-b-c-c)	B	B
	(a-b-((c)*(c))	4	4
	(a-c-c-b-((c-c-a)&(a))	3	3
	(a-c-c-b-((c-c-a)&(a))	3	3
	a-c_1-a	J	J
	a-b-(c-c-b)*(b))-a	R	R
	a-b-((b)*(c-b))-a	A	A
Image 2	c-c-c-c	D	D
	a-c_1-c_2-c_1-c_1-a	5	5
	(b-b-c-c)+(b-b-c-c)	8	8
	a-c_1-c_2-c_1-c_1-a	5	5
	a-c_2-c_2-a	U	U
	a-c-b-((a)&(a))	F	F
	a-c_2-c_2-a	V	W
Image 3	a-c-b-((a)&(c-a))	E	E
	a-c_2-a	L	L
	a-b_2-((a)&(a))	T	F
	a-c-b-(c-c-c)-c	9	9
	a-a	1	1
	a-a	1	1
	a-b_1-((a)&(a))	Y	Y
Image 4	a-c-b-((a)&(a))	F	F
	a-c_2-a	V	V
	a-c_2-a	L	L
	a-c_1-c_2-c_1-c_1-a	5	5
	a-a	1	1
	a-c-b-(c-c-c)-c	9	9
	a-b_2-((a)&(a))	T	T
Image 5	a-b-((b)*(c-b))-a	A	A
	a-c-b-(c-c-c)-c	9	9
	a-c_1-c_2-c_1-c_1-a	5	5
	none	-	3
	a-c_2-c_2-a	U	U
	a-b_2-((a)&(a))	T	T
	none	-	U

Figure 5.19 Results of the Recognition System

33 out of 35 characters are successfully extracted. 32 of these are correctly identified. Two characters are not detected since they are connected to the region outside the plate after the image binarization and are therefore not detected as characters. A single character 'W' in image 2 is not correctly identified. Of the 33 extracted characters there are 2 connected pairs. All of these pairs were successfully separated and identified.

5.12. Separation of Connected Characters

Three pairs of characters were connected. They are submitted to the 'SPLIT' program not having been successfully identified by the recognition process. In all cases the pairs of characters are successfully separated and re-submitted to the recognition process.

CHAPTER 6

CONCLUSIONS

CHAPTER 6

CONCLUSIONS

The recognition of alpha-numeric characters in unconstrained environments is a complex problem which requires a wide range of image processing algorithms. The objective of the work here was to develop new approaches to this aspect of character recognition.

The application chosen to illustrate the experimental work is the reading of vehicle licence plates. However, many of the techniques used and principles examined are equally applicable to a wide range of image recognition systems.

New algorithms were developed which cover the entire recognition process from the finding of the licence plate through to the identification of the characters.

The performance of the algorithms was rigourously tested using a wide range of input images. The behaviour of five of these images was discussed in detail and the results shown at each stage of the recognition process.

The system developed runs quickly, using standard IBM PC compatible machines and has shown itself to be capable of reading the vehicle licence numbers with a high degree of success.

6.1 Conclusions

A major problem in image processing is that each image contains such a large amount of information. The complexity of the problem is surprising since humans perform the recognition process with such ease. At present there is no vision processing system available which is capable of examining an image and identifying all the objects it contains except in a very constrained environment. The system developed recognizes this and at each stage of the analysis reduces

both the complexity and the size of the image being processed in order to extract the specific information required.

A metric was developed which allows the licence plate region to be identified by using known characteristics of the region. The use of metrics has been proved to be successful. This technique is very rapid compared to alternative methods which attempt to extract patterns and compare them against a database. The second difference metric performed well and allowed a global threshold to be calculated and the image to be binarized.

Global binarization was very rapid to execute and for this application yielded superior results to the more complex and time consuming adaptive thresholding algorithms. If a-priori knowledge is available global binarization often yields excellent results.

The information reduction process continued with region growing which yielded black characters on a white background. The extraction of characters was achieved by a novel single pass technique, which proceeded rapidly and was straightforward to implement.

The identification of the characters is seen to highlight many problems with conventional recognition algorithms. For unconstrained characters to be recognized by techniques such as comparison against templates, extensive normalization of the character for size, position and orientation is required. However, even by the use of such techniques, problems still arise since there is a wide variation in the form of the patterns that are recognized as representing a particular character. For these reasons a novel approach was adopted, which expressed the patterns in terms of the connectivity of their strokes. In order to do this it is necessary to skeletize the patterns. A study of existing skeletizing or thinning algorithms revealed a series of problems which caused distortion to the character, for example, the relative thickness of the strokes. For characters which are thick, i.e. with a height:stroke ratio of less than about 8:1, the existing

algorithms required many passes and were slow to execute. The patterns produced are often not close to those which are produced by a human. In the light of these difficulties a new thinning algorithm was developed which operated quickly and produced clear representations of the patterns. It is most suitable for patterns with thick strokes.

The syntactic approach used is a viable method for the identification of characters. The grammar is based upon the intersection of strokes and is independent of size, position, orientation, and stroke thickness.

The system developed considers the entire image recognition problem from beginning to end rather than considering a single aspect of it. As a whole, the system performed well with a high proportion of the characters being correctly identified. The speed of the system is superior to comparable systems.

6.2 Future Developments

The aim of this work was not to produce an excellent user interface, or to optimise the speed of execution. The design of the software was intended to demonstrate each stage of the image processing. The performance of the system could therefore be increased by enhancing the design of the software and using a faster processor.

The recognition process lends itself to parallelism and the greatest improvements would be achieved by using a parallel processing system.

The use of cameras for identifying traffic offenders is becoming more popular. Such a system coupled with a vision processing system of the type described could readily produce an automatic vehicle identification scheme. The success rate of a system of this type could be enhanced by triggering the camera using, for example, pneumatic tube vehicle detectors in the road. In these circumstances the position of the vehicle would be more constrained. A series of

images taken with the same background would enable the metrics used for finding the region of interest to be enhanced and therefore their performance to be increased. The system also has the potential to be used for charging for tolls for road usage.

A major problem of vision processing is the interpretation of different patterns which represent the same object. The syntactic approach used provides a technique for expressing the structural form of a pattern and thereby allowing it to be identified. This is a powerful technique and extension of the work on syntactic pattern recognition is likely to be an area of fruitful research.

Neural nets are being used for character recognition. These techniques could be combined with the syntactic approach by using the neural net to recognize the nodes of characters. After node identification the pattern string could be generated as described.

BIBLIOGRAPHY

BIBLIOGRAPHY

Aho [1972]; Aho A; Peterson T.

A minimum distance error correcting parser for context free languages.

SIAM Journal of Computing. Dec 1972 Vol 4.

Ali [1977]; Ali F; Pavlidis T.

Syntactic recognition of handwritten numerals.

IEEE Trans Systems MAN Cybernetics. SMC 7 1977 .

Al-Dabass [1981]; Al-Dabass D; Moualed R.

A Vision Algorithm for Distance and Orientation Measurements.

Proc IASTED International symposium on Modelling, Identification and Control, Davos Feb 1981

Al-Dabass [1985]; Al-Dabass D;

Characteristics Estimation of Point Objects using Stereo Vision

IEE Colloquium on Digital Signal Processing, London December 1985.

Aoki [1979]; Aoki M.

Rectangular Region Coding for Image Data Compression

Pattern Recognition 1979 Vol 11.

Arcelli [1975]; Arcelli C; Cordella L.

Parallel thinning of binary pictures.

Electron Lett. 1975 Vol 11 no 7.

Arcelli [1981]; Arcelli C; Cordella L.

From local maxima to connected skeletons

IEEE Trans pattern Analysis machine Intelligence. PAMI 3 1981.

Badie [1980]; Badie K; Shimura M.

Feature extraction and primitives.

Paper from - pattern recognition in practice. ed. Gelsema E.S. 1980.

Badii [1983]; Badii F; Peikari B.

Invariant numerical shape modeling

Paper from proceedings of IEEE computer society conference on computer vision and pattern recognition. June 1983. Washington.

Ballard [1982]; Ballard D.H; Brown M.

Computer Vision.

ISBN 0-13-165316-4.

Blesser [1973]; Blesser B. A.

Character recognition based on phenomenological attributes.

Visible Language 1973 Vol 7[3].

Blum [1973]; Blum H.

Biological shape and visual science.

Journal theoretical biology. Part 38, 1973.

Blum [1978]; Blum H; Nagel H.

Shape recognition using weighted symmetric axis features.

Pattern recognition Vol 10.

Boyle [1988]; Boyle R.D; Thomas R. C

Computer Vision - A First Course.

Blackwell Scientific Publications. ISBN 0-632-01577-2.

Bribiesca [1979]; Bribiesca E; Guzman A.

How to describe pure form and how to measure differences in shapes using shape numbers.

Paper from conference on pattern recognition and image processing. Aug 1979.

Brzeski [1987]; Brzeski M.

Using the EGA card for colour graphics.

.EXE Jan 1987.

Burgess [1984]; Burgess R. S

An introduction to program design using JSP.

ISBN 0-09-154961-2.

Burr [1980]; Burr D. J.

Designing a Handwriting Reader.

Proc of 5th International Conference on Pattern Recognition. 1980.

Calabi [1968]; Calabi L.

Prairie fires, convex deficiencies and skeletons.

Am. Math. Mon. 1968 Vol 75.

Canny [1983]; Canny J.

A variational approach to edge detection.

Paper from National Conference on AI. Aug 1983. AAAI-83.

Canny [1986]; Canny J.

A computational approach to edge detection

IEEE trans on Pattern analysis and machine intelligence.

Chien [1983]; Chien C.H; Aggarwal J.K.

A normalised quadtree representation.

Paper from proceedings of IEEE computer society conference on computer vision and pattern recognition. June 1983. Washington.

Clancy [1985]; Clancy M. J; Dawson J.; et al

Electronic Road pricing in Hong Kong.

Proceeding PTRC Annual Meeting 1985.

Cox [1982]; Cox C; Coveignoux P.

Skeletons: A link between theoretical and physical letter descriptions.

Pattern recognition 1982, Vol 15.

D'Amato [1982]; D'Amato D; Pintsov L; Hoay H.

High speed pattern recognition system for alphanumeric hand printed characters.

Proc IEEE pattern recognition and image processing Conf. Las Vegas 1982.

Davies [1989]; Davies P; Ayland N.

Automatic vehicle identification for heavy vehicle monitoring.

IEE Conference on Road Traffic Monitoring 1989

Davis [1974]; Davis. L. S.

A survey of edge detection techniques.

Computer graphics and image processing Vol 4 1974.

Deravi [1983]; Deravi F; Pal S.K.

Grey level thresholding using second order statistics.

Pattern recognition letters 1 1983.

Deutsch [1972]; Deutsch E. S.

Thinning algorithms on rectangular hexagonal and triangular arrays.

A.C.M. 1972 Vol 15 part 9.

Eager [1990]; Eager J.

Character Building

PC Magazine (UK Edition) June 1990 Vol 3 Issue 6

Eden [1962]; Eden M.

Handwriting and pattern recognition.

IRE Transactions on Information Theory. Vol IT-8. 1962.

Fischler [1983]; Fischler M.A.

Perceptual organisation and curve partitioning.

Paper from proceedings of IEEE computer society conference on computer vision and pattern recognition. June 1983. Washington.

Foote [1981]; Foote R. S.

Prospects for non-stop toll collection using automatic vehicle identification.

Traffic Quarterly 1981. p.35.

Freeman [1961]; Freeman H.

On the Encoding of Arbitrary Geometric Configurations.

IRE Trans on Electronic Computers EC-10. Part 2. 1961.

Fu [1980a]; Fu K. S.
Syntactic pattern modelling using stochastic tree grammars.
Computer Graphics image proc. 1980 Vol 12.

Fu [1980b]; Fu K. S.
Syntactic models in pattern recognition and applications.
Paper from - pattern recognition in practice. ed. Gelsema E.S. 1980.

Fu [1986]; Fu K. S.
Syntactic pattern recognition
Paper from handbook of pattern recognition and image processing. ed. Young T.Y. 1986.

Giuliano [1977]; Giuliano E; Paitra O.
Electronic Character reading system.
U.S. Patent No. 4-047-152. 1977.

Gosch [1988]; Gosch J.
Philips moves to grab automatic toll market.
Smart card Monthly, 1, 1988.

Grimsdale [1959]; Grimsdale R; Sumner F; Tunis C; Kilburn T.
A system for the automatic recognition of patterns.
IEE Vol 106 part B No. 26. March 1959.

Haralick [1973]; Haralick. R. M.
Textual features for image classification
IEEE Transaction. Man. Cybernet. 3

Haralick [1984]; Haralick R.M.
Digital step edges from zero crossing of second directional derivatives.
IEEE trans pattern Analysis Machine intelligence Vol 6 No.1 1984.

Heuckel [1971]; Heuckel M.

An operator which locates edges in digital pictures.

Jnl. Ass Comput. Mach. 1971 Vol 18.

Hilditch [1969]; Hilditch C.

Linear Skeletons from Square Cupboards.

Machine intelligence IV. Eds. Meertzer B; Michie D. University Press. Edinburgh 1969.

Hills [1989]; Hills P.J; Blythe P.

Paving your way

IEE Review November 1989

Himmel [1976]; Himmel D.P.

Some real world experiences with handwritten optical characters.

Proceedings of IEEE international conference on Cybernetics and Society, Washington 1976.

Imaging Technology [1987]; Imaging Technology

PCVISIONplus Frame Grabber User Manual

Jackson [1983]; Jackson M.

System Development.

Prentice Hall International. ISBN 0-13-880328-5

Jones [1989]; Jones G.

Accustomed to your face.

Computer Weekly May 25 1989.

Kashyap [1986]; Kashyap R.

Image models

Paper from handbook of pattern recognition and image processing. ed. Young T.Y. 1986.

Keng [1976]; Keng J; Fu K

A syntax directed method for land-use classification of LANDSAT images.

Proceedings of Symposium of Current Mathematical Problems in Image Science. Monterey California, Nov 1976.

Kim [1983]; Kim Y. C; Aggarwal J.K.

Rectangular coding for binary images.

Paper from proceedings of IEEE computer society conference on computer vision and pattern recognition. June 1983. Washington.

Kittler [1983]; Kittler J; Paler K.

An absorption edge detector.

Paper from proceedings of IEEE computer society conference on computer vision and pattern recognition. June 1983. Washington.

Kittler [1985]; Kittler J; Illingworth J.

Threshold selection based on simple image statistics

Computer Vision Graphics and image processing 1985 Vol 30.

Ledley [1964]; Ledley R. S.

High speed automatic analysis of biomedical pictures.

Science. Vol 146. 1964.

Ledley [1965]; Ledley R; et al

FIDAC film input to digital automatic computer and...

Optical and electro optical info Proc. Eds Tioppet S ; Beckowitz J. MIT Press Cambr. USA 1965.

Lee [1972]; Lee H; Fu K.S.

A Stochastic syntax analysis procedure and its application to pattern classification.

IEEE Transactions on Computers. Vol C21. No. 7. 1972

Levenshtein [1966]; Levenshtein A. V.

Binary codes capable of correcting deletions, insertions and reversals.

Soviet. Physics. Doklady. Vol 10. No. 8. Feb 1966.

Li [1981]; Li M; Grosky W. I; Jain R.

Normalized quadtrees with respect to translation

Proc. PRIP-81 Dallas Texas.

Lu [1978]; Lu S; Fu K.

Error correcting tree automata for syntactic pattern recognition.

IEEE Trans Computers. November 1978 Vol C-27 No 11.

Marr [1980]; Marr D; Hildreth E.

Theory of edge detection

Proc. Royal Society London. 1980 Vol B207.

Marr [1983]; Marr D; Hildreth E.

The detection of intensity changes by computer and biological vision systems.

Computer vision graphics and image processing 1983 (22)

Microsoft [1987]; Microsoft Corporation

QuickC Programmers Guide

QuickC Language Reference

QuickC Run-Time library Reference

Milgram [1979]; Milgram D. I.

Region extraction using convergent evidence.

Computer graphics image processing 1979. Vol 11 part 1 12 p23.

Murthy [1974]; Murthy W; Voupa K.

A search algorithm for skeletonisation of thick patterns.

Computer Graphics image processing 1974 Vol 3.

Naito [1978]; Naito S; Arakawea H.

Recognition of handwritten alphanumerics and symbols on centroid lines.

Paper from 4th international joint conference on pattern recognition 1978.

Naito [1983]; Naito S; Hagita N; Masuda I.

Handwritten Kanji recognition by feature matching methods and its applications to OCRs.

Paper from proceedings of IEEE computer society conference on computer vision and pattern recognition. June 1983. Washington.

Nevatia [1980]; Nevatia R; Babu R.

Linear feature extraction and description.

Computer Graphics Image Processing. 1980 Vol 13.

Ohlander [1978]; Ohlander R; Price K; Reddy D.

Picture segmentation using a recursive region splitting method.

Computer graphics image processing. 1978 Vol 8.

Palumbo [1986]; Palumbo P; Swaminathan P.
Document image binarization : Evaluation of algorithms.
Proc of SPIE Symposium on applications of digital image processing IX Washington. 1986.

Pavlidis [1973]; Pavlidis T; Horowitz S. L.
Piecewise approximation of plane curves.
Proceedings 1st international joint conference pattern recognition.

Pavlidis [1974]; Pavlidis T; Horowitz S.
Segmentation of plane curves.
IEEE Transaction on computers. Aug 74 Vol C 23.

Pavlidis [1975]; Pavlidis T; Ali F.
Computer recognition of handwritten numerals by polygonal approximations.
IEEE Trans. Syst. Man. Cybernetics. Nov 1975 Vol SMC 5.

Pavlidis [1977]; Pavlidis T.
Structural Pattern Recognition.
Springer-Verlag Berlin Heidelberg New York. 1977
ISBN 3-550-08463-0.

Pavlidis [1982]; Pavlidis T.
Algorithms for graphics and image processing
ISBN 0-914894-65-X.

Pavlidis [1983]; Pavlidis T.
Effects of distortion on the recognition rate of a structural OCR system.
Paper from proceedings of IEEE computer society conference on computer vision and pattern recognition. June 1983. Washington

Pfaltz [1969]; Pfaltz J; Rosenfeld A.

Web Grammars.

Paper from proceedings of the 1st international conference on A.I. Washington 1969.

Roberts [1965]; Roberts L. G.

Machine Perception of three dimensional solids.

Optical and opto electrical information processing, Tippett.J.P et al (Editors) Cambridge MA: MIT Press 1965

Rosenfeld [1971]; Rosenfeld A.

Connectivity in digital pictures.

Journal A.C.M. 1971 Vol 17 Part Jan.

Rosenfeld and Thurston [1971]; Rosenfeld A; Thurston M.

Edge and curve detection for visual scene analysis.

IEEE Trans Comput 1971 Vol C20.

Rosenfeld [1982]; Rosenfeld A; Kak A.C

Digital Picture Processing Vol 2.

Academic Press ,Inc. London. ISBN 0-12-597302-0

Rutovitz [1966]; Rutovitz D.

Pattern Recognition.

Jnl Royal Statistics Soc, Vol 129, Series A. 1966

Sidhu [1972]; Sidhu G; Boute R. T.

Property encoding : Application in binary picture encoding and boundary following.

IEEE Trans on Computers. Vol C21 Part 11 1972.

Smeed [1964]; Smeed Committee Report
Road Pricing: the economic and technical possibilities.
Ministry of Transport 1964. HMSO.

Srihari [1987]; Srihari S.N.
Recognising address blocks on mail pieces.
AI magazine Vol 8 no 4 Winter 1987.

Stefanelli [1971]; Stefanelli R; Rosenfeld A,
Some parallel thinning algorithms for digital pictures.
Jnl. Assoc. Computing machines 1971 Vol 18 no 2

Suen and Mori [1982]; Suen C. Y. ; Mori S.
Standardisation and automatic recognition of hand-printed characters.
Computer analysis and perception. Vol 1 Visual signals (CRC Press, Boca Raton, Florida 1982)

Suen [1982]; Suen C. Y.
Distinctive features in automatic recognition of handprinted characters.
Signal processing April 1982. Vol 4.

Suen [1986]; Suen C. Y.
Character recognition by computer and applications.
Paper from handbook of pattern recognition and image processing. ed. Young T.Y. 1986

Tamura and Mori [1978]; Tamura H; Mori S.
Textual features corresponding to visual perception
IEEE Trans on Systems Man and Cybernetics.

Tamura [1978] Tamura H.

A comparison of live thinning algorithms from a digital geometry viewpoint.

Proceedings of 4th international conference on pattern recognition.

Tanimoto [1975]; Tanimoto S; Pavlidis T.

A hierarchical data structure for picture processing.

Computer graphics and image processing. Part 4. 1975

Taylor [1988]; Taylor R.

Board-level decision (image processing survey)

Systems International November 1988.

Thomason [1975]; Thomason M; Gonzalez R.

Error detection and classification in syntactic pattern structure.

IEEE Trans Computers. Jan 1975 Vol C-24 Part 1.

Toussaint [1970]; Toussaint G; Donaldson R.

Algorithm for recognising contour traced handprinted characters.

IEEE Trans. Comput. Vol C-19 1970

Tretiak [1979]; Tretiak O.

A parametric model for edge detection

Proceedings 3rd COMPSAC. Nov 1979.

White [1983]; White J. M; Rohrer G. D

Image Thresholding for character image extraction and other applications requiring.

IBM Journal 1983 Vol 27 part 4.

Yamamoto [1978]; Yamamoto K; Mori S.

Recognition of handwritten characters by outermost point method.

Paper from proceedings of the 4th international joint conference on pattern recognition. Kyoto 1978.

Yokoi [1973]; Yokoi S; Toriwaki J.

Topological properties in digitised binary pictures.

Systems Computers Controls Vol 4, Part 6. 1973

Zucker [1976]; Zucker S. W.

Region growing : childhood and adolescence.

Computer graphics and image processing 1976 (5)

Zucker [1977]; Zucker S; Hummel R; Rosenfeld A.

An application of relaxation labelling to line and curve enhancement.

IEEE Trans Comput 1977 Col C26.

GLOSSARY

GLOSSARY

ANSI - American National Standards Institute.

ECMA - European Computer Manufacturers Association.

OCR - Optical Character Reader.

OCRA - Optical Character Reader (font A).

OCRB - Optical Character Reader (font B).

PTT - Postal, Telegraphic, Telecommunication.

SAT - Symmetric Axis Transform.

APPENDIX 1

SECOND ORDER DIFFERENCE METRIC

APPENDIX 1

SECOND ORDER DIFFERENCE METRIC

This appendix contains tables which show the values of the second order differences. The technique used is described in detail in chapter 3. In all cases the distance between the rows is ten pixels. The size of the pixel groups in which the differences are summed in these sample results is 50 and 100.

A.1.1 Image 1 - Summation length 50 Pixels

Row Number ↓	Column Number → 0	50	100	150	200	250	300	350	400	450
V 0	126	118	112	0	0	0	0	67	182	126
10	147	106	91	0	0	0	0	60	***248***	131
20	104	111	139	14	0	0	0	34	225	131
30	130	165	105	68	26	0	0	37	209	119
40	135	200	149	95	43	0	0	16	164	93
50	152	156	85	86	43	0	2	14	169	107
60	102	157	143	173	217	100	104	100	131	149
70	136	220	179	128	135	168	116	226	233	85
80	149	168	77	82	103	129	109	108	200	116
90	128	106	117	88	108	73	91	123	193	160
100	111	113	67	86	124	80	87	106	136	136
110	151	107	87	93	52	88	120	128	89	122
120	185	106	103	125	78	110	142	140	107	129
130	134	100	110	108	105	138	101	94	162	100
140	115	107	91	104	112	97	117	98	96	159
150	106	94	109	106	91	102	67	99	109	157
160	64	113	117	106	115	113	106	90	82	130
170	130	120	193	82	112	116	125	109	102	108
180	179	21	165	113	106	78	112	108	182	120
190	90	57	158	129	123	110	125	152	164	152
200	88	46	201	120	107	120	140	127	121	135
210	122	104	177	95	90	125	117	87	171	151
220	144	142	152	108	132	188	102	109	189	104
230	137	127	110	114	132	214	120	91	151	192
240	142	120	161	93	129	154	103	106	136	191
250	127	140	204	120	140	114	134	121	262	196
260	105	101	114	92	120	126	96	153	119	144
270	84	112	112	108	102	110	106	121	181	118
280	88	69	93	106	117	110	100	119	148	135
290	84	105	118	92	96	128	98	108	119	120
300	92	106	107	123	112	112	109	115	134	127
310	115	112	90	129	150	120	114	103	125	111
320	105	135	109	191	92	139	95	114	118	111
330	102	96	115	***280***	***311***	144	186	134	113	96
340	89	120	101	236	219	181	***291***	171	124	118
350	86	93	77	***273***	218	206	***293***	178	94	84
360	76	116	127	***248***	***255***	228	***303***	138	88	82
370	93	101	104	119	230	221	***285***	170	93	85
380	88	120	89	99	216	164	128	140	105	98
390	91	87	122	100	107	86	111	126	85	111
400	106	92	96	93	117	113	87	85	85	119
410	128	102	118	89	116	104	102	110	105	136
420	111	94	92	122	125	80	92	116	107	87
430	113	115	87	104	95	98	97	94	88	112
440	118	126	114	92	99	112	103	93	101	86
450	131	101	113	105	92	80	99	85	106	99
460	90	108	100	138	121	121	89	110	82	109
470	145	82	106	96	90	90	121	112	108	120
480	160	116	125	91	115	78	97	112	101	147
490	129	100	94	100	96	101	97	102	110	128
500	151	94	107	70	104	89	131	109	86	142
510	91	105	124	104	84	81	105	98	109	103

A.1.2 Image 1 – Summation length 100 Pixels

Row Number ↓	Column Number → 0	100	200	300	400
0	244	112	0	67	308
10	253	91	0	60	379
20	215	153	0	34	356
30	295	173	26	37	328
40	335	244	43	16	257
50	308	171	43	16	276
60	259	316	317	204	280
70	356	307	303	342	318
80	317	159	232	217	316
90	234	205	181	214	353
100	224	153	204	193	272
110	258	180	140	248	211
120	291	228	188	282	236
130	234	218	243	195	262
140	222	195	209	215	255
150	200	215	193	166	266
160	177	223	228	196	212
170	250	275	228	234	210
180	200	278	184	220	302
190	147	287	233	277	316
200	134	321	227	267	256
210	226	272	215	204	322
220	286	260	320	211	293
230	264	224	346	211	343
240	262	254	283	209	327
250	267	324	254	255	458
260	206	206	246	249	263
270	196	220	212	227	299
280	157	199	227	219	283
290	189	210	224	206	239
300	198	230	224	224	261
310	227	219	270	217	236
320	240	300	231	209	229
330	198	395	455	320	209
340	209	337	400	***462***	242
350	179	350	424	***471***	178
360	192	375	***483***	441	170
370	194	223	***451***	***455***	178
380	208	188	380	268	203
390	178	222	193	237	196
400	198	189	230	172	204
410	230	207	220	212	241
420	205	214	205	208	194
430	228	191	193	191	200
440	244	206	211	196	187
450	232	218	172	184	205
460	198	238	242	199	191
470	227	202	180	233	228
480	276	216	193	209	248
490	229	194	197	199	238
500	245	177	193	240	228
510	196	228	165	203	212

A.1.3 Image 2 – Summation length 50 Pixels

Row Number ↓	Column Number → 0	50	100	150	200	250	300	350	400	450
V 0	128	119	115	0	0	0	0	0	0	0
10	175	114	145	118	138	157	121	0	0	0
20	195	134	123	130	123	123	108	121	115	0
30	174	110	122	232	149	134	137	143	181	***259***
40	129	109	167	117	191	124	113	126	167	119
50	100	120	170	146	135	65	109	83	147	147
60	100	73	153	125	129	118	134	118	106	181
70	101	108	149	99	106	103	100	102	105	140
80	118	98	182	102	111	121	115	121	187	75
90	120	102	165	123	129	111	132	109	94	111
100	88	103	127	104	108	121	89	78	126	95
110	101	103	133	123	105	126	113	116	116	146
120	114	93	99	76	94	110	164	131	118	112
130	112	84	128	115	109	94	148	117	98	106
140	105	112	128	107	96	100	82	115	106	152
150	66	101	148	119	142	125	99	111	102	134
160	134	131	123	120	130	97	115	92	100	107
170	162	157	120	99	111	93	123	121	93	115
180	140	137	148	96	119	103	83	112	89	100
190	105	111	133	119	172	121	84	131	88	153
200	112	107	141	104	167	145	90	101	111	98
210	108	101	86	87	158	190	101	91	99	108
220	96	129	129	108	119	141	86	101	106	111
230	78	114	103	120	108	151	128	127	101	114
240	126	123	95	105	83	76	102	81	88	111
250	131	95	116	119	101	101	103	143	111	98
260	123	112	137	100	119	105	111	84	82	89
270	106	123	125	133	112	127	115	116	124	112
280	95	103	180	187	152	92	90	73	96	98
290	144	129	214	185	140	105	100	126	115	116
300	163	105	***238***	***252***	192	218	144	135	114	111
310	147	98	195	176	***246***	219	207	208	103	88
320	162	119	209	***272***	***254***	***262***	183	242	118	128
330	98	126	205	***236***	***243***	215	198	228	101	122
340	126	132	158	137	192	194	221	170	119	119
350	152	128	141	151	170	163	219	216	108	101
360	110	139	130	97	130	108	187	189	120	141
370	96	98	89	96	113	94	92	127	113	162
380	136	107	130	102	100	89	101	133	125	115
390	114	87	104	104	99	79	107	97	120	100
400	150	106	95	97	100	116	99	87	106	125
410	145	119	115	103	112	126	78	98	113	173
420	91	91	150	219	237	160	183	199	97	87
430	117	214	172	200	***248***	185	140	136	99	97
440	102	72	91	105	101	79	103	102	103	94
450	76	96	93	74	47	14	40	68	108	126
460	85	106	64	91	82	127	96	99	78	93
470	113	91	99	95	76	81	76	112	95	133
480	82	113	85	129	118	116	114	117	142	82
490	101	80	99	114	74	97	129	70	107	93
500	82	89	94	79	113	122	99	112	110	119
510	100	102	95	98	85	96	107	103	91	112

A.1.4 Image 2 – Summation length 100 Pixels

Row Number ↓	Column Number → 0	100	200	300	400
0	247	115	0	0	0
10	289	263	295	121	0
20	329	253	246	229	115
30	284	354	283	280	440
40	238	284	315	239	286
50	220	316	200	192	294
60	173	278	247	252	287
70	209	248	209	202	245
80	216	284	232	236	262
90	222	288	240	241	205
100	191	231	229	167	221
110	204	256	231	229	262
120	207	175	204	295	230
130	196	243	203	265	204
140	217	235	196	197	258
150	167	267	267	210	236
160	265	243	227	207	207
170	319	219	204	244	208
180	277	244	222	195	189
190	216	252	293	215	241
200	219	245	312	191	209
210	209	173	348	192	207
220	225	237	260	187	217
230	192	223	259	255	215
240	249	200	159	183	199
250	226	235	202	246	209
260	235	237	224	195	171
270	229	258	239	231	236
280	198	367	244	163	194
290	273	399	245	226	231
300	268	***490***	410	279	225
310	245	371	***465***	415	191
320	281	***481***	***516***	425	246
330	224	441	***458***	426	223
340	258	295	386	391	238
350	280	292	333	435	209
360	249	227	238	376	261
370	194	185	207	219	275
380	243	232	189	234	240
390	201	208	178	204	220
400	256	192	216	186	231
410	264	218	238	176	286
420	182	369	397	382	184
430	331	372	433	276	196
440	174	196	180	205	197
450	172	167	61	108	234
460	191	155	209	195	171
470	204	194	157	188	228
480	195	214	234	231	224
490	181	213	171	199	200
500	171	173	235	211	229
510	202	193	181	210	203

A.1.5 Image 3 - Summation length 50 Pixels

Row Number ↓	Column Number → 0	50	100	150	200	250	300	350	400	450
V 0	28	172	200	0	0	0	0	0	122	***240***
10	0	84	188	54	0	0	0	0	88	217
20	94	0	76	126	0	0	0	0	51	262
30	74	113	125	140	22	0	0	0	48	***272***
40	83	110	114	97	48	0	0	0	35	***282***
50	92	128	88	120	148	0	0	0	28	226
60	73	116	108	73	169	127	113	9	16	***313***
70	119	154	98	97	145	155	75	127	118	***250***
80	135	128	133	113	124	139	23	126	143	199
90	92	146	108	110	100	125	87	85	131	223
100	127	104	138	159	89	117	139	105	114	164
110	170	193	152	161	212	121	116	134	145	209
120	109	333	164	137	270	111	110	160	113	117
130	134	130	108	151	193	119	144	166	101	111
140	144	104	114	101	243	128	93	94	112	115
150	125	96	100	115	118	147	119	215	130	101
160	133	108	138	117	89	155	133	159	152	128
170	124	132	109	113	83	122	121	87	128	98
180	116	104	126	117	96	82	125	127	153	140
190	107	99	86	116	116	104	94	188	119	115
200	68	111	108	122	108	129	125	221	140	129
210	138	139	126	141	96	113	96	102	114	140
220	81	131	133	117	123	85	80	80	156	144
230	102	118	108	149	99	82	85	123	140	138
240	84	127	100	89	125	106	138	113	91	151
250	80	94	109	96	104	115	130	117	122	143
260	91	128	75	86	124	104	112	104	119	166
270	85	90	127	85	92	110	108	98	119	98
280	86	95	132	116	110	101	95	93	123	93
290	108	111	127	111	159	94	108	78	138	111
300	85	119	187	184	186	128	149	100	127	89
310	81	117	167	193	***265***	***293***	154	106	118	97
320	66	126	206	161	174	***317***	136	86	88	89
330	70	93	162	180	169	***249***	146	101	87	142
340	72	116	146	96	145	***269***	142	95	106	166
350	79	138	87	91	124	107	138	89	75	153
360	113	258	98	97	98	96	128	85	105	179
370	83	174	112	91	116	108	104	106	115	131
380	107	108	106	79	104	97	139	144	166	117
390	110	190	106	87	92	105	155	168	125	139
400	107	97	110	78	92	94	150	186	152	123
410	85	89	88	101	106	118	140	205	137	116
420	90	111	97	91	87	105	106	166	160	136
430	101	98	99	90	88	91	114	137	139	159
440	78	112	85	91	112	103	121	152	173	137
450	112	109	73	91	86	100	99	114	160	185
460	78	82	90	112	85	96	88	124	179	164
470	89	85	96	112	117	94	97	87	165	188
480	60	84	74	93	89	112	104	86	235	189
490	89	99	95	126	97	83	96	94	185	176
500	87	144	75	86	112	100	96	98	181	211
510	95	113	106	122	110	105	91	111	161	158

A.1.6 Image 3 – Summation length 100 Pixels

Row Number ↓	Column Number → 0	100	200	300	400
0	200	200	0	0	362
10	84	242	0	0	305
20	94	202	0	0	313
30	187	265	22	0	320
40	193	211	48	0	317
50	220	208	148	0	254
60	189	181	296	122	339
70	273	195	300	202	368
80	263	246	263	149	342
90	238	218	225	172	354
100	231	297	206	244	278
110	363	313	333	250	354
120	442	301	381	270	230
130	264	259	312	310	212
140	248	215	371	187	227
150	221	215	265	334	231
160	241	255	244	292	280
170	256	222	205	208	226
180	220	243	178	252	293
190	206	202	220	282	234
200	179	230	237	346	269
210	277	267	209	198	254
220	212	250	208	160	300
230	220	257	181	208	278
240	211	189	231	251	242
250	174	205	219	247	265
260	219	161	228	216	285
270	175	212	202	206	217
280	181	248	211	188	216
290	219	238	253	186	249
300	204	***371***	314	249	216
310	198	360	***558***	260	215
320	192	367	***491***	222	177
330	163	342	***418***	247	229
340	188	242	***414***	237	272
350	217	178	231	227	228
360	371	195	194	213	284
370	257	203	224	210	246
380	215	185	201	283	283
390	300	193	197	323	264
400	204	188	186	336	275
410	174	189	224	345	253
420	201	188	192	272	296
430	199	189	179	251	298
440	190	176	215	273	310
450	221	164	186	213	345
460	160	202	181	212	343
470	174	208	211	184	353
480	144	167	201	190	424
490	188	221	180	190	361
500	231	161	212	194	392
510	208	228	215	202	319

A.1.7 Image 4 – Summation length 50 Pixels

Row Number ↓	Column Number → 0	50	100	150	200	250	300	350	400	450
V 0	86	113	109	97	122	138	106	94	93	117
10	101	105	98	50	0	0	0	79	110	117
20	90	93	97	94	38	72	23	88	126	111
30	82	128	86	95	81	211	95	84	85	119
40	65	93	73	82	95	181	107	99	77	109
50	102	115	101	88	130	159	83	69	86	105
60	80	90	115	84	118	99	95	106	95	86
70	97	94	104	67	98	101	92	115	88	115
80	88	119	96	110	103	85	91	94	102	108
90	100	115	95	106	109	81	75	92	108	100
100	89	82	96	113	92	88	71	82	80	86
110	99	93	111	81	106	97	98	112	103	102
120	65	119	90	111	111	94	96	98	113	92
130	88	110	102	107	121	88	90	127	106	109
140	104	123	139	98	88	93	101	104	100	146
150	117	123	194	157	165	147	162	175	98	113
160	109	116	186	147	143	123	161	178	184	96
170	86	121	137	206	168	135	175	176	159	127
180	103	107	91	119	114	110	109	106	119	125
190	92	120	96	110	102	115	91	115	89	131
200	96	158	114	115	97	103	79	119	102	190
210	102	122	135	128	100	106	102	119	110	142
220	82	128	111	***213***	130	111	172	171	159	112
230	71	142	139	171	186	107	***207***	186	131	178
240	106	185	133	***192***	148	150	***205***	182	185	113
250	95	***220***	166	***229***	155	120	***203***	***190***	***187***	145
260	118	***200***	150	168	151	101	186	140	182	110
270	92	140	165	160	150	172	184	169	146	135
280	108	90	179	176	130	150	265	175	167	135
290	74	184	141	82	148	192	195	136	164	104
300	101	127	133	122	106	132	98	99	118	118
310	95	121	117	125	119	126	80	108	159	147
320	94	101	105	115	82	150	81	131	99	130
330	104	120	106	115	98	126	109	124	96	136
340	103	125	102	99	105	124	141	136	96	123
350	104	105	106	101	146	113	113	92	127	115
360	139	110	107	89	135	124	148	128	98	119
370	125	125	109	94	111	103	134	100	109	135
380	97	101	116	106	107	127	102	123	142	92
390	106	104	92	100	104	108	78	105	114	115
400	82	111	81	94	121	111	82	73	89	94
410	58	111	123	97	100	116	122	83	88	119
420	107	110	113	100	93	122	104	94	131	120
430	68	91	96	114	101	92	100	99	110	82
440	67	108	104	117	100	116	108	99	112	85
450	90	120	100	114	122	109	107	119	98	107
460	80	95	93	104	125	116	103	91	75	106
470	60	87	126	99	128	105	103	86	98	96
480	77	99	114	75	102	123	140	83	103	107
490	90	103	96	90	80	87	108	96	88	103
500	83	104	86	81	67	91	72	88	106	84
510	87	102	85	94	75	66	74	96	88	97

A.1.8 Image 4 – Summation length 100 Pixels

Row Number ↓	Column Number → 0	100	200	300	400
0	199	206	260	200	210
10	206	148	0	79	227
20	183	191	110	111	237
30	210	181	292	179	204
40	158	155	276	206	186
50	217	189	289	152	191
60	170	199	217	201	181
70	191	171	199	207	203
80	207	206	188	185	210
90	215	201	190	167	208
100	171	209	180	153	166
110	192	192	203	210	205
120	184	201	205	194	205
130	198	209	209	217	215
140	227	237	181	205	246
150	240	***351***	312	337	211
160	225	333	266	339	280
170	207	343	303	351	286
180	210	210	224	215	244
190	212	206	217	206	220
200	254	229	200	198	292
210	224	263	206	221	252
220	210	324	241	343	271
230	213	310	293	393	309
240	291	325	298	387	298
250	315	***395***	275	393	332
260	318	318	252	326	292
270	232	325	322	***353***	281
280	198	***355***	280	***440***	302
290	258	223	340	331	268
300	228	255	238	197	236
310	216	242	245	188	306
320	195	220	232	212	229
330	224	221	224	233	232
340	228	201	229	277	219
350	209	207	259	205	242
360	249	196	259	276	217
370	250	203	214	234	244
380	198	222	234	225	234
390	210	192	212	183	229
400	193	175	232	155	183
410	169	220	216	205	207
420	217	213	215	198	251
430	159	210	193	199	192
440	175	221	216	207	197
450	210	214	231	226	205
460	175	197	241	194	181
470	147	225	233	189	194
480	176	189	225	223	210
490	193	186	167	204	191
500	187	167	158	160	190
510	189	179	141	170	185

A.1.9 Image 5 – Summation length 50 Pixels

Row Number ↓	Column Number → 0	50	100	150	200	250	300	350	400	450
V 0	82	114	126	85	94	109	136	157	138	125
10	117	119	101	93	57	136	112	121	136	118
20	112	126	122	83	113	96	100	119	143	103
30	121	133	104	106	131	116	114	161	156	104
40	186	125	101	121	107	107	107	114	127	102
50	174	147	155	104	108	86	95	193	125	146
60	87	149	193	193	211	89	88	116	125	113
70	118	146	121	191	102	136	98	117	151	143
80	89	103	116	117	102	198	75	119	144	161
90	122	116	140	117	102	112	125	134	139	137
100	99	129	126	130	133	129	156	173	149	163
110	97	103	126	121	85	151	155	115	111	100
120	95	100	134	103	114	148	120	103	88	118
130	89	107	113	99	130	222	124	103	95	127
140	86	106	121	102	100	173	125	113	85	93
150	85	139	126	98	75	138	135	103	117	104
160	112	122	105	110	119	133	105	114	86	135
170	102	134	100	99	106	105	113	110	138	126
180	102	98	118	111	120	130	94	99	100	133
190	107	92	138	103	106	103	93	105	115	112
200	73	129	152	86	123	98	156	134	108	103
210	104	110	120	103	108	114	161	142	125	138
220	75	111	99	102	97	124	179	131	109	94
230	127	119	135	78	91	100	109	113	110	98
240	116	86	137	100	122	84	112	99	128	107
250	132	105	143	99	86	114	111	112	96	108
260	96	98	96	117	117	107	98	82	95	76
270	114	126	111	128	97	103	90	112	98	115
280	125	89	97	146	117	111	96	86	86	146
290	122	108	98	162	173	169	196	***206***	164	159
300	129	94	91	164	***210***	***237***	165	***216***	***218***	197
310	97	104	106	175	184	171	162	174	198	151
320	108	102	109	***192***	191	137	171	162	***269***	177
330	106	103	75	154	177	169	121	***205***	183	181
340	127	108	112	167	146	190	***205***	***212***	***198***	167
350	108	94	87	157	139	160	177	155	211	149
360	101	112	105	120	71	116	112	112	115	107
370	125	100	99	118	119	131	109	112	86	111
380	95	118	137	83	107	120	108	108	106	133
390	87	115	135	90	115	113	99	103	96	113
400	97	106	84	89	88	101	90	91	89	98
410	88	101	118	89	126	124	87	107	89	111
420	113	93	85	69	95	90	124	120	95	100
430	79	84	100	96	116	139	97	84	86	81
440	103	95	110	89	109	97	122	89	114	83
450	86	116	97	89	101	98	131	106	93	85
460	77	89	94	96	105	93	116	108	104	97
470	76	99	104	112	92	130	83	65	103	89
480	85	100	93	112	97	106	101	88	108	116
490	95	135	87	97	102	110	111	95	115	105
500	102	84	112	125	116	90	111	102	103	111
510	84	90	106	93	96	85	91	99	120	91

A.1.10 Image 5 – Summation length 100 Pixels

Row Number ↓	Column Number → 0	100	200	300	400
0	196	211	203	293	263
10	236	194	193	233	254
20	238	205	209	219	246
30	254	210	247	275	260
40	311	222	214	221	229
50	321	259	194	288	271
60	236	386	300	204	238
70	264	312	238	215	294
80	192	233	300	194	305
90	238	257	214	259	276
100	228	256	262	329	312
110	200	247	236	270	211
120	195	237	262	223	206
130	196	212	352	227	222
140	192	223	273	238	178
150	224	224	213	238	221
160	234	215	252	219	221
170	236	199	211	223	264
180	200	229	250	193	233
190	199	241	209	198	227
200	202	238	221	290	211
210	214	223	222	303	263
220	186	201	221	310	203
230	246	213	191	222	208
240	202	237	206	211	235
250	237	242	200	223	204
260	194	213	224	180	171
270	240	239	200	202	213
280	214	243	228	182	232
290	230	260	342	***402***	323
300	223	255	***447***	***381***	***415***
310	201	281	355	336	349
320	210	301	328	333	***446***
330	209	229	346	326	364
340	235	279	336	417	365
350	202	244	299	332	360
360	213	225	187	224	222
370	225	217	250	221	197
380	213	220	227	216	239
390	202	225	228	202	209
400	203	173	189	181	187
410	189	207	250	194	200
420	206	154	185	244	195
430	163	196	255	181	167
440	198	199	206	211	197
450	202	186	199	237	178
460	166	190	198	224	201
470	175	216	222	148	192
480	185	205	203	189	224
490	230	184	212	206	220
500	186	237	206	213	214
510	174	199	181	190	211

APPENDIX 2

SOFTWARE STRUCTURE CHARTS

APPENDIX 2

SOFTWARE STRUCTURE CHARTS

A.2.1 Program SECOND.C

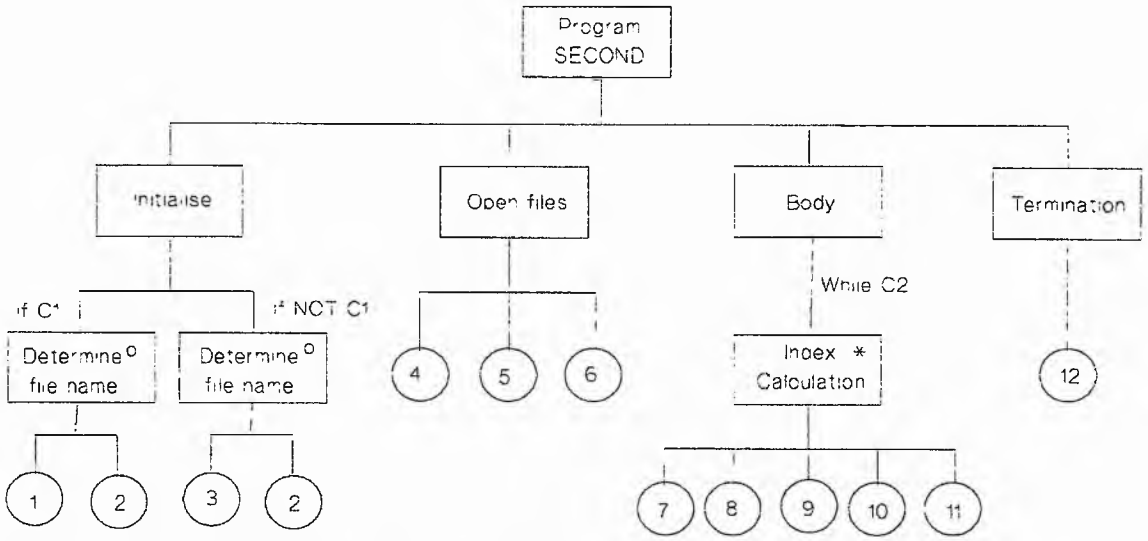


Figure A2.1 Structure chart of SECOND.C

A.2.1.1 Conditions

C1. In-line parameter found.
C2. Not end-of-file.

A.2.1.2 Actions

1. Get in-line image file name.
2. Open image file for reading.
3. Get prompted image file name.
4. Open FILE.NAM for writing.
5. Write image file name to FILE.NAM.
6. Open SECOND.OUT for writing.
7. Read one line from image file.
8. Calculate second difference.
9. Sum groups of adjacent second differences.
10. Output result to SECOND.OUT.
11. Increment image file pointer by GAP_BETWEEN_LINES lines.
12. Close all files.

A.2.2 Program BINARY.C

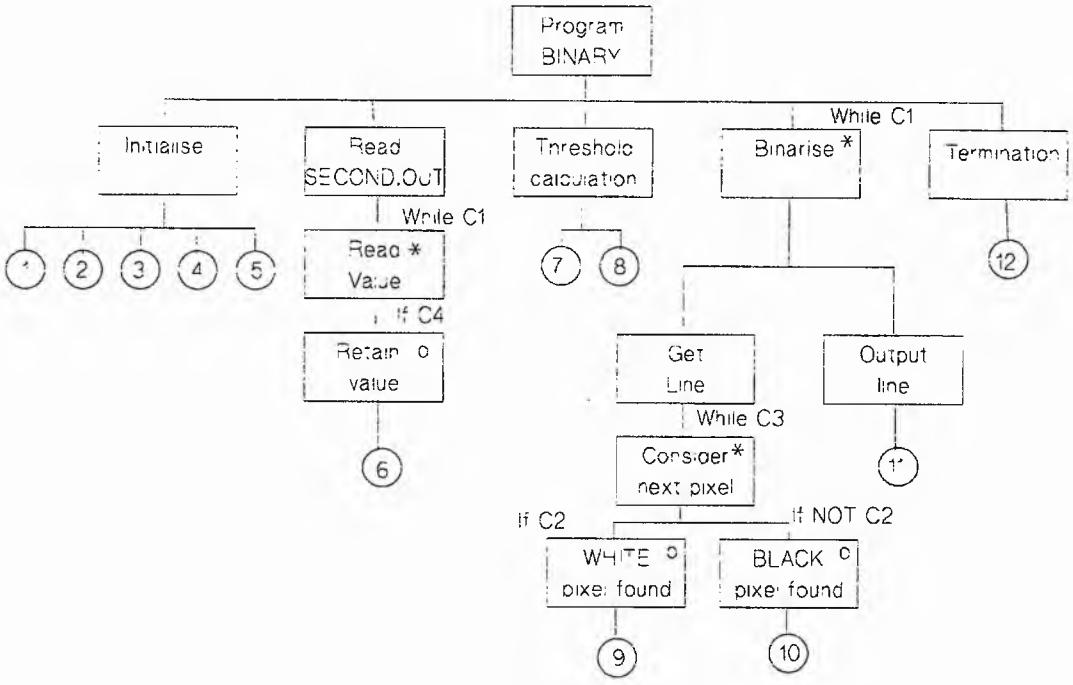


Figure A2.2 Structure chart of BINARY.C

A.2.2.1 Conditions

C1. Not end-of-file.
C2. If pixel > threshold.
C3. Not end of line.
C4. If value > maximum value so far read.

A.2.2.2 Actions

1. Open FILE.NAM for reading.
2. Read name of image file from FILE.NAM.
3. Open image file for reading.
4. Open SECOND.OUT for reading.
5. Open BINARY.IMG for writing.
6. Calculate OFFSET into image file.
7. Read GROUP_SIZE bytes from image file at OFFSET.
8. Threshold = (maximum + minimum)/2
9. Convert pixel in line to WHITE.
10. Convert pixel in line to BLACK.
11. Output converted line to BINARY.IMG.
12. Close files.

A.2.3 Program RGROW.C

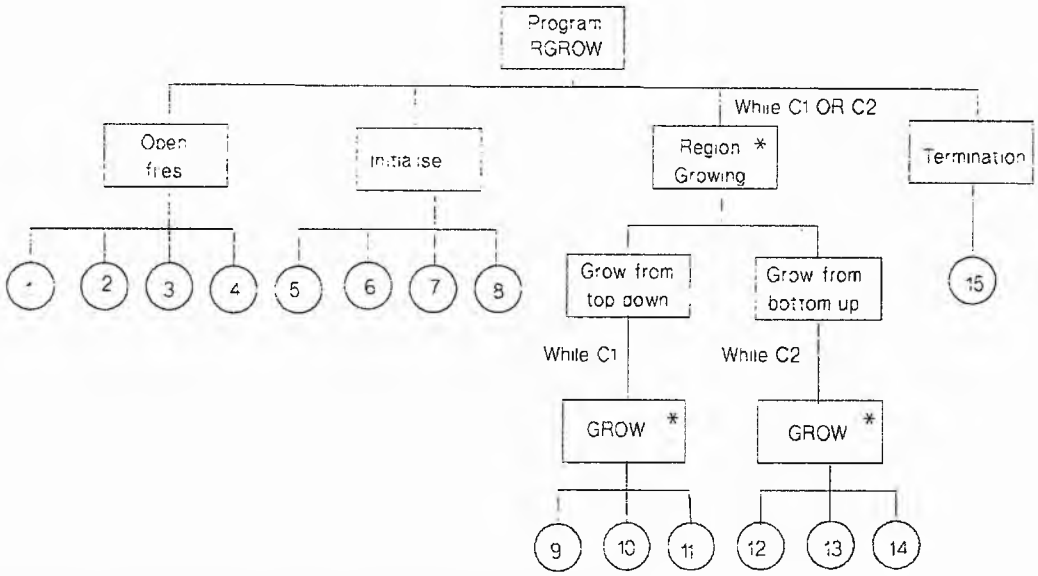


Figure A2.3 Structure chart of RGROW.C

A.2.3.1 Conditions

C1. Pixel intensities change on growing down.
C2. Pixel intensities change on growing up.

A.2.3.2 Actions

1. Open BINARY.IMG for reading.
2. Open RGROW.IMG for writing.
3. Open SECOND.OUT for reading.
4. Open FILE.NAM for reading.
5. Get maximum second difference from SECOND.OUT.
6. Calculate offset into image file for region growing.
7. Read GROUP_SIZE pixels in the image file at the calculated offset and convert WHITE pixels to GREY.
8. Top of region growing = current line in image file.
9. Convert WHITE pixels touching GREY above to GREY.
10. Bottom of region growing = current line.
11. Read line below from image file.
12. Convert WHITE pixels touching GREY below to GREY.
13. Top of region growing = current line.
14. Read line above from image file.
15. Close all files.

A.2.4 Program RECT.C

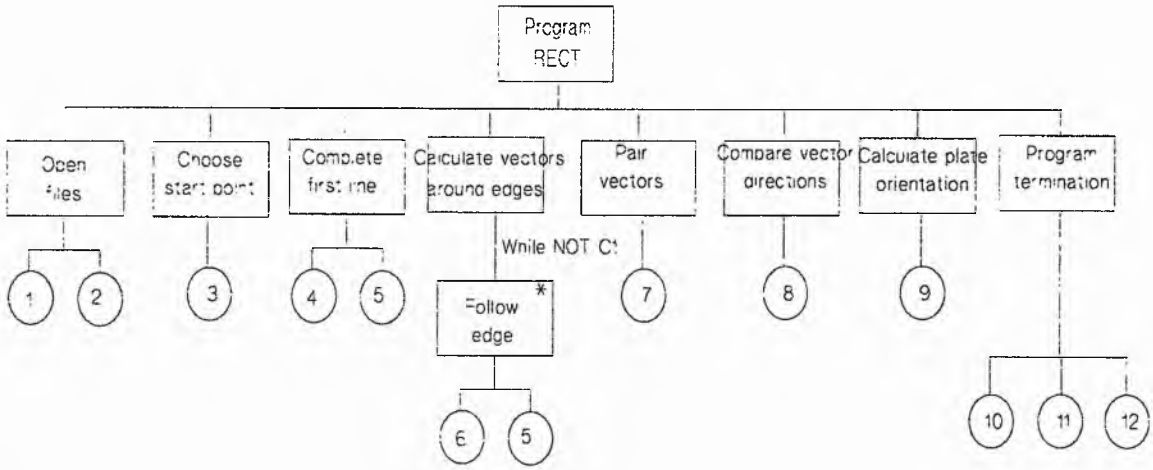


Figure A2.4 Structure chart of RECT.C

A.2.4.1 Conditions

C1. Perimeter of plate traversed.

A.2.4.2 Actions

1. Open RGROW.IMG for reading.
2. Open RECT.TXT for writing.
3. Follow edge from start point until abrupt change in direction.
4. Follow edge from start point in the untraversed direction.
5. Calculate vector for line.
6. Follow edge until abrupt change.
7. Pair vectors which are in the same direction.
8. Determine if vector pairs are at right angles.
9. Calculate orientation of long horizontal size of plate = direction of greatest vector.
10. Output orientation to RECT.TXT.
11. Display statement of results.
12. Close files.

A.2.5 Program EXT.C

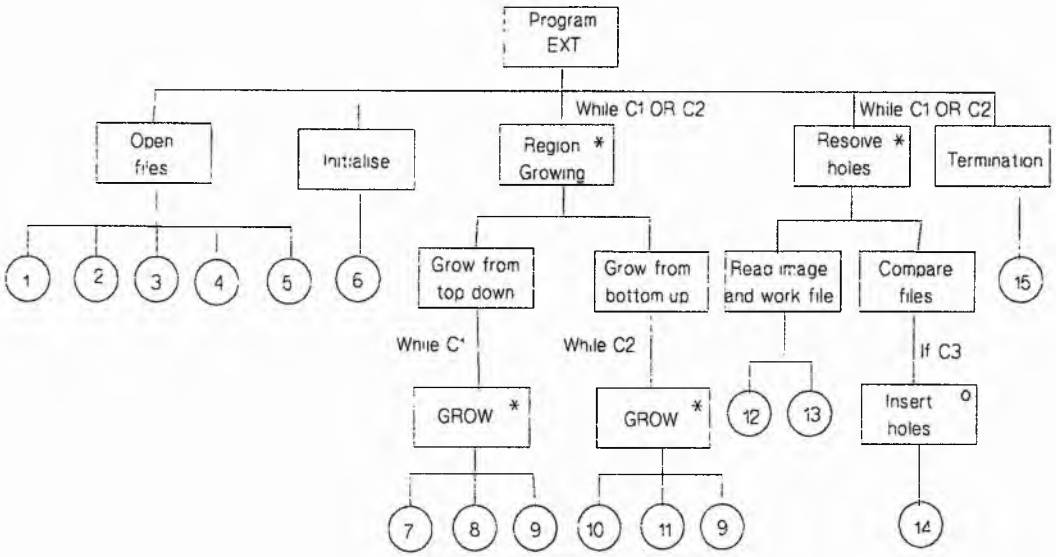


Figure A2.5 Structure chart of EXT.C

A.2.5.1 Conditions

C1. Pixel intensities change on growing down.
C2. Pixel intensities change on growing up.
C3. WHITE in source and BLACK in work file.

A.2.5.2 Actions

1. Open RGROW.IMG for reading.
2. Open EXT.IMG for writing (work file).
3. Open SECOND.OUT for reading.
4. Open FILE.NAM for reading.
5. Open BINARY.IMG for reading (source file).
6. Set pixels in top left corner to GREY. (Top line = current line).
7. Read line below from image file.
8. New current line = line below current line.
9. Convert WHITE pixels touching GREY to GREY.
10. New current line = line above current line.
11. Read line below from image file.
12. Read line from image file.
13. Read line from work file.
14. Convert BLACK pixels in work file to WHITE.
15. Close all files.

A.2.6 Program EXTRACT.C

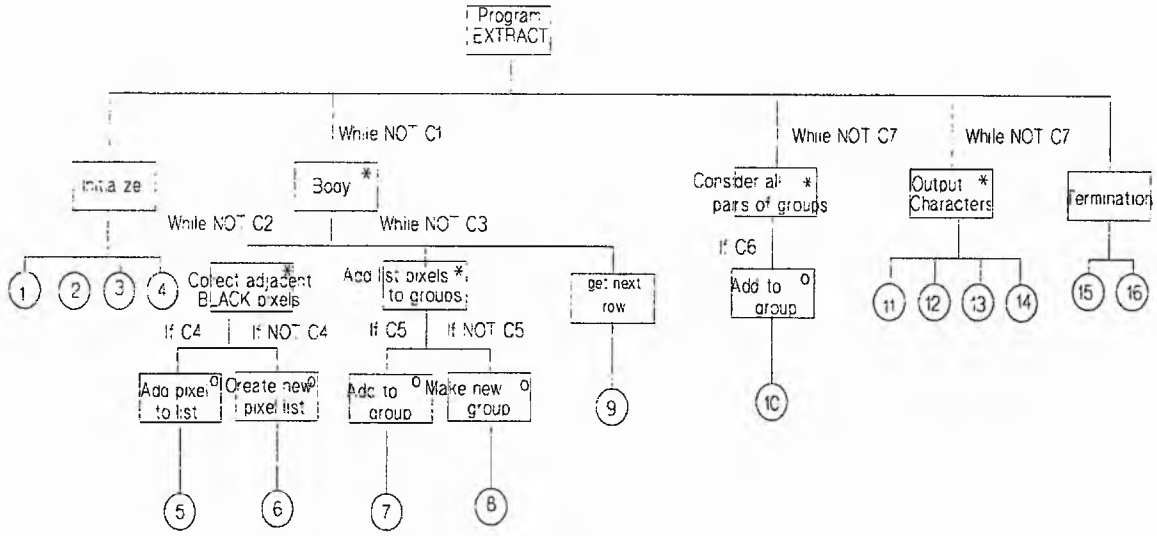


Figure A2.6 Structure chart of EXTRACT.C

A.2.6.1 Conditions

C1. Last row.
C2. Last pixel in this row.
C3. Last pixel list for this row.
C4. Last pixel in row.
C5. Pixel list touches existing pixel list.
C6. Pixel groups touch.
C7. Last pair of pixel group.

A.2.6.2 Actions

1. Open EXT.IMG.
2. n = 1.
3. Open FILE.NO for writing.
4. Current row = top row of pixels in EXT.IMG.
5. Add pixel to existing list of BLACK pixels for this row.
6. Create new BLACK pixel list for this row.
7. Add pixel list to an existing group of pixels.
8. Create new pixel group.
9. Read new pixel row from EXT.IMG.
10. Combine the pair of pixel groups under consideration.
11. Open .SOn file for writing.
12. Write pixel group to .SOn file as 100×100 bit map.
13. Close .SOn.
14. Increment n.
15. Write number of characters (= n-1) to FILE.NO.
16. Close all files.

A.2.7 Program THIN.C

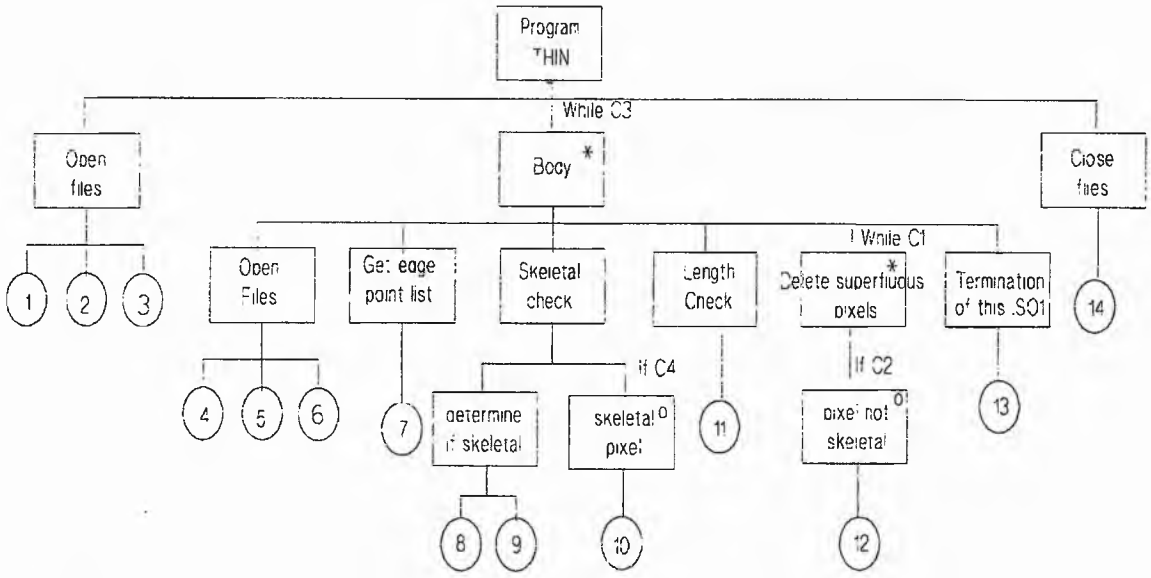


Figure A2.7 Structure chart of THIN.C

A.2.7.1 Conditions

C1. All skeletal pixels have not yet been considered.

C2. If pixel and neighbours match one of templates, remove superfluous pixel.

C3. .SOn files processed so far < value read from FILE.NO.

C4. Pixel is skeletal (distance to two opposite edges is the same).

A.2.7.2 Actions

1. Open FILE.NO for reading.
2. Read number of files to be processed from FILE.NO.
3. n = 1.
4. Open .SOn file for reading.
5. Open .SKn file for writing.
6. Increment n.
7. Generate edge point list.
8. Calculate distance to nearest edge.
9. Generate vector in opposite direction.
10. Ensure pixel is skeletal in .SKn file.
11. Delete skeletal chains 2 or less pixels long.
12. Ensure pixel is not skeletal in .SKn file.
13. Close .SOn and .SKn files.
14. Close files.

A.2.8 Program EXTEND1.C

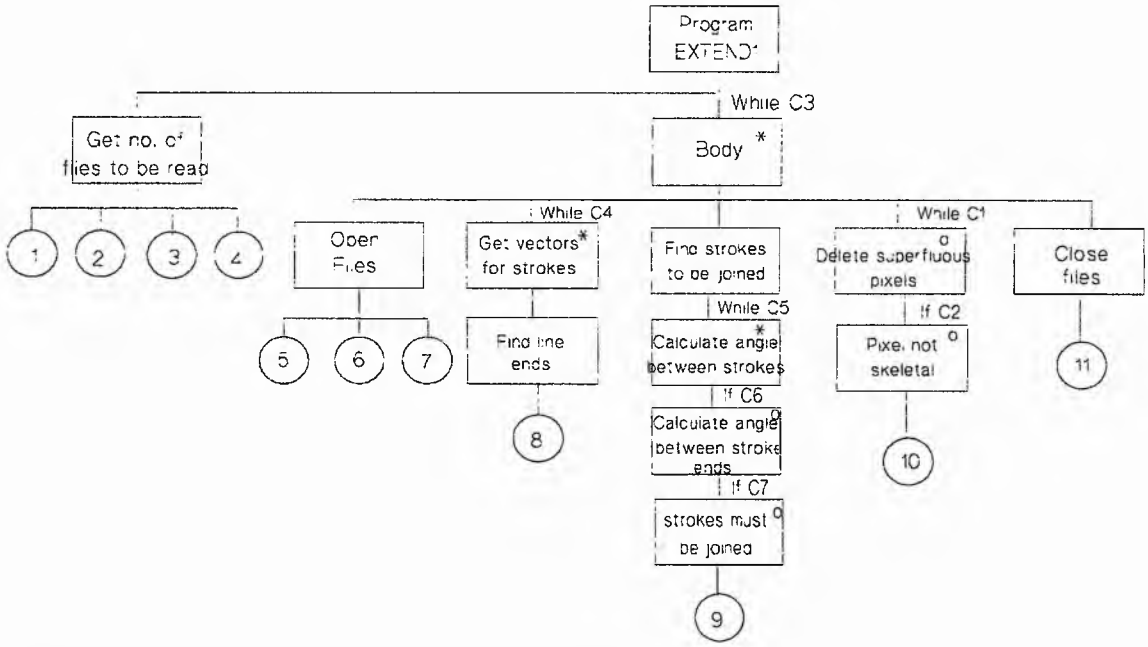


Figure A2.8 Structure chart of EXTEND1.C

A.2.8.1 Conditions

C1. All skeletal pixels have not yet been considered.
C2. If pixel and neighbours match one of templates.
C3. .SOn files processed so far < value read from FILE.NO.
C4. All strokes have not yet been considered.
C5. All pairs of strokes have not yet been considered.
C6. If strokes are at 180±15 degrees to each other.
C7. If vector joining stroke ends is parallel to strokes ±20 degrees.

A.2.8.2 Actions

1. Open FILE.NO for reading.
2. Read number of files to be processed from FILE.NO.
3. n = 1.
4. Close FILE.NO.
5. Open .SKn file for reading.
6. Open .SLn file for writing.
7. Increment n.
8. Determine direction of both ends of stroke.
9. Connect ends of strokes.
10. Ensure pixel is not skeletal in .SLn file.
11. Close files.

A.2.9 Program EXTEND2.C

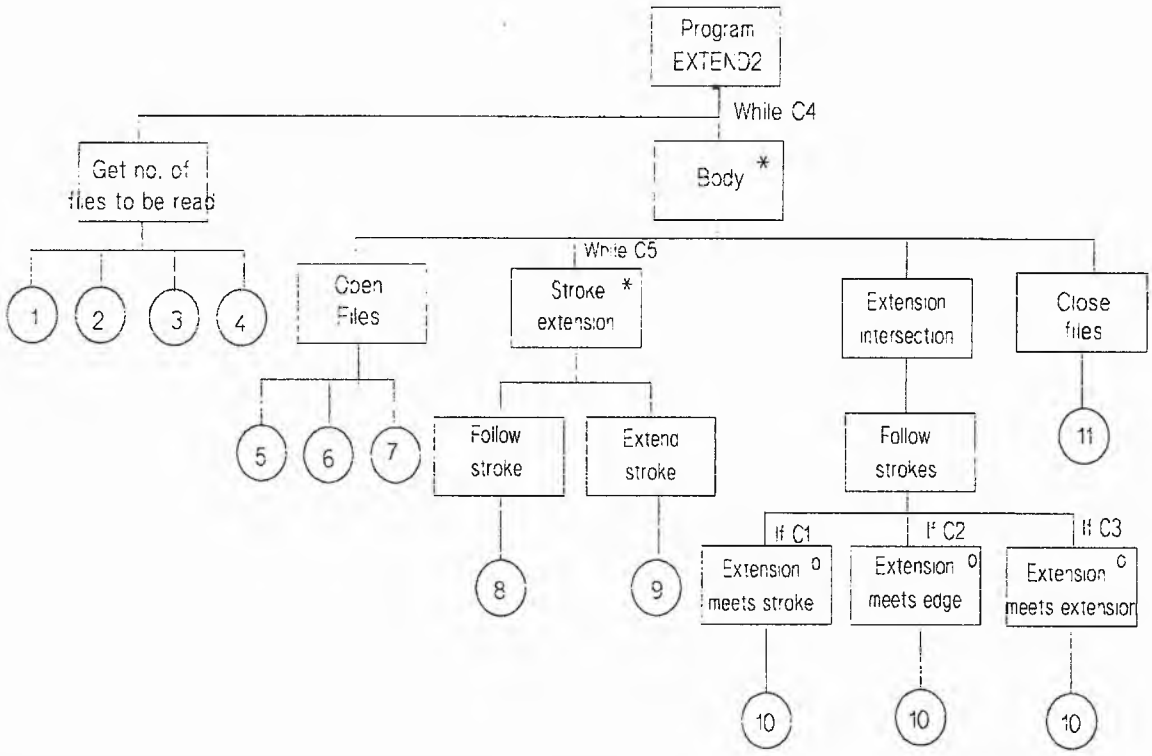


Figure A2.9 Structure chart of EXTEND2.C

A.2.9.1 Conditions

C1. Extended stroke meets a stroke.
C2. Extended stroke meets an edge.
C3. Extended stroke meets another extension.
C4. files processed so far < value read from FILE.NO.
C5. All strokes have not yet been considered.

A.2.9.2 Actions

1. Open FILE.NO for reading.
2. Read number of files to be processed from FILE.NO.
3. n = 1.
4. Close FILE.NO.
5. Open .SLn file for reading.
6. Open .SMn file for writing.
7. Increment n.
8. Determine direction of both ends of stroke.
9. Extend stroke from both ends.
10. Delete remainder of stroke.
11. Close files.

A.2.10 Program SYNTAC.C

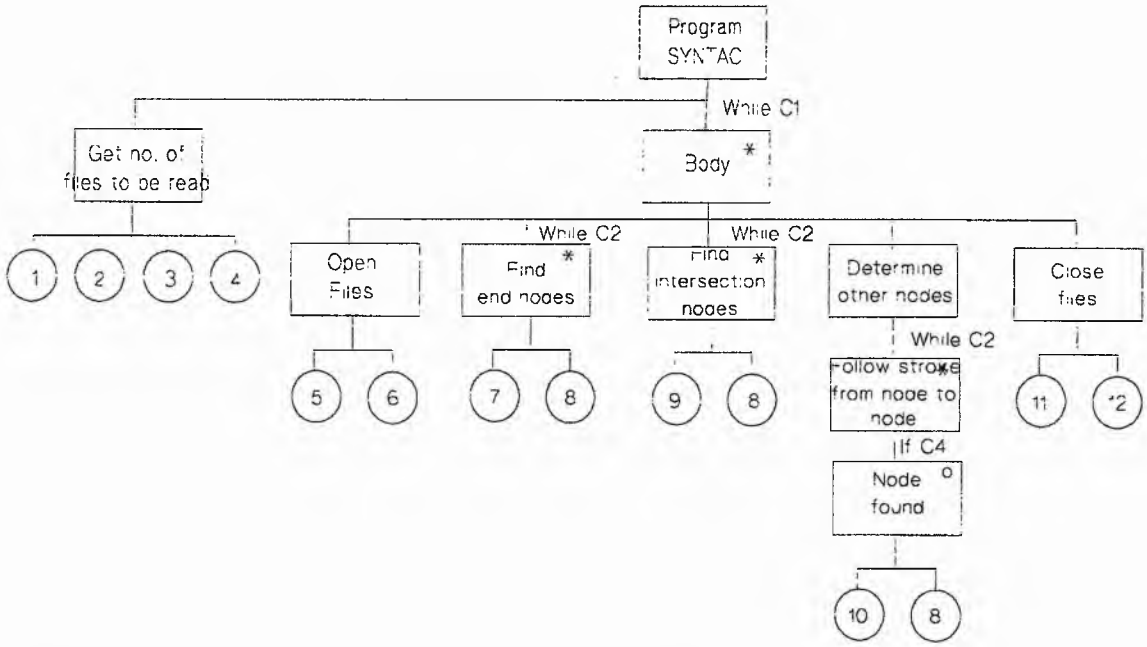


Figure A2.10 Structure chart of SYNTAC.C

A.2.10.1 Conditions

C1. files processed so far < value read from FILE.NO.
C2. All skeletal pixels have not yet been considered.
C3. All strokes have not yet been considered.
C4 Type 'c' node found.

A.2.10.2 Actions

1. Open FILE.NO for reading.
2. Read number of files to be processed from FILE.NO.
3. n = 1.
4. Close FILE.NO.
5. Open .SMn file for reading. Open .SYn file for writing
6. Increment n.
7. Type 'a' node.
8. Log node type and connectivity.
9. Type 'b' node.
10. Type 'c' node.
11. Output node details to .SYn file.
12. Close files.

A.2.11 Program STRING.C

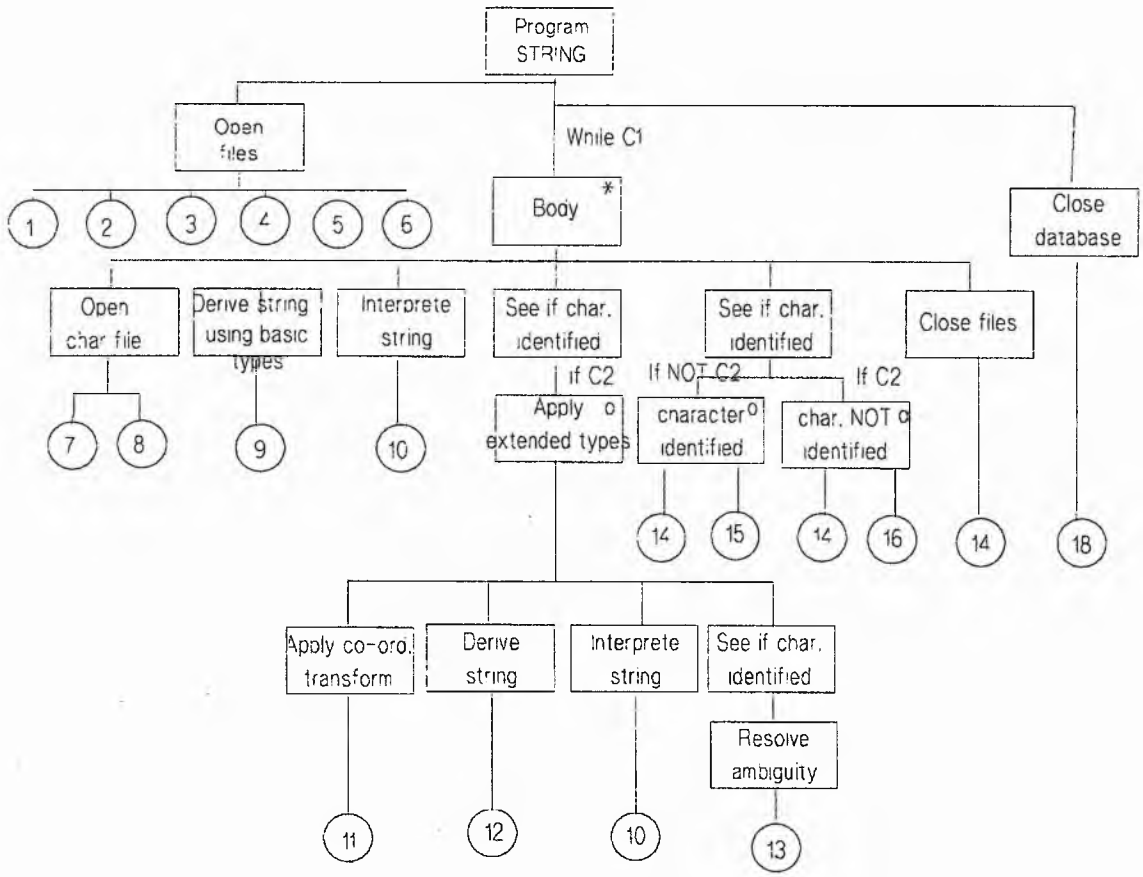


Figure A2.11 Structure chart of STRING.C

A.2.11.1 Conditions

C1. Characters processed so far<value read from FILE.NO.
C2. String representation ambiguous.

A.2.11.2 Actions

1. Open STRING.TXT for reading (database file).
2. Open FILE.NO for reading.
3. Read number of files to be processed from FILE.NO.
4. n = 1.
5. Close FILE.NO.
6. Open RECT.TXT for reading.
7. Open character connectivity (.SYn) file for reading.
8. Increment n.
9. Produce string representations using simple types.
10. Compare derived strings to STRING.TXT database.

11. Convert node co-ordinates to co-ordinate frame derived from RECT.TXT if character >45 degrees from image co-ordinate frame.
12. Produce string representations using extended types.
13. Attempt to identify character using context information.
14. Display string.
15. Display character.
16. Display failure to recognize message.
17. Close character connectivity file.
18. Close files.