J. Indian Inst. Sci., Nov.–Dec. 1996, 76, 813–826.

AMUS—an automatic map-understanding system

R. VEENA VANI, K. JAIRAM HEBBAR AND P. SUBHADRA*
Computer Processing Group, National Remote Sensing Agency, Bala Nagar, Hyderabad 500 037, Andhra Pradesh, India.
e-mail: jhk %nrsa-hyd@uunet.in

Received on November 29, 1994; Revised on June 2, 1995.

Abstract

The main obective of automatic map-understanding system (AMUS) is to automatically recognise and extract various themes present in a map and input them into a geographic information system. This paper presents the details of automated text string separation from linear features. The algorithm used is relatively independent of changes in the text font, size and also orientation invariant. The system outputs two images, the text strings and the graphics information. Then the characters (in the text string file) are recognised by applying artificial neural network techniques. Experimental results are presented in the paper.

Keywords: Threshold, vectorization, symbol recognition, artificial neural networks, pattern recognition, preprocessor, geographic information system (GIS).

1. Introduction

Recent years have seen increasing use of computers in office automation, engineering and mathematical applications. Land records, architectural maps, road maps and other geographical maps are generally stored in the form of blueprints or other types of hard copies. This has resulted in hampering computerization in various sectors of government wherein spatial data is essential. Cadastral maps (land register maps) are generally used at Panchayat or Mandal level. These maps are generally in scales of 1:4000/1:5000/1:10000. Typically, they show details like housing plots, roads, lanes, drainage, etc., and hence are important sources of information for planning and execution.

Conventional methods of digitizing are expensive and time consuming as a large number of Cadastral maps are to be digitized. Unless quick, convenient methods are used, it is a herculean task. Cartographic database creation involves the acquisition of huge amounts of data from paper drawings. Effective operational techniques for automatic digitization of drawings into a database are not universally available and many efforts in this direction over the past 20 years have found limited success. Recently, however, substantial advances have been achieved in this field.

AMUS—a tool for automatic map understanding—is the experimental system for the automatic aquisition of land register map which describes the geometry of land proper-

* Current address: 7921, Bramble Wood Drive, Cross 1A, LANSING, Michigan + 48917, USA

ties and buildings in a geographical context. They divide the territory into a number of polygons, each representing a piece of land, a building, a street or a body of water. The development of AMUS is for conversion of paper-based documents into digital form and finally for integration into a spatial database. This system relies upon and is driven by the semantics of land register maps[1].

2. Previous work

Several algorithms for automatic document analysis have been presented in literature. In the Linetrac system of Baker *et al.*[2] a hardware processor was developed to convert raster-scanned contour line data into vectorized digital form by scanning, thresholding, noise reduction, line thinning and line following. Seuffert[3] presents a system for acquiring map data. Users can scan maps over a drum scanner before processing the maps through the phases of thresholding, line thinning, line following, data compression and separation of characters based on size criteria. Finally, the user can edit the map manually. The Sysscan system, described by Leberl and Oslon[4], uses raster scanning with a white light and a CCD array sensor. An automated editing process supports the operator intervention.

Harris *et al.*[5] describes a general system that automatically interprets the binary pixel representation of an image on the basis of a predefined model. The main components are raster-to-vector(RTV) conversion rule, a database module, a symbol extraction module, and a polygon-recognition module for mapping. Harris also provides an overview of an algorithm for determining true centre line and connectivity.

Intergraph[6] offers large-format drum scanners. These scanners recognise arcs and ellipses as line strings and use optical character recognition (OCR) for typed characters. The original pixel image can be displayed as a background overlay to support manual editing and for verifying results. Clement *et al.*[7] describe another experiment for the vector form representation which considers drawings as line segments. Nackunstz[8] presents a video scan approach wherein the scanning resolution is adjusted automatically by means of a computer-controlled zoom objective. An algorithm for line detection and line following, which allows processing from the stored grey-level image without skeletonization of binary images is described. SKANTEK system[9] uses a CCD camera with fibre-optic technology. The system separates text from graphics and vectorizes the geometry and unrecognised text as line segments. However, during editing, the operator can overtype the alphanumeric characters to change from vector to ASCII form and therefore improve on data compaction. The system offered by Metagraphics[10] provides means for automating digitization. First, it scans a drawing and then the operator guides the system to identify the text. The operator traces over the geometry on the screen.

3. AMUS overview

Automating the process of digitization, vectorization and interpretation of maps, charts and diagrams is of great importance as described earlier. AMUS consists of the process

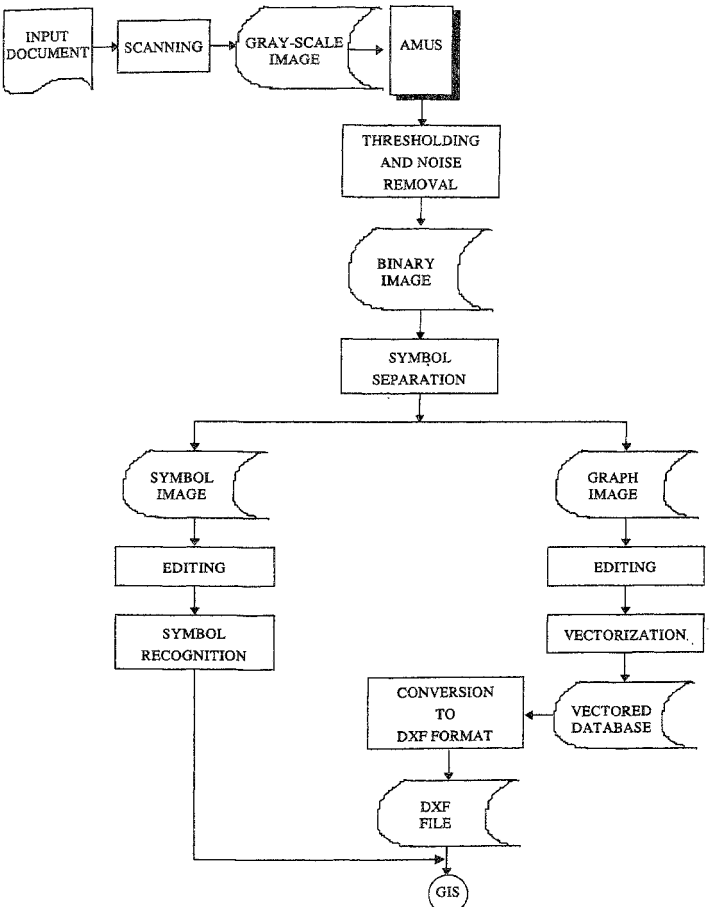


FIG. 1. AMUS implementation chart.

of symbol separation and recognition and vectorization of the graphic entities based on simple heuristics.

Processing a map begins with its digitization by a scanning device. A key step in the system is the conversion from raster format to graph representation, a special binary image format suitable for processing line structures. Subsequent steps include vectorization of the line structures and recognition of the symbols interspersed in the drawing. Editing is required to resolve ambiguities and correct errors in the process. The final result is a set of descriptors for all detected map entities, which is then stored in a database compatible to GIS. A block diagram of the overall system is shown in Fig. 1. The key elements of AMUS are: (i) separation of text from graphics, (ii) character recognition, and (iii) vectorization of graphic entities.

3.1. *Separation of text from graphics*

In order to efficiently interpret maps or drawings, it is necessary to separate text from graphics.

Several algorithms for text and graph separation have been presented in literature. Many of them are very restrictive and are therefore not useful in a general document-understanding system. For example, Chen *et al.*[11] presented symbol-matching algorithm, which is sensitive to text font and size. The block segmentation technique of Wahl *et al.*[12] broadly classifies regions into text and graphics. The algorithm of Bley *et al.*[13] is also sensitive to variation of text font and size.

The algorithm presented here is mainly derived from simple heuristics based on the characteristics of text strings as given by Fetcher *et al.*[14] The algorithm is designed to separate text strings from graphics, regardless of string orientation or font, size or style. This cannot recognise individual characters but can locate potential text string and separate it from the map, generating two files (one for text strings and the other for graphic entities). The block diagram of the process is depicted in Fig. 2.

Connected component generation involves grouping together black pixels which are 8-connected to one another (assuming a black image on white background). In this technique, 8-connected pixels belonging to individual characters are enclosed in circumscribing rectangles. The output of the connected component-generation algorithm is an array which specifies the maximum and the minimum coordinates of the top and bottom sides of each connected component.

A priori knowledge about the document type (such as the amount of text data compared to graphics), and its area facilitates the computation of the threshold to separate text from linear features. This is not possible always. So a flexible procedure of symbol height estimation by taking histogram of the connected components has been adopted. This yields better results. Very long lines are unlikely to be text characters and should be discarded. Isolated elements whose length lies within the threshold are considered as text. The peak of the histogram is taken as the typical character height. Components with low height are considered as noise and filtered out. Isolated straight lines within the image may be discarded on the basis of dimension ratio (for example, if a component has a ratio of less than 1:20 or greater than 20:1, it is discarded). These aspects are taken care of in the 'filtering approach'.

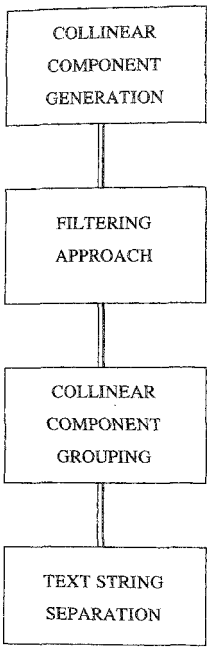


FIG. 2. Block diagram of text string separation.

Collinear component grouping involves grouping of clusters of text into a single block. This is achieved on the basis of satisfying the inter-text spacing threshold value. Once all strings are extracted from the graphic entities, they are deleted from the image array leaving behind graphic entities. The output at this stage consists of two binary files (the graphic entities and the text). Individual characters are extracted from the text string file and then submitted for recognition.

3.2. *Character recognition*

A symbol is extracted from the text file and is recognised. The output of this phase is the symbol and the coordinates at which the symbols have been found. Symbol recognition is basically a pattern-recognition problem. Any recognition system for character classification should be rotation, scaling and translation invariant. This effect is typical, when the

scanning device (suppose a camera) changes its orientation or distance from the specimen. To achieve this, either the system should employ features that are transformation invariant or there should be a preprocessor to take care of the rotational, scaling and translation invariancy as given by Yuckler *et al.*[15]

A two-stage pattern classification system based on a pattern preprocessor and an artificial neural network (ANN) classifier that can recognise patterns even when they are deformed by transformation of rotation, scaling, and translation or a combination of these has been developed.

3.2.1. *Artificial neural networks*

ANN are computational models and massively parallel networks comprising a large number of simple processing units (called artificial neurons).

Two approaches to classification using ANN have been tried. The first extracts features and then employs ANN to perform the classification on the extracted features. A second and newly developing approach lets the features be determined and extracted by the ANN itself. Backpropagation model[16] of Rumelhart is widely used and hence has been adopted for symbol recognition.

3.2.2. *Architecture of the recognition system*

In view of the specific objective of recognition of the characters extracted from a Cadastral map, the system design was influenced by the following considerations :

(i) The system must be able to deal with characters of different font, size, and deformed by transformation (rotation, scaling and translation or a combination).

(ii) Recognition should take very little time. Each module of the system has to be optimised to achieve a high recognition speed.

The character recognition system (Fig. 3(a)) is modular, consisting of three main blocks, a preprocessor, a recogniser and an interpreter. The blocks are cascaded to enable the digital image to be first preprocessed, and then classified.

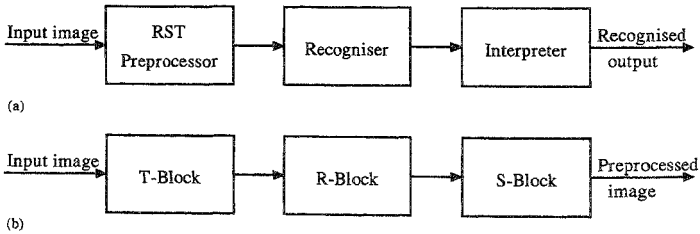


FIG. 3. Block diagram of (a) character recognition system, and (b) RST preprocessor.

3.2.2.1. *Input image*

The processing of the map begins with its digitization by a scanning device. The raster image is converted to a binary image using thresholding algorithm. Subsequent steps include separation of text from graphics and vectorization of the graphic entities. Individual characters extracted from text string file are input to the preprocessor.

3.2.2.2. *RST preprocessor*

The preprocessor (Fig. 3(b)) has three cascaded blocks T-, S- and R-blocks. T-block maintains translational invariancy by computing the centre of gravity of the pattern and translating the pattern so that the centre of gravity of the pattern coincides with the origin. S-block maintains the scaling invariancy by scaling the image so that the average radius for the ON pixels is equal to one fourth of the grid size. The R-block maintains rotational invariancy by rotating the image. The preprocessed characters are then input to the recogniser.

3.2.2.3. *Recogniser (N.N)*

The current implementation of the system employs a multilayer feed-forward network for the recogniser block. Such a network has a layered structure and only connections between neurons at subsequent layers are permitted. The training algorithm is the widely used backpropagation algorithm. Since the input is an image in pixel-map form, the number of nodes in the input is fixed and equal to the number of pixels. Further, since the output neurons are organised such that each node represents a class, the number of output neurons is also fixed. The backpropagation network consists of a weight initialiser, a training module and a recognition module. The weights are initialised to small random values. Supervised training is used for teaching the network.

(a) *Training*

The drawback of the backpropagation algorithm is that the time taken for 'training' is very high. Reducing the training time is one of the design parameters which has to be taken care of to get a usable system. Hence, SuperSAB strategy adopted by Tom Tollenaere[17] (an adaptive acceleration strategy for error backpropagation learning) has been adopted. With this, a drastic reduction in 'training' time was achieved.

The heuristics for increasing the rate of convergence (improving training time) are summarised:

1. Every weight should have its own individual step size.
2. Step sizes should be allowed to vary over time. In order to take appropriate steps as the weight varies over its possible values, the step size should be changed accordingly.
3. When the derivative of a weight possesses the same sign for consecutive steps, the learning rate for weight is increased.
4. When the derivative of a weight changes sign, the learning rate is decreased.

Based on the above, SuperSAB strategy is outlined here. Let $n+$ be the increase factor and $n-$ be the decrease factor in learning rate for the step size, and n-start the initial value for n_{ij} for all i, j.

* Set all n_{ij} to an initial value n-start;
* Do backpropagation step with momentum parameter;
* For each W_{ij}, as long as the W_{ij} derivative does not change sign, set

$$n_{ij}^{n+1} = n + n_{ij}^{n};$$

* When a change in the sign of the W_{ij} derivative is detected;
* Undo the previous weight update (which caused the change in gradient sign). This can be done by using

$$\delta w_{ij}^{n+1} = -\delta w_{ij}^{n}$$

* Set $n_{ij}^{n+1} = n - n_{ij}^{n}$;
* Then set $\delta w_{ij}^{n+1} = 0$ which causes the next invocation of step 2 not to take the step previously made into account.

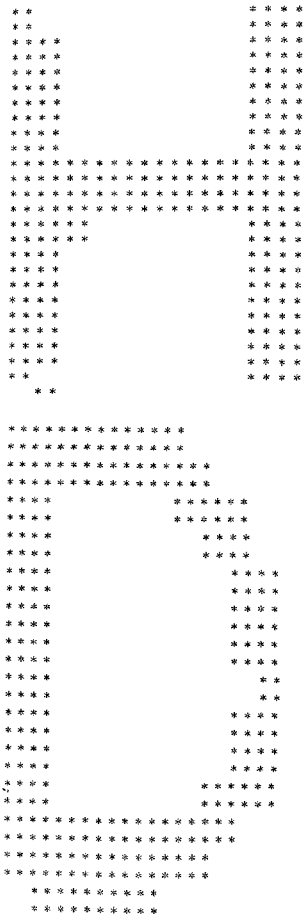
(b) *Modular approach*

In addition to SuperSAB strategy the modular approach implemented reduced the 'training' time considerably. This also has the advantage of new module can being trained individually and added to the original trained network. Each canonical set of 26 alphabet has been given in 5 nets (A to E, F to J, K to Q, P to T, U to Z), each having a 5–20–5 network structure. All these five nets are trained individually to recognise the characters present in that set and finally trained for overall integration of the network to branch to one of the five nets. This process is repeated for all canonical patterns produced by the preprocessor.

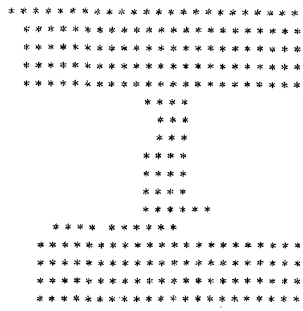
(c) *Interpreter*

One of the key determinants of the system performance is the success in the interpretation of the recogniser outputs. Since we have employed an artificial neural network for the recogniser, the classification result will be in the form of activation values of the neurons in the output layer. The first alternative is to use a simple maximum finder block for the interpreter. However, the performance will be moderate since it will always decide on one of the classes whether the class chosen is dominant on others or not. Thresholding can be applied to the outputs so that a class is selected if the outputs exceed this predetermined threshold. However, this method may indicate multiple classes for certain inputs. A more promising method is to report no discrimination as long as the ratio of the maximum output to the next highest output remains below a predetermined threshold value. When the ratio exceeds the threshold, which means that the maximum output is dominant on the other outputs, the interpreter decides on the class with maximum output. The interpreter block of our system employs the last approach. If the ratio of the maximum output to the next highest output exceeds the predetermined threshold, the interpreter reports that a discrimination could be made and the pattern belongs to the class with the maximum output. If not, no unique discrimination could be

CHARACTERS EXTRACTED FROM MAP



PREPROCESSED CHARACTER 'H'



PREPROCESSED CHARACTER 'D'

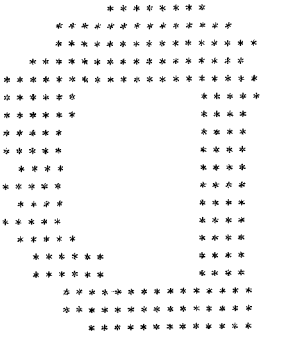
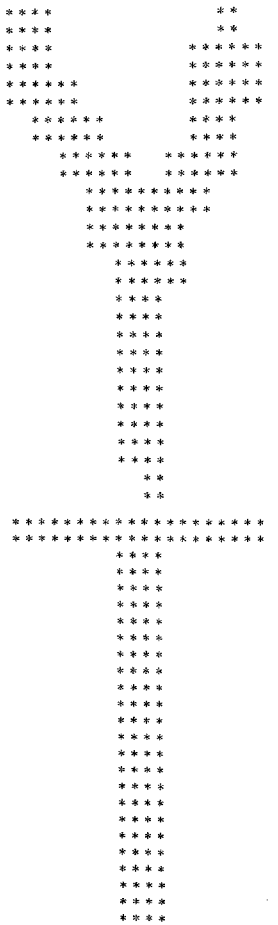
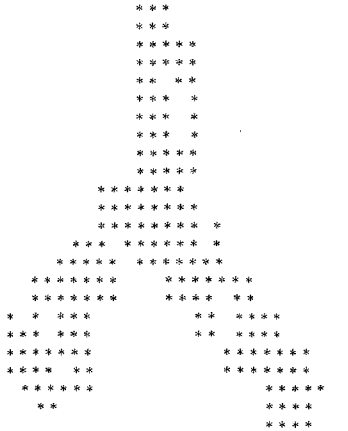


FIG. 4(a).

CHARACTERS EXTRACTED FROM MAP



PREPROCESSED CHARACTER 'Y'



PREPROCESSED CHARACTER 'T'

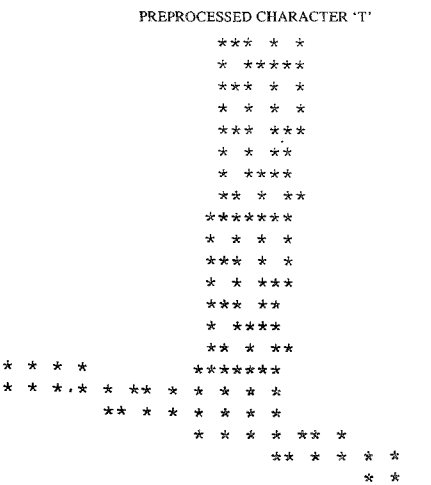


FIG. 4(b).

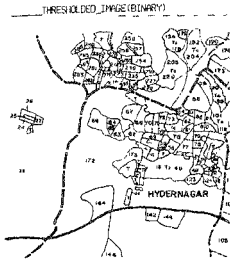


FIG. 5. Original input image (binary).

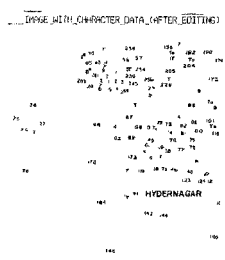


FIG. 6. Image with linear data.

made. This simple method has been observed to perform well in the evaluation of the recogniser outputs.

3.3. *Vectorization*

Symbol separation produces two binary outputs (character data, graphic entities). The graphic output is vectorized based on line-following technique. The vectorized output preserves the thickness of the line and also stores the direction along with the coordinate end points.

4. Implementation and results

AMUS is developed on a UNIX-based system. The Cadastral map of Hyderabad is taken as test data. This is scanned at a sampling rate of 200 dpi with a dynamic range of 256 grey levels. The resultant output is a 800×1250 grey-level image. The grey image is threshold to obtain a binary image (Fig. 5). For this test data, semantic thresholding[18] yielded better results. The binary image is further processed as described earlier. The total processing time (CPU time + user interaction time) was (approximately) about 30 minutes. The images with linear and text data (after separation) are shown in Figs 6 and 7, respectively.

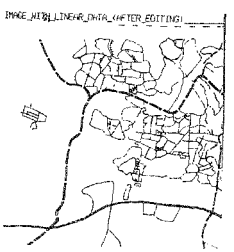


FIG. 7. Image with character data.

4.1. *Artificial neural network design details*

The size of the characters extracted from the map was approximately 32×32 pixels. Hence, a multilayer feed-forward network with 1024 input nodes, each corresponding to one of the pixels of the 32×32 output image and 26 output neurons (one for each alphabet) has been designed. An output close to 1 is interpreted as a strong membership, while a value close to 0, a loose one. It is seen that a network with 1024 input nodes, 20 neurons in the single hidden layer and 26 neurons in the output layer performs the best. Since the network is modular, nets with 5–20–5 structure are considered and trained individually. In the training phase, the network is trained on the canonical example patterns (which are outputs of the alphabets submitted to the preprocessor) until it manages to successfully classify the characters.

Individual characters extracted from the text string are submitted to the preprocessor and the resultant outputs to the neural network. The outputs are shown in Fig. 4. The results are interpreted as shown in Table I. All map-extracted characters are tested and finally tabulated (Table II).

5. Conclusion

In this paper, character separation from graphics and then recognition of characters by ANN, followed by vectorization of the graphic entities, is presented. The graph image representation of AMUS is advantageous for interpreting land register maps. It provides a formal description of the image's topological and metrical properties, which has proved successful in solving many problems of automatic interpretation of line drawings. The system successfully handles real maps that often contain formal errors and semantic ambiguities. It is fault tolerant and robust. Moreover, it is based on more general concepts and methodologies which makes it suitable for the interpretation of other kind of line drawings. Menu-driven and user-friendly features are built in. This system is not restrictive to Cadastral maps or engineering drawings, but will find application, in other user areas. The preprocessor takes care of all rotational, translational and scaling distortions, and enhances the recognition accuracy of the characters. The classification system of AMUS is independent of the application domain. Features like thinning, deglitching, smoothing of the lines in the maps and different format conversions which are compatible to GIS can be added to this tool. The creation of large geographic database for civic purposes is a pressing application of such an automated tool.

Table I
Interpretation of results

Input pattern	*Label*	*Output*	*Discrimination ratio*
H	H	0.91179	19.5
	K	0.0467	
D	D	0.8184	17.3
	B	0.047264	
Y	Y	0.619685	17.4
	K	0.03167	
T	T	0.8654	18.6
	I	0.0466	

Table II
Percentage of correct classification

Transformation	*Preprocessor*
Scaling	90%
Translation	80%
20% noise	91%

Acknowledgements

Acknowledgements are due to Prof. B. L. Deekshatulu, Director, National Remote Sensing Agency (NRSA), Hyderabad, for constant encouragement to work in this area. The authors also thank Mr T. Karunakar for helpful suggestions and critical review of the paper.

References

1. BOATTO, L. *et al.* — An interpretation system for land register maps, *IEEE Computer*, 1992, **25**, 25–33.

2. BAKER, D. J. *et al.* — LINETRAC—A hardware processor for scanning and vectorizing contours, and associated systems software for automatic generating digital terrain elevation data, *Tech. Papers Am. Soc. Photogrammetry*, APM-ACSM Fall Technical Meeting, San Francisco, Sept. 1981, pp. 9–11.

3. SEUFFERT, P. — An application of line and character recognition in cartography, *Proc. IEEE Pattern Recognition and Image Processing Conf.*, 1977, pp. 338–343.

4. LEBERL, F. W. AND OSLON, D. — Raster scanning for operational digitizing of graphical data, *Photogrammetric Engng Remote Sensing*, 1982, **48**, 615–627.

5. HARRIS, J. F. *et al.* — Modular system for interpreting binary pixel representations of line-structured data, *Pattern Recognition Theory Applic.*, 1982, 311–351.

6. — *Scanned data capture system*, Intergraph Corp, Huntsville, Ala., 1983, pp. 1.1–1.14.

7. CLEMENT, T. P. — The extraction of line-structured data from engineering drawings, *Pattern Recognition*, 1981, **14**.

8. NACKNUSTZ — An automatic system for digitizing line drawings, *Proc Sixth Int. Conf. on Pattern Recognition*, Munich, Oct. 1982.

9. HARRIS, M. A. — SKANTEK fiber optic scanner shrinks digitize that enters drawings into CAD system, *Electronics*, 1983.

10. KLEIN, S. — Metagraphics Inc. tackles CG'S big bottleneck: Converting drawings to a CAD/CAM format, *S. Klein Newsl. Comput. Graphics*, 1984, **6**, 1–2, and private communication.

11. CHEN, W. H. *et al.* — Combined symbol matching facsimile data compression system, *Proc. IEEE*, 1980, **68**, 786–796.

12. WAHL, F. M. *et al.* — Block segmentation and text extraction in mixed text/image documents, *Computer Vision, Graphics, Image Processing*, 1982, **20**, 375–390.

13. BLEY, H. — Segmentation and preprocessing of electrical schematics using picture graphs, *Computer Vision, Graphics, Image Processing*, 1984, **28**, 271–288.

14. FETCHER, L. A. *et al.* — A robust algorithm for text/graphics images, *IEEE Trans.*, 1988, **PAM-10**, 910–918.

15. YUCEER, C. *et al.* A rotation, scaling and translation invariant pattern classification system, *Pattern Recognition*, 1991, **26**, 687–710.

16. RUMELHART, D. E. AND WILLIAMS, R. J. Learning internal representations by error propagation, *Parallel distributed processing* (D. E. Rumelhart and J. L. McClelland, eds), Vol. 1, pp. 318–362, 1986, MIT Press.

17. TOLLENAERE, T. SuperSAB: Fast adaptive backpropagation with good scaling properties, *Neural Networks*, 1990, **3**, 561–573.

18. AVIAD, Z. AND IOZINISKII, E. Semantic thresholding, *Pattern Recognition Lett.*, 1987, **5**, 321–328.