

## Understanding paper documents

ARUN AGARWAL

Artificial Intelligence Laboratory, School of Mathematics and Computer/  
Information Sciences, University of Hyderabad, Hyderabad 500 134, India

**Abstract.** We describe the organization and several components of an automated document processing system that begins with digitized images of documents and produces representations at higher levels. Such representations include: the visual sketch (connected components extracted from the background), physical layout (spatial extents of blocks corresponding to text, graphics), logical layout (grouping of strings into words and phrases), and block primitives (e.g., recognised characters and words in text blocks, recognition of hand-drawn line drawings i.e. schematic electronic circuits). We describe algorithms for deriving several of the representations and describe the interaction of different modules. The methods are illustrated with examples.

**Keywords.** Connected component analysis; segmentation; thinning; text and graphic recognition.

### 1. Introduction

Paper documents continue to be the most common medium of information transmission in society. Understanding messages conveyed by printed documents is a common intelligent activity of humans. Making a computer perform the task of analysing and understanding a document is a challenging one, and requires the development and integration of techniques in several subareas of vision, artificial intelligence, knowledge representation, language understanding and control.

Computer document processing is a highly useful technique. The use of computers can increase ease of document processing and storing in electronic databases. It is also a convenient and efficient medium for storage, document update and revision. Examples where frequent document revisions are needed are technical manuals and parts catalogs. In all these examples, if the information is first captured digitally and stored as an archival database, the problem of document processing becomes simpler. Text and graphic editors can then be used to update such digitized documents in the new environment. Note that it is necessary to generate a description of graphical elements in the document (rather than a bit-map) in order to decrease the storage and processing time. Document processing systems are useful not only for document revision but also for technical design. A database of design building blocks can be built from the existing paper-based material, and can be used in the improvement of existing products or design of new ones with the use of computer aided design (CAD)

facilities. Automated document analysis systems should have intelligence incorporated to specifically identify graphic components in a document.

Two issues for interpreting digitized documents are:

- 1) the amount of data and the amount of storage space for a raster-oriented digital picture is very large. It is easily seen that the compression of data is possible by interpreting the image at a higher level. This description would, for example, provide American Standard Code for Information Interchange (ASCII) for text and describe graphics as lines or even higher levels of abstraction;
- 2) to make the document compatible with CAD or other document processing tools in order to provide faster throughput.

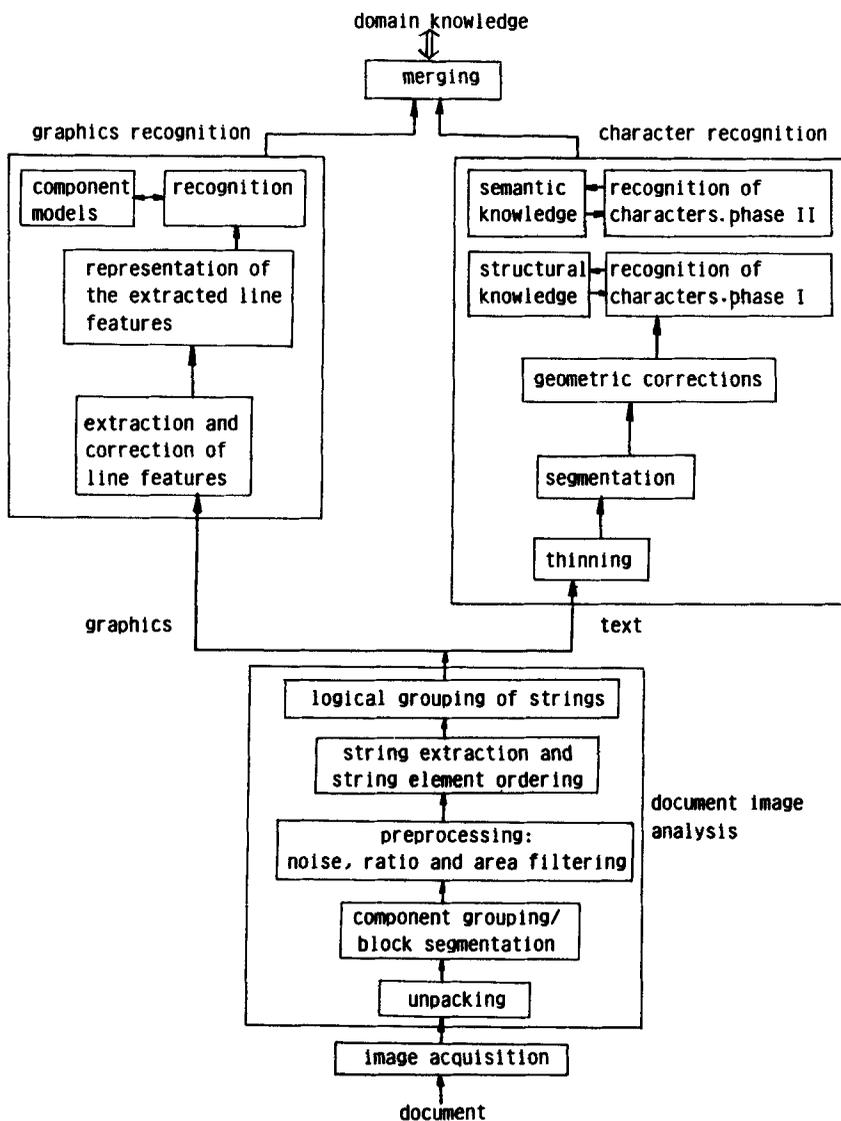


Figure 1. Data flow between levels in automated document processing system.

The goals of an automated document analysis system are the following: real time processing; compact output representation (vectors, ASCII); presentation of line coding; efficient data structures for easy access to elements; symbol and character identification irrespective of orientation and scaling; ambiguity and error resolution by applying syntactic and semantic constraints on the data (e.g., statistical experience, geometrical knowledge, application oriented rules).

The task of an automated document-processing system has been analysed at several conceptual levels. Within these levels there are several interacting modules as follows: image acquisition, unpacking, block segmentation, text line separation, grouping of text into words and phrases, graphics recognition, text recognition. The interactions between these processes is shown in figure 1.

## 2. Block segmentation

After the image is acquired, the image has to be segmented into distinct blocks. A brief review of the work done in block segmentation follows.

The layout structure of a document is complicated i.e. it may contain several columns of text in various fonts, pictures etc. Segmentation techniques based on recursive  $x - y$  cuts (Wong *et al* 1982) prove to be inadequate in this domain. Moreover it requires *a priori* processing if the document is skewed. Constrained Run Length algorithm (CRLA) (Wahl *et al* 1982; Wong *et al* 1982; Srihari & Zack 1986) is inadequate in the sense that text must be horizontally oriented in order to extract text clearly. Also the format is constrained, i.e. there must be vertical or horizontal rivers of white between text lines and between text and graphic entities. The Combined Symbol Matching algorithm (Pratt *et al* 1980) is sensitive to changes in text font style and size: the algorithm must be run once for each font type using different parameters, words of less than three characters which are embedded in longer strings are not removed, and the string removal is only along a specified orientation. The Bley algorithm (Bley 1984) is also sensitive to variations in text font styles and size within the image. In addition, this technique breaks graphics components into subcomponents, which is undesirable from the point of view of further processing for graphics recognition. Thus there is no one single algorithm which is robust enough to accommodate images with mixed font styles and sizes, strings of arbitrary orientation and images with mixed text and graphics. We have adopted a method suggested by Fletcher & Kasturi (1988) for connected component generation with modifications incorporated into it for improving speed and efficiency. The algorithm implemented by us (Mallikarjuna & Nagaraju 1990) for connected component method is a one-pass algorithm which merges horizontal collinear runs of black pixels into components that are connected to the line above. If there is more than one component connected they are all merged into one. This means that there need only be  $n + 2$  checks for connectedness for an  $n$ -pixel run.

Another advantage of this algorithm is that it requires only two 1-D arrays, for holding two image scan lines instead of the whole image as in case of the Depth First Search algorithm.

Connected components are isolated by raster scanning the image, isolating horizontal runs of pixels. The following rules generate connected components.

(1) If the current run is not connected to the previous run, then start a new component,

noting the top, bottom, left and right pixel coordinates of the run, and the number of pixels in the run.

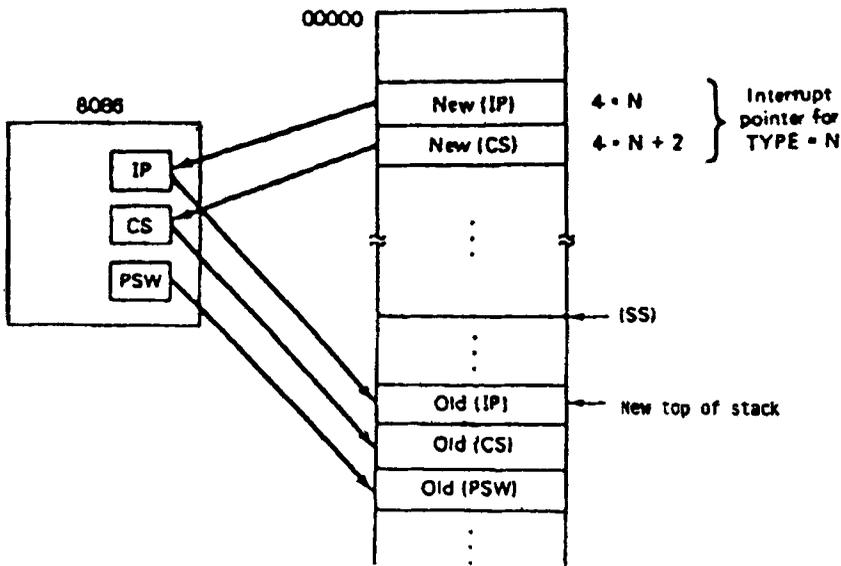
(2) If the current run is connected to exactly one component above, then merge the run into the component (i.e., add the number of pixels, adjust the bottom, and adjust the left-most and right-most pixel coordinates, if necessary).

(3) If the current run is connected to more than one component above, then merge the run into the lowest-valued (earliest connected in image) component, as in step (2), then merge all other components connected above into this lowest valued component.

Applying these three rules, results in an information array which has many invalid components (components which have been merged into another). A final garbage collection step removes these components. The array has the following characterizing data about each component: the left, right, top and bottom-most pixel coordinates of the component i.e. Minimum Bounding Rectangle (MBR), the area of the rectangle these values represent, and the pixel density. These data are used by the other stages of the segmentation algorithm, thereby minimizing the operations on the large-sized image array. The result of applying this algorithm to figure 2 is shown in figure 3.

2.1 Preprocessing: Noise/area/ratio filters

The noise filter simply removes very small components from consideration as text. This provides valuable savings, since the large number of pixels due to noise or lines containing dots will require the algorithm to consider many unnecessary groups of these pixels when grouping components into collinear strings. This filter will always



Note: The instruction being executed at the time the interrupt occurs is completed before the interrupt sequence begins

Figure 4-25 Interrupt sequence.

; determined by the type of interrupt, and clearing the IF and

Figure 2. Test image.



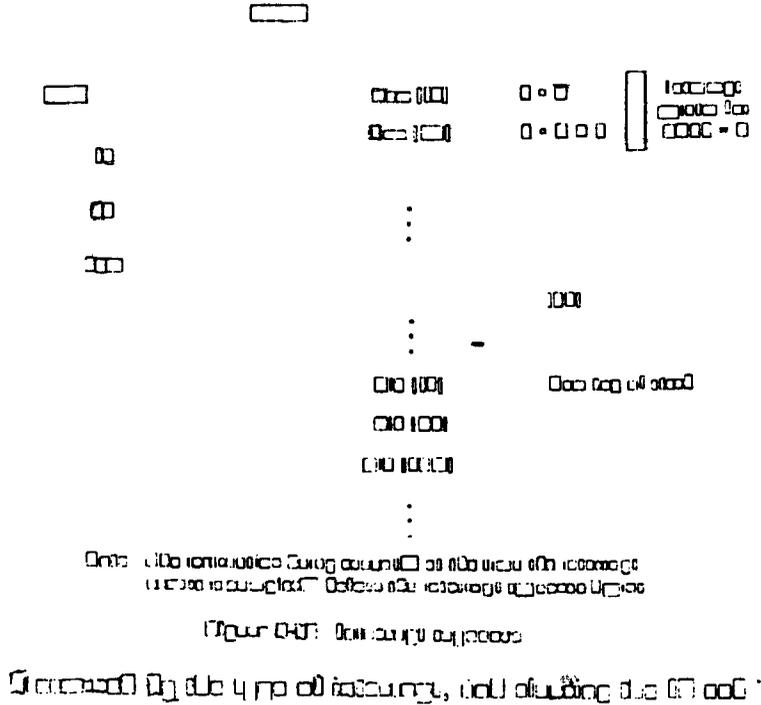


Figure 4. Connected components of test image after filtering.

## 2.2 Text line separation

**2.2a String extraction/ordering:** We have defined a text string as a group of at least three characters which are collinear and satisfy certain proximity criteria. This part of the algorithm identifies collinear characters by applying the distance criterion to the centroids of the connected components. In order to logically connect or group together characters into strings, it is necessary to determine which characters in the image actually lie along any given straight line. This is accomplished by sorting the text components on the y-coordinates of their centroids, so that the text is sorted into horizontal runs of components.

By grouping components into strings associated with a particular line, the components can be ordered along the line by sorting the components on their x-coordinates of the centroids.

**2.2b Grouping into words and phrases:** After a collinear string is extracted, it is checked by an area filter, which is similar to the one discussed earlier, so that the ratio of the largest to the smallest component in the group is less than 5. This is necessary to prevent large components, which do not belong to the string under consideration (but having their centroids in line with the string), from biasing the thresholds such as inter-character and inter-word gaps. The components are further analysed to group them into words and phrases using the following rules:

- (1) The inter-character gap less than or equal to  $A_h$ .
- (2) The inter-word gap less than or equal to  $2.5 * A_h$ .
- (3) Gaps  $> 2.5 * A_h$  indicate a phrase break.

Here  $A_h$  is the local average height and is computed using the four components which are on either side of each gap.

To be clear, a phrase is expected to be one group of words or symbols which are logically connected, i.e., they relate to the same expression, and will have the same font size. Once the phrases are generated, each phrase must satisfy the following criteria.

- (1) There are no oversized components in the phrase.
- (2) There are at least three components per phrase.
- (3) There are not more than two consecutive single component words within any phrase.

The check for oversized components is somewhat similar to the area filter, except that in this case only components which make up a given collinear string are considered. This check preserves uniformity in the size of characters within each string.

The string is refined according to some further heuristics. First, there must be no phrase containing fewer than 3 components. Also there can be no more than 2 single component words in a row. If the phrases do not conform to these rules, then the string is reorganized and the components are dropped until a valid text string is left.

Using the above criteria the ordered components are segmented to produce logically grouped strings.

### 2.3 *Post-processing to refine the segmentation*

The algorithm described above classifies broken lines (e.g., dashed lines) and repeated characters (e.g., a string of asterisks) as text strings since they satisfy all the heuristics for the text strings. It is desirable to label such components as graphics. On the other hand, text strings in which some of the components are connected to graphics are incorrectly segmented (e.g., underlined words in which characters with descenders touch the underline). Thus the strings which are identified are further processed to refine the segmentation. This is what we call post-processing.

**2.3a *Repeated character filter:*** Strings made up of a character repeated many times are separated from other strings by checking for consistency in variance of the pixel density, the area, and the ratio (of the length of the sides of the rectangle). These three variances are normalized by dividing by their thresholds. Whenever the product of the normalized variances is less than '1' it is classified as graphics. This method of combining the three features allows for some latitude in choosing the individual thresholds.

**2.3b *Dashed line filter:*** Although the above filter may work for certain dashed lines, it is easily seen that dashed lines are quite often irregularly broken, either for variety (e.g., lines made up of alternating short and long segments) or because they are hand drawn. Since dashed lines are so common in graphics, a specialized algorithm for their recognition is required. The only reliable characteristic of dashed lines is that the variance in thickness is low. Since this information is not available in the connected component data, it is obtained from the image. The thickness of the components at each point in a direction perpendicular to the direction of the string is calculated. While calculating this thickness the number of transitions from black to white,  $N_{,}$ , along the perpendicular is also calculated, which should be one for an

ideal line. When this number is larger than '1' at several points, the process is aborted to save processing time. Each thickness contributes to the variance calculation, except when the thickness is zero (inter-character gaps). The variance in thickness is then compared to a single threshold.

In these calculations, care is taken to include only those pixels which belong to the component under consideration, even when there are pixels belonging to other components within the rectangle. More details are available in Mallikarjuna & Nagaraju (1990).

### 3. Graphics recognition system

Line drawings and tables are commonly encountered in documents. Their analysis involves a raster to vector conversion. The aim of a raster to vector process is to convert a binary pixel representation of an image to a connected set of segments and nodes. A segment is typically a primitive such as a straight line or a parameter curve.

The vectorization of raster images has numerous applications including cartography, computer aided design, character recognition, conversion of drawings to edible form, minimization of electronic document storage space, extraction of structural features for automatic shape recognition etc.

Lin & Pun (1978) proposed a method to recognize hand drawn sketches. The pattern is denoted by a set of two-dimensional vector points, i.e., a subfigure. The system then estimates some parameters such as centre of gravity, figure size and the angle needed to estimate the orientation of the subfigure. Classification is performed using a specific discriminant function. Murase & Wakahara (1986) proposed an algorithm for on-line hand-sketched figure recognition. This algorithm has 3 steps: (a) candidate-figure extraction for each symbol based on a graph search and distance calculation between candidate figures and reference patterns; (b) selection of the symbol sequence which minimizes the total sum of these distances; (c) establishing connective relations among the symbols in the tentative recognition. The paper presented by Watson *et al* (1984) and its improved version by Bixler & Sanford (1985) aim at taking the scanned input and obtaining line information from it. Clement (1982) describes a way of changing the pixel resolution depending on the complexity of the image. His work did not address the problem of dealing with small objects having high information content.

We now examine how we have addressed the problem of extraction of structural features and its recognition. We consider recognizing electronic components in a complex circuit.

#### 3.1 Pre-processing

3.1a *Thinning*: Thinning plays a very important role in pattern recognition. It deals with extracting the distinctive features known as 'skeletons' from the patterns. It gets rid of all redundant points but preserves the basic structure and characteristic features of the pattern.

The algorithm implemented here was basically developed by Zhang & Suen (1984), modified by Lu & Wang (1985) and finally, again modified by Ethindra Kumar (1990).

This is a parallel algorithm. Using this algorithm, each pattern can be thinned down to a skeleton of unitary thickness and experimental results show that this algorithm is very good with respect to both connectivity and contour noise immunity.

3.1b *Segmentation*: The thinned image is segmented into meaningful recognizable features. Before understanding the extraction of segments from the given image, the definitions of the terminology used should be understood.

- \* End point is a point in the image having only one neighbour.
- \* Tree point is that point or pixel where we have more than one way to proceed in order to trace the segment.
- \* Turning point is that point or pixel for which the difference in the direction of its unmarked neighbour (not yet visited) and the direction of its last visited neighbour is more than 45 degrees.

*Algorithm 1. Extraction of the segments*

*Step 1.* Find all the end points and mark them as unvisited. Initialize segment count to zero.

*Step 2.* Take any arbitrary end point which is not yet visited. If not found, go to step 7.

*Step 3.* Store the point as the starting point of the current segment, mark it as visited, and traverse the image in the direction of its neighbour and mark it as visited. Store the directions in the vector to build up the chain code vector of the current segment.

*Step 4.* If while traversing, a tree point is encountered, mark it as unvisited which makes it an end point and keep on traversing in the appropriate direction.

*Step 5.* If a turning point is found, mark it as unvisited, i.e., make it an end point and mark it as the end point of the current segment. Increment segment count and go to step 2.

*Step 6.* If another end point is reached, then mark it as the end point of the current segment and as visited. Increment segment count and go to step 2.

*Step 7.* /\* All the segments have been extracted \*/ Output the segment count, chain code vector of each segment along with number of points traced, starting point and end point coordinates.

*End of the algorithm*

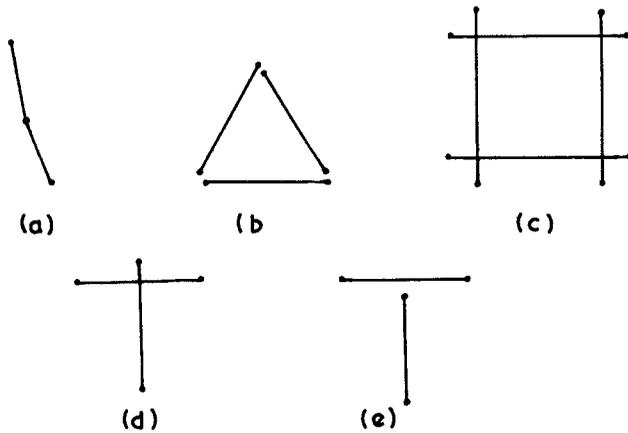
### 3.2 Geometrical correction and recognition

3.2a *Distortion correction*: The reasons for the distortions are casual sketching (in hand-drawn sketches) by the sketcher and some minor flaws in the feature extraction techniques used. The system has to be equipped to compensate for or to nullify the distortions present. The method used for feature extraction plays a major role in the deformation of the line features. The best method for a particular situation may not give desirable results for a different situation. Some of the cases of distortions are shown in figure 5. Figure 5a shows that the end points of the two-line segments are almost collinear and so a single straight line has to be formed out of these two lines.

Further, each point in the picture where two or more line segments meet (close to one of their end points), should be considered as a vertex. Hence, in the correction stage, the end points should be altered if there are such intersections. The possible cases where corrections may be needed are shown in figures 5b–e.

Possible corrections made are shown in figures 6 and 7.

3.2b *Recognition phase*: After the above geometric corrections are applied to the extracted segments they are then given as input to the recognition phase which requires proper representation of extracted segments. This is briefly discussed below:



**Figure 5.** Possible cases of distortions. (a) Collinear line segment. (b)–(e) Line segments for which corrections are required.

**Representation** – To solve any problem using a computer, the way in which the relevant information is organized, i.e., the data structure used to represent the lines, plays an important role. The selection of a suitable data structure for an algorithm makes the manipulation of the information easier, saves storage space, and often leads to an efficient method of finding a solution (Shapiro *et al* 1987).

Hence, in the object recognition stage, finding a data structure in which the features of the line drawings can be represented to ease the process of recognition is a major issue.

The adjacency list representation, with some modifications, best suits our purpose (Lakshminarayanan 1989). In the modified adjacency list representation, with each node in the adjacency list of the nodes in the main list, a pointer which points to its corresponding node in the main list is attached. This representation is suitable for the search method used for recognition. The modified adjacency list for the graph in figure 8 is represented in figure 9.

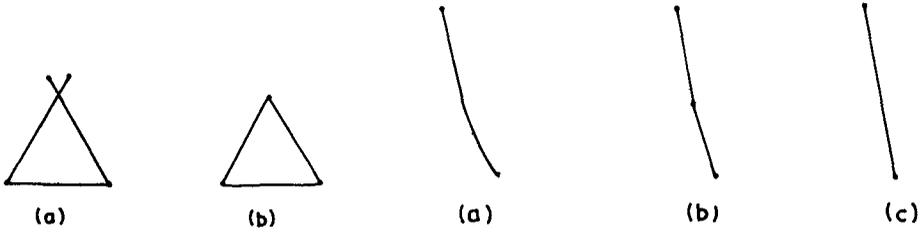
**Recognition** – Once the data structure for the representation of line features is finalised, the next issue is that of finding a suitable method for recognizing the objects in the line drawings.

**Example procedure for OP-AMP and IC recognition**

```

procedure op_amp_and_ic()
begin
  for "all v in the set of vertices" do
  begin
    search_cycle(v);
    if there is a cycle
    begin
      if "length = 3" print (OPAMP);
      if "length = 4" print (IC);
      remove_edges();
    end;
  end;
end;
end;

```



**Figure 6.** Example for line corrections. (a) Input sketch, (b) after correction.

**Figure 7.** Line straightening. (a) Input line, (b) line extracted, (c) corrected line.

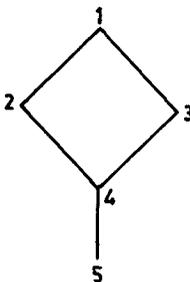
### 3.3 Results

The input sketch is shown in figure 10, the extracted segments in figure 11 and the corrected segments in figure 12. Components recognised are shown in figure 13. The system can be expanded to recognise some more components by appending appropriate rules in the recognition stage. By preserving the relationships among the components and by having pre-defined symbols for components, the output of the system can be improved to represent a better reproduction of the input sketch. By incorporating some knowledge about electronic circuits, the system can be made to understand the functioning of the circuit.

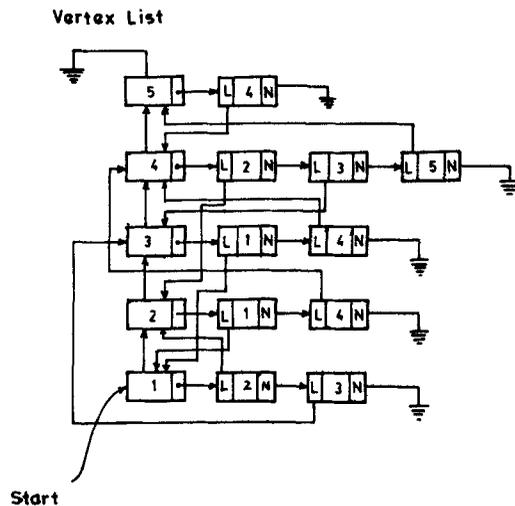
## 4. Character recognition

This section describes the development and implementation of a structural character recognition system relatively independent of text font and style.

There are three steps in the algorithm. The first step deals with the pre-processing of the input image through thinning and extraction of the chain code of distinguishable segments comprising the character image. The second step labels the extracted segment



**Figure 8.** An undirected graph.



**Figure 9.** Modified adjacency list. L: location pointer; N: next node pointer.

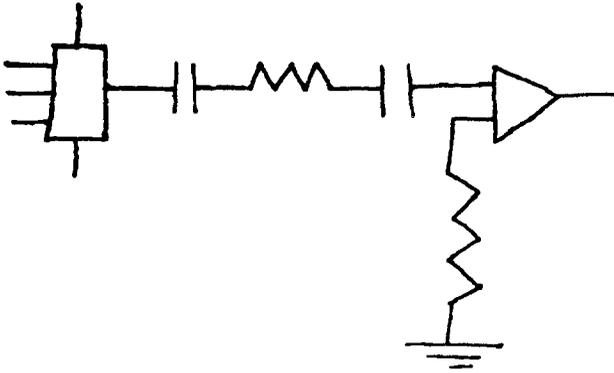


Figure 10. Input sketch.

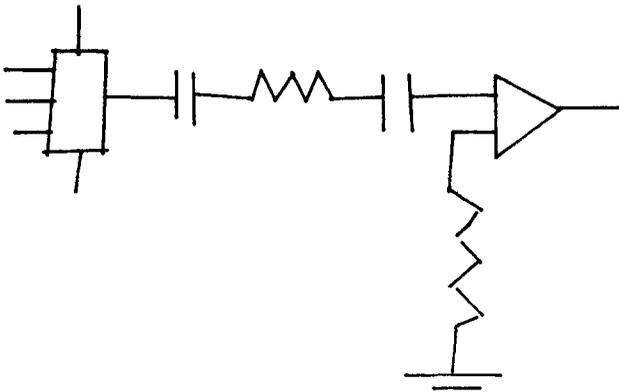


Figure 11. Extracted segments.

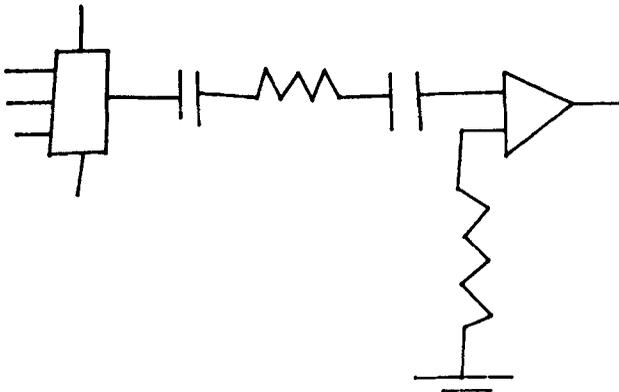


Figure 12. The corrected segments.

into eight primitive strokes and curves. The third and final step recognizes the character using topological information about the different segments extracted. The output of the system is the ASCII equivalent of the processed document.

Optical character recognition (OCR) has been the subject of intensive research efforts for roughly 30 years. The immense activity drawn to OCR is a testimony to its challenge and practicality. Clearly there is much motivation to provide automated text and data entry into computerised systems (Wolberg 1986). The definitions of the major schemes in OCR are as follows:

```

ic : (91 , 264), (108 , 89), (201 , 90), (193 , 263),
op_amp : (911 , 278), (911 , 136), (1030 , 192),
resistor : ( 824 , 656 ), ( 911 , 234 )
resistor : ( 361 , 165 ), ( 706 , 173 )
connector : (153 , 263) , (141 , 334)
connector : (154 , 12) , (151 , 89)
connector : (1030 , 192) , (1141 , 191)
connector : (104 , 122) , (17 , 121)
connector : (99 , 176) , (20 , 176)
connector : (328 , 171) , (197 , 168)
connector : (911 , 169) , (757 , 172)
connector : (94 , 230) , (33 , 230)
capacitor : (330 , 124) , (328 , 203) , (361 , 127) , (362 , 216)
capacitor : (707 , 133) , (706 , 228) , (755 , 132) , (760 , 229)
ground : (915 , 658) , (740 , 656) , (877 , 681) , (790 , 680)

```

Figure 13. Recognised components output.

- (1) Fixed-font character recognition is the recognition of specific fonts (OCR-A, OCR-B, Pica, Elite, Courier etc) typewritten characters.
- (2) On-line character recognition is the recognition of single hand-drawn characters, where not only the character image is provided but also the timing information of each stroke.
- (3) Handwritten character recognition is the recognition of single hand-drawn alphabetical characters which are unconnected and not written in calligraphy.
- (4) Script recognition is the recognition of unrestricted handwritten characters which may be connected and cursive.

The order of difficulty of character recognition problems is (starting from the most difficult): script recognition, handwritten character recognition, on-line character recognition that provides timing information as well as the character image, fixed-font character recognition and magnetic or mechanical characters.

#### 4.1 Major research work

One of the earliest attempts in character recognition was that of Grimsdale *et al* (1959) where in a pioneering work they described a method where the input pattern is scanned by a flying spot scanner; subsequently, it is analysed for shape by a digital computer which extracts the basic features. These features are compared to values, which have been stored in the computer, in order to reach the proper decision. Another work in that period was that of Bledsoe & Browning (1959) who described a rather more general approach using a photo mosaic of  $10 \times 15$  photocells. In the early 1960s, Eden (1968, pp. 138–54) at MIT put forward the idea that all Latin script characters can be formed by 18 strokes, which in turn can be generated from a subset of four strokes, called segments. This concept was known as the “analysis-by-synthesis method”. Casey & Nagy (1966) at IBM presented one of the first attempts in Chinese character recognition. Since the Chinese alphabet has about 1000 characters a step-by-step approach was applied. That is, they first grouped similar characters, and then “group masks” and finally “individual masks” were employed. However, later work showed that a number of strokes and their positions can be adequate for recognition of Chinese characters. Casey & Nagy (1968) introduced an unconventional approach to character recognition, the so-called “autonomous reading machine”, since no previous training or *a priori* information about the character was needed, but instead the known letter-pair frequencies of the language were used for recognition.

In the early 1970s, Parks *et al* (1968, 1974) produced a feature extraction method for extracting lists of topological features and a hierarchically structured articulate method of recognition. Tou & Gonzalez (1972) introduced a two-stage scheme where, in the first stage, measurements are made with a measurement grid in order to separate the pattern classes into several subgroups, and, in the second stage, a number of specialised features are extracted. In the mid 1970s, Pavlidis & Ali (1975), and Ali & Pavlidis (1977), making use of their split-and-merge algorithm, produced polygonal approximations of the characters which could provide enough information for both the character shapes and the syntactic analyser. A paper by Farag (1979) investigated the cursive script whole word recognition of a small dictionary of keywords such as may be found in handwritten computer programs or some business applications. Incoming script was digitised such that one sample was taken every 1/8 in. pen movement and was converted into an eight-direction Freeman chain-code. A dictionary of pre-stored words are available and classification made by a maximum likelihood classifier using a set of ten keywords, each written twenty times, produced current classification of 98%. Badie & Shimura (1982) have attempted machine recognition of Arabic cursive script using 14 basic curve features. The script was encoded as a Freeman chain-code before features were extracted. Feature points were placed at end points, points of inflection and cross points. These were stored in a graph matrix. Recognition was by identifying the basic strokes between feature points as one of the 14 features.

#### 4.2 Methodologies in character recognition

We have surveyed and determined the following methodologies applied in character recognition.

(i) *Point by point global comparison*: This process involves a point-by-point comparison of all the pixels that are stored with the pixels of the image.

*COMMENT*. Not resilient under deformation.

(ii) *Global transformations*: For example Karhunen–Loeve, Fourier, moment calculations and finally rotations according to the principal axis of inertia.

*COMMENT*. Not resilient under deformation. Structure not adequately captured.

(iii) *Extraction of the local properties*: Such local properties are the end points, the pronounced angles or corners, the *T*-junctions, and the crossing of strokes. This extraction can be achieved by various methods, e.g. by means of fixed masks or masks of variable positions. It is often prerequisite to thin the character and then to apply the above methods of extraction. This method (many papers have been published on this) has the drawback that it cannot recognise those characters in which segments are unconnected or overshoot.

*COMMENT*. Structures not adequately captured.

(iv) *Structural methods*: Such methods include decomposition of the character into its constituent elements, the topological description and the reduction of the character into a graph.

*COMMENT*. More resilient under deformation. Structure adequately captured. Not much work attempted along these lines.

We followed the structural methods approach. With this approach we could recognise printed characters of different sizes and fonts and are extending it to handwritten characters and script.

### 4.3 Design of character recognition system

A schematic block diagram of the proposed character recognition system is as shown in figure 14. A brief description of each block is given as follows.

The output from the separator comprises the coordinates of the Minimum Bounding Rectangle (MBR) for each character. This block is read from the whole image, then unpacked and stored in a two-dimensional array as the binary image of the character.

The binary image is passed through a thinning algorithm discussed in §3.1a to obtain the unitary skeleton of the image.

In the next module, main segments are extracted from the thinned image and the chain code vector of each segment is produced from this module. The chain code vector for each segment is obtained by convolving a  $3 \times 3$  window, with the grid representing the raster image of the input pattern from left to right and from top to bottom. The window is moved over each pixel, beginning from an end point which

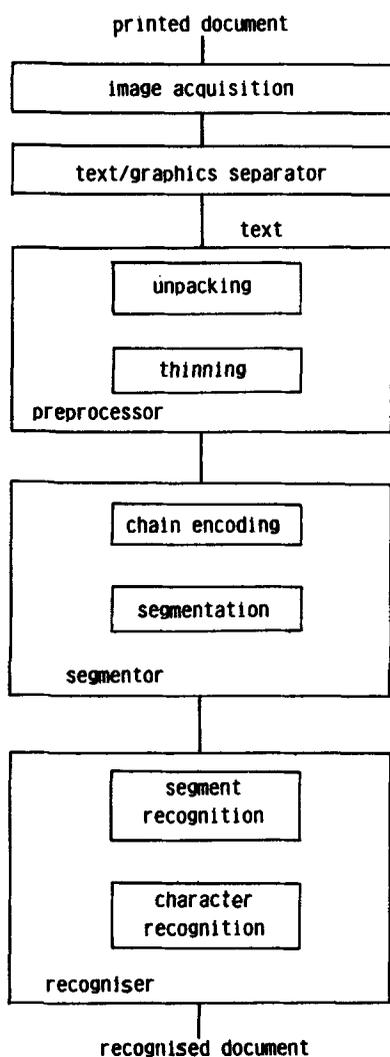


Figure 14. Schematic block diagram of the designed OCR system.

**Table 1.** Strokes and curves.

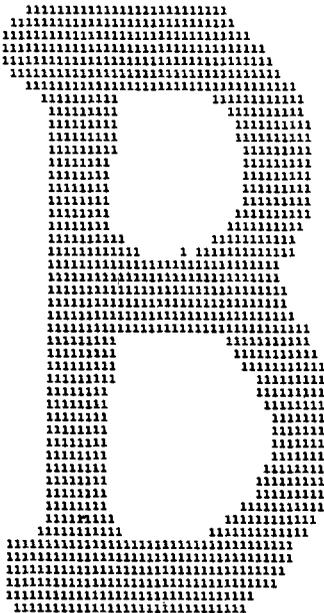
Generic instance	Specific instance				
	—		/	\	
Stroke	Horiz	Vert	Lvert	Rvert	
∪	∩	⊂	⊃	∪	
Curve	acurve	ccurve	dcurve	ucurve	

is not yet visited. If there is no end point, the first image pixel struck while scanning from left to right and from top to bottom, is taken as the starting point for extracting the segments from the character image.

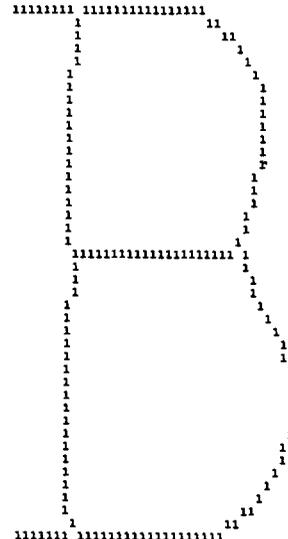
The chain code vector of each segment is used to draw a histogram for each segment representing the frequency of each direction in the chain code of that particular segment.

The histograms of all the segments are passed on to the next block, which recognises these segments in terms of the eight primitive strokes and curves as shown in table 1. In this block, certain segments are further subdivided so that those sub-segments can be labelled or recognised in terms of those eight primitive strokes and curves.

These labelled segments are now used for recognising the character in the next module. Some of the characters can be directly identified by the number of different segments present in its image. For other characters, relative orientation of various segments present in the image is checked to find out what character it represents. Further structural checks are applied to resolve any ambiguities between different characters.



**Figure 15.** Binary image of the character.



**Figure 16.** Thinned output obtained from function Thin().



RECOGNITION PHASE...

VARIOUS SEGMENTS OBTAINED AFTER LABELLING ARE :

VERTICAL LINES = 1  
 HORIZONTAL LINES = 3  
 D\_CURVES = 2

SEGMENTS AFTER MERGING AND REMOVAL :

HORIZONTAL LINES = 3  
 VERTICAL LINES = 1  
 D\_CURVES = 2

THE CHARACTER IN THE GIVEN BLOCK IS : B  
 ITS ASCII VALUE IS : 66

Figure 18. Final output obtained from the labelling process.

FONT (A)	{	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789:;<=>?@{ }~■ !"#%&'()*+,-./[\]^_`
FONT (B)	{	ABCDEFGHIJKLMNOPQRSTUVWXYZ ABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789:;_=#?@§¶†™■ !"#%&'()*+,-./[*]°_°
FONT (C)		ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789:;<=>?@{ }~☼ !"#%&'()*+,-./[\]^_`
FONT (D)		ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 0123456789:;<=>?@^ !"#%&'()*+,-./[\]~{ }_☼

Figure 19. Fonts tested with the present character recognition system.

**Table 2.** Results obtained from the present system.

Font	Results (%)
(A)	100
(B)	95
(C)	95
(D)	90

## 5. Conclusions

Document image understanding is a complex task that begins with pixel processing and ends with text and graphics recognition. Thus it is an area of research that draws upon techniques of image processing, pattern recognition and artificial intelligence.

Starting with a digitized document image, we are applying different techniques to analyse the whole image in some instances and specific parts of the image in other instances. The high resolution binary input image is segmented into a text string image and a graphics image. Then line segments with distortions are recognised. It also recognises hand drawn schematic electronic circuits and identifies the electronic components. Processing is then done on the text string image to recognise the text. In this paper we try to recognise text of any font. The system generates an output description file which has adequate information to faithfully reconstruct the original. The algorithms are computationally intensive and the results obtained are encouraging.

## References

- Ali F, Pavlidis T 1977 Syntactic recognition of handwritten numerals. *IEEE Trans. Syst. Man Cybern.* 7: 537–541
- Badie K, Shimura M 1982 Machine recognition of Roman cursive scripts. *Proc. 6th Int. Joint Conf. on Pattern Recognition*, Munich (New York: IEEE) pp. 28–30
- Bixler P, Sanford P J 1985 A technique for encoding lines and regions in engineering line drawings. *Pattern Recogn.* 18: 367–377
- Bledsoe W W, Browning I 1959 Pattern recognition and reading by machine. *Proc. Eastern Joint Computer Conference*, pp. 225–232
- Bley H 1984 Segmentation and preprocessing of electronic schematics using picture graphs. *Comput. Vision, Graph. Image Process.* 28: 271–284
- Casey R, Nagy G 1966 Recognition of printed Chinese characters. *IEEE Trans. Elec. Comput.* 15: 91–101
- Casey R, Nagy G 1968 An autonomous reading machine. *IEEE Trans. Comput.* 17: 492–503
- Clement T P 1982 Extraction of line structured data from engineering drawings. *Pattern Recogn* 14: 43–52
- Eden M 1968 Handwriting generation and recognition. *Recognizing pattern* (eds) Kolars, M Eden (Cambridge, MA: MIT Press)
- Ethindra Kumar R 1990 *Offline shorthand recognition system*, M Tech project report, University of Hyderabad, Hyderabad
- Farag R F H 1979 Word level recognition of cursive script. *IEEE Trans. Comput. C-28*: 172–175
- Fletcher L, Kasturi R 1988 A robust algorithm for text string separation from mixed text/graphics images. *Pattern Anal. Mach. Intell.* 10: 910–918
- Grimsdale R L, Sumner F H, Tunis C J, Kilburn T 1959 A system for the automatic recognition of patterns. *Proc. Inst. Electr. Eng.*, 106: 210–221

- Lakshminarayanan V 1969 *2D schematic understanding system*, B Tech project report, University of Hyderabad, Hyderabad
- Lin W C, Pun J H 1978 Machine recognition and plotting of hand-sketched line figures. *IEEE Trans. Syst. Man Cybern.* 8: 52-57
- Lu H E, Wang P S P 1985 An improved fast parallel thinning algorithm for digital patterns. *Proc. IEEE Conf. Comput. Vision Pattern Recogn.* (New York: IEEE Press)
- Mallikarjuna B, Nagaraju A 1990 *Automated document processing system*, M Tech dissertation, University of Hyderabad
- Midha M, Mahajan R K 1990 *A structural character recognition system*, MCA project report, University of Hyderabad, Hyderabad
- Murase H, Wakahara T 1986 On-line hand-sketched figure recognition. *Pattern Recogn.* 19: 147-160
- Parks J R, Bell D A, Watson R S, Cowin G, Olding S E 1974 An articulate recognition procedure applied to handprinted numerals. *Proc. 2nd Int. Joint Conf. on Pattern Recognition*, Copenhagen, pp. 416-420
- Parks J R, Elliot R J, Cowin G 1968 Simulation of an alphanumeric character recognition system for unsegmented low quality print. *Proc. IEEE and NPL Conf. on Pattern Recognition*, pp. 95-105
- Pavlidis T, Ali F 1975 Computer recognition of handwritten numerals by polygonal approximation. *IEEE Trans. Syst. Man Cybern.* 5: 610-614
- Pratt W K, Capitant P J, Chen W, Hamilton E R, Wallace R H 1980 Combined symbol matching facsimile data compression system. *Proc. IEEE* 68: 786-796
- Shapiro L G, Mac Donald R S, Sternley S R 1987 Ordered structural shape matching with primitive extraction by mathematical morphology *Pattern Recogn.* 20: 75-90
- Srihari S N, Zack G W 1986 Document image analysis. *Proc. 8th Int. Conf. on Pattern Recognition* (New York: IEEE) pp. 434-438
- Tou J T, Gonzalez R C 1972 Automated recognition of handwritten characters via feature extraction and multilevel decision. *Int. J. Comput. Inf. Sci.* 1: 43-65
- Wahl F M, Wong K Y, Casey R G 1982 Block segmentation and text extraction in mixed text/image documents. *Comput. Vision Graph. Image Process.* 20: 375-390
- Watson L T, Arvind K, Ehrich R W 1984 Extraction of lines and regions from graytone line drawing images. *Pattern Recogn.* 17: 493-507
- Wolberg G 1986 A syntactic omni-font character recognition system. *IEEE Comput. Soc. Conf. on Comput. Vision Pattern Recogn.* (New York: IEEE) pp. 168-173
- Wong K Y, Casey R G, Wahl F M 1982 Document analysis system. *Proc. 6th Int. Conf. Pattern Recognition*, Munich
- Zhang T Y, Suen C Y 1984 A fast parallel algorithm for thinning digital patterns. *Commun. ACM* 27: 236-239