renal disease," *Electroenceph. Clin. Neurophysiol.*, vol. 39, pp. 377–388, 1975.
[3] J. R. Bourne *et al.*, "Computer quantification of electroencephalograms recorded from renal patients," *Computers and Biomed. Research*, vol. 8, pp. 461–473, 1975.
[4] J. R. Bourne *et al.*, "Analysis of EEG's for the national cooperative dialysis study," in *Proc. 31st Annual Conf. Eng. Med. Bio.*, 1978.
[5] P. S. Bowling and J. R. Bourne, "Discriminant analysis of electroencephalograms recorded from renal patients," *IEEE Trans. on Biomed. Eng.*, vol. BME-25, pp. 12–17, Jan. 1978.
[6] R. W. Coger *et al.*, "Factor analytic relations among EEG variables in an alcoholic population," in *Proc. 15th Annual San Diego Biomed. Symp.*, vol. 14, pp. 67–70, 1975.
[7] K. S. Fu, *Syntactic Methods in Pattern Recognition*. New York: Academic, 1974.
[8] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. New York: Academic, 1972.
[9] A. S. Gevins *et al.*, "Automated analysis electrical activity of the human brain (EEG): A progress report," *Proc. of IEEE*, vol. 63, pp. 1382–1399, Oct. 1975.
[10] R. C. Gonzalez and M. G. Thomason, *Syntactic Pattern Recognition*. Reading, MA: Addison-Wesley, 1978.
[11] G. D. Held *et al.*, "Ingres—A relational data base management system," in *Proc. 1975 National Computer Conf.* New York: AFIPS Press, 1975.
[12] B. Hjorth, "EEG analysis based on time domain properties," *Electroenceph. Clin. Neurophysio.*, vol. 29, pp. 306–310, 1970.
[13] S. L. Horowitz, "A syntactic algorithm for peak detection in waveforms with applications to cardiography," *Communications of ACM*, vol. 18, no. 5, pp. 281–285, May 1975.
[14] L. E. Larsen and D. O. Walter, "On automatic methods of sleep staging by EEG spectrum," *Electroenceph. Clin. Neurophysiol.*, vol. 29, pp. 516–520, 1970.
[15] A. V. Pollak, *Syntactic Analysis of the EEG*. Paris: Biosigma, 1978.
[16] A. Rechtschaffen and A. Kales, Eds., *A Manual of Standardized Terminology, Techniques and Scoring System for Sleep States of Human Subjects*. Washington, D.C.: U.S. Government Printing Office, Public Health Service, 1968.
[17] D. M. Ritchie and K. Thompson, "The unix time sharing system," *Bell Syst. Tech. J.*, vol. 57, pp. 1905–1930, 1978.
[18] J. T. Tou and R. C. Gonzalez, *Pattern Recognition Principles*. Reading, MA: Addison-Wesley, 1974.
[19] G. Stockman *et al.*, "Structural pattern recognition of cartoid pulse waves using a general waveform parsing system," *Communications of the ACM*, vol. 19, no. 12, Dec. 1976.
[20] T. P. Yunck, "Automatic classification of the EEG," unpublished dissertation, Yale Univ., 1977.
[21] T. P. Yunck and F. B. Tuteur, "Comparison of decision rules for automatic EEG classification," submitted to *IEEE Trans. Bio. Eng.*, 1978.

Correspondence

Machine Recognition of Devanagari Script

R. M. K. SINHA AND H. N. MAHABALA

Abstract—**Unlike simple juxtaposition of characters in Roman script, a word in Devanagari script is normally a composition of several characters, half forms, upper and lower vowel-modifier symbols, and diacritical marks; a classificatory approach is inadequate to deal with patterns of this complexity. A syntactic pattern analysis system with an embedded picture language has been designed for the purpose. The system stores structural descriptions for each symbol of the script in terms of primitives and their relationships. The input word is digitized, cleaned, thinned, segmented (to extract composite characters), and labeled (a local feature extraction process). Recognition involves a search for primitives on the labeled pattern based on the stored description. Contextual constraints are also utilized to arrive at the correct interpretation.**

I. Introduction

Devanagari script (script for Sanskrit, Hindi,[1] Marathi, and Nepali languages) is a moderately complex pattern. Unlike simple juxtaposition in Roman script, a word in Devanagari script is composed of composite characters joined by a horizontal line at the top. The basic alphabet of Devanagari consists of 13 to 16 vowels, 34 to 39 consonants, and 3 to 5 diacritical marks totaling 50 to 60 basic symbols. In addition, corresponding to most of the vowels, it has vowel-modifier symbols (called "matra" in Devanagari) which are attached to consonants; similarly for most of the consonants there are "half forms" or other derived forms which are used in making conjuncts. Thus the total number of symbols in Devanagari varies from 90 to 100. Composite characters (which sometimes become syllables) are formed by a valid graphical composition of consonants, their half forms, diacritical marks, and vowel modifiers and their number vary from 300 to a few thousand depending upon the quality in production and purity in script desired.

As an example, let us consider the composition of the Devanagari word NISHKRIYA (निष्क्रिय). It has two composite characters NI (नि) and SHKRI (ष्क्रि) and a simple character YA (य). The composite character NI (नि) is made up of a consonant NA (न) and a vowel-modifier symbol corresponding to the vowel E (इ) which is graphically represented as ि and the rule for composition is that the vowel modifier E is attached before the consonant. The composite character SHKRI (ष्क्रि) is more complex in composition. It consists of the half form SHA (ष) which is derived from its full form by deleting the vertical bar, a further composition with consonants KA (क) and RA (र) leading to a graphical representation of KRA (क्र) and finally with the combination of the two, the vowel modifier E is attached resulting in the composite character SHKRI (ष्क्रि). The composite characters NI (नि) and SHKRI (ष्क्रि) and character YA (य) are joined together by the top horizontal line to give the word NISHKRIYA (निष्क्रिय) in

Manuscript received July 31, 1978; revised April 16, 1979.

R. M. K. Sinha is with the Computer Centre, Indian Institute of Technology, Kanpur 208016 India.
H. N. Mahabala is a visiting consultant to Tata Consultancy Services, Nariman Point, Bombay 400021 from the Computer Centre, Indian Institute of Technology, Madras, India.

[1] Hindi is the official language of India.

0018-9472/79/0800-0435$00.75

Devanagari. As a result of composition of meaningful symbols of the script, natural breaks occur in composite characters. Thus Devanagari script recognition consists of three phases: segmentation, i.e., segmenting a word in Devanagari script into composite characters; decomposition, i.e., decomposing the composite character into meaningful parts; and recognition, i.e., recognizing the decomposed parts of the composite characters.

The classificatory approach to pattern recognition is not at all suited to deal with complex patterns like Devanagari script. On the other hand, out of the linguistic approaches most of the existing models for the description and analysis of patterns [1]–[5] had many shortcomings for the purpose. A new descriptive schema for generation and interpretation of patterns made up of curvilinear line-like elements has been developed and implemented [6]. The implementation called syntactic pattern analysis system (SPAS) has an embedded picture language called PLANG (a picture language for a class of pictures). The system SPAS has been tested for recognition of Devanagari script. In this correspondence the essential blocks of the recognition schema employed for recognition of Devanagari script are outlined. As PLANG forms the heart of the system SPAS, its general framework is outlined in the following section.

II. Outline of the Picture Language PLANG[2]

The picture language PLANG concerns two-dimensional pictures made up of line-like curvilinear elements. It assumes a rectangular picture frame for every picture. It consists of the following components: a) *primitives*, which are atomic pictures for the picture class; b) *partitioning functions*, which partition the picture frame to extract regions; c) *compose macros*, which define subpatterns which form part of many main patterns of the picture class, in terms of primitives, other compose macros, and an unassigned region; d) *frame functions*, which operate upon picture frame(s) to transform it (them) as per specifications. A sentence in PLANG describes a pattern in terms of primitives and compose macros specifying their relationships with respect to various regions of the picture frame or in terms of subpatterns operated upon by a frame function(s).

The PLANG description of a picture starts with a label denoting the picture name, followed by the word "Begin" indicating the beginning of the body of description. A set of PLANG sentences then describe the composition of the picture. Then the composition of all the compose macros used in the PLANG sentences are defined. The word "End" marks the end of the body of description. The form of a typical description is given below.

```
Picture name    Begin
Describe picture  (PLANG sentence)
Compose         (Macro name (dummy variable))
                (PLANG sentence using dummy variable
                 as the total frame)
Ω               (Frame dimensions)
                End                                  (1)
```

We shall explain the PLANG structure by means of an example from Devanagari script. The "partitioning function" is assumed to be a function which partitions the picture frame into nine equal regions as shown in Fig. 1. Every partitioned region could be further partitioned to get finer "regions" by repeatedly applying the partitioning function on a partitioned region. The union of regions could be defined by using U operator. The operator U combines the effect of its two operand partitioning functions in such a manner that the resulting "region" dimensions are just sufficient to enclose the regions obtained by applying partitioning functions individually. These are depicted in Fig. 2. Two frame functions named "superimpose" (denoted by an asterisk) and "append" (denoted by a dot) have been used for this class of pictures. The asterisk superimposes the operand picture frames of equal dimensions over each other. The attribute list for frame function (*) is always "nil" as it does not need any qualifier. The frame function "append" appends the two operand picture frames in such a manner that the center of the second picture frame is at coordinates given by frame function attribute list with respect to the center of first picture frame. If b_1, h_1 are breadth and height, respectively, for the first frame, b_2, h_2 are for the second frame on which the frame function "append" operates, and $(t_1\ t_2\ t_3\ t_4)$ $(m_1\ m_2\ m_3\ m_4)$ form the frame-function attribute list, the x, y coordinates of the center of the second frame with respect to the center of the first frame is given by

$$x = t_1 b_1 + t_2 h_1 + t_3 b_2 + t_4 h_2$$
$$y = m_1 b_1 + m_2 h_1 + m_3 b_3 + m_4 h_2. \qquad (2)$$

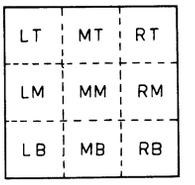


Fig. 1. Partitioning function.

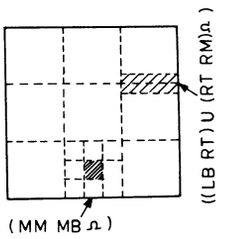


Fig. 2. Operator U and repartitioning.

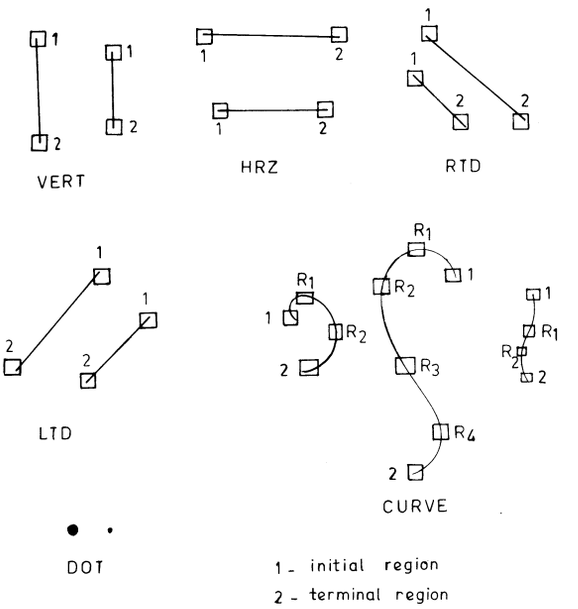
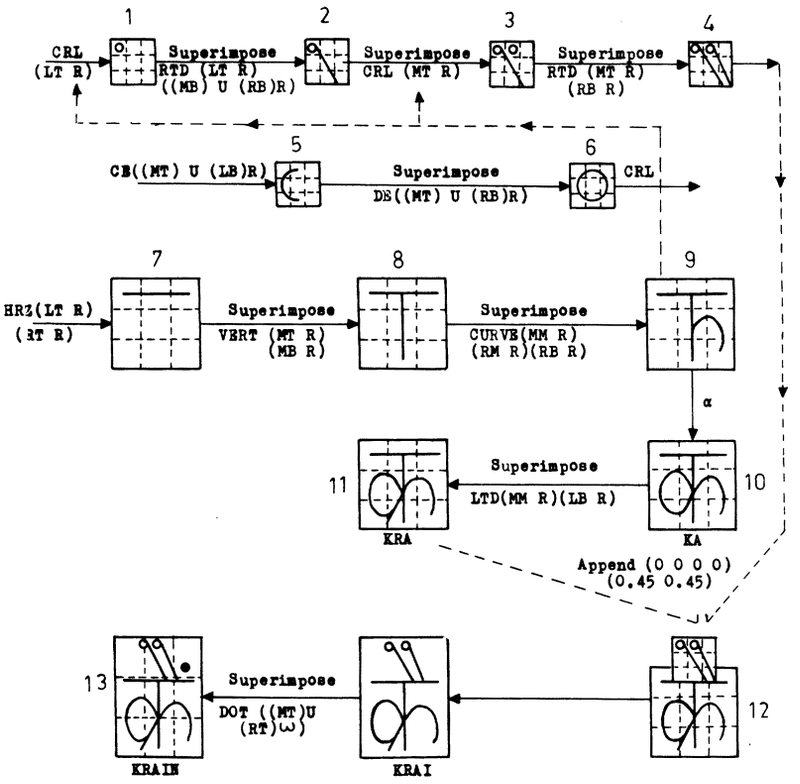


Fig. 3. Primitives.

[2] A more detailed discussion is to appear in the paper entitled "PLANG—a picture language for a class of pictures" in the journal *Pattern Recognition*.



α : Superimpose CURVE(MM R)((MT LM) U (RT LM)R)((LM LM) U (MM LB)R)((MM LB) U (MT MB)R) (MM R)

Fig. 4. Composition of Devanagari composite character KRAIN.

The resulting picture frame out of the "append" operation is the "extended union" of the operand frames (minimum sized picture-frame enclosing the operand frames). HRZ (horizontal line), VERT (vertical line), LTD (left-going diagonal), RTD (right-going diagonal), CURVE (arbitrary curve passing through specified regions in that order), and DOT (a dot) are primitives of the picture class. These are graphically illustrated in Fig. 3.

Utilizing the above mentioned partitioning functions, frame functions, and primitives, the PLANG description of a composite character called KRAIN (क्र) of Devanagari script is given in (3) as an example. Fig. 4 depicts the corresponding steps of composition. The frames 1 to 4 in Fig. 4 show composition for compose macro AI (vowel-modifier symbol), frames 5 to 6 for CRL, frames 7 to 11 for KRA and finally frame 13 contains the composite character KRAIN. In (3) Ω represents the total frame and CRL (a circle-like curve), CE (C-shaped curve), DE (D-shaped curve), KA and KRA are compose macros defined with respect to an unassigned region *R*. The region *R* assumes a value as per the use of the compose macro in the description.

Thus all compose macros once defined could be treated as primitives for all practical purposes in the description of the total picture. The compose macros are analogous to subroutines or macros in a computer program. A subroutine, when supplied with arguments and called, returns desired computation. In a similar fashion, a compose macro, when supplied with a frame, returns desired composition over the frame.

It is evident that a compose macro needs only one region for its specification whereas a primitive may need a list of regions. Thus a picture "frame" consists of a primitive followed by a list of regions or a compose macro followed by a region. Although primitives used in the description (3) do not have attribute lists in general, a primitive may have an associated attribute list which gives a description of some of its features. The attributes of a primitive might be in the form of its spread, thickness, color, etc. As far as possible attributes are assigned values relative to picture frame dimensions. Some of the attributes like color will be independent of frame dimensions and will be in absolute terms.

It is also evident that a sentence in PLANG is represented as

```
DEVKRAIN    Begin
Describe KRAIN   (*((KRAI)(DOT((MT) U (RT)Ω))))
Describe KRAI    ((.(0 0 0 0) (0 .45 0 .45))((KRA ((LM) U (RB)Ω))(AI ((LT) U (RT)Ω))))
Compose     ((KRA(R))(*((KA(R))(LTD(MM R) (LB R)))))
Compose     ((KA (R)) (*((HRZ(LT R)(RT R)) (VERT(MT R)(MB R))(CURVE (MM R)(RM R) (RB R)) (CURVE
            ((MT MM) U (LB MM)R) ((MT LM) U (RT LM)R) ((LM LM) U (MM LB)R) (MM LB)
            U (MT MB)R) ((MT MM) U (LB MM)R)))))
Compose     ((AI(R))(*((CRL (LT R))(RTD (LT R)((MB) U (RB)R))(CRL (MT R))(RTD (MT R) (RB R)))))
Compose     ((CRL (R))(*((CE((MT) U (LB) R))(DE(MT) U (RB)R)))))
Compose     ((CE(R))(CURVE (RT R) ((MT) U (RB LT)R) ((LM) U (MM)R)((MB) U (RT LB)R)(RB R)))
Compose     ((DE(R))(CURVE (LT R)((MT) U (LB RT) R) ((RM) U (MM) R)((MB) U (LT RB)R)(LB R)))
Ω           (coordinates of lower left corner of picture-frame height width)
            End
```

(3)

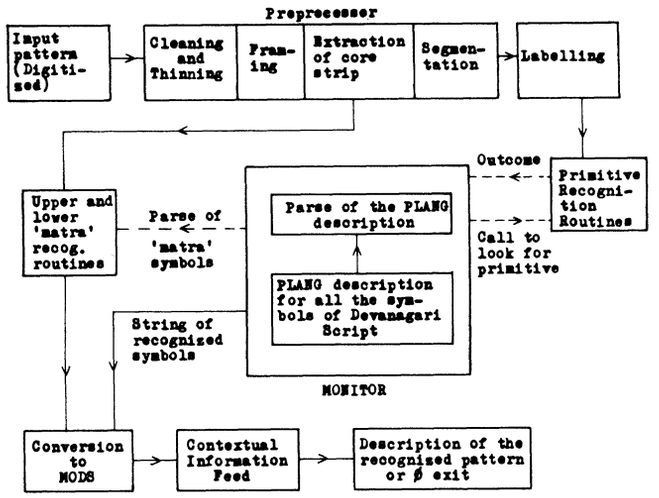


Fig. 5. Schematic block diagram for Devanagari script recognition.

the outcome of a "frame-function" operating upon a list of "frames" or another "PLANG-sentence". Two or more picture frames may be combined in a desired manner to make a larger picture frame or an original picture frame could be transformed to a new picture frame in a desired manner utilizing a set of "frame-functions."

III. System Design Detail

Fig. 5 shows block schematic diagram for Devanagari script recognition.

The monitor of the system stores PLANG descriptions for all the symbols of the script. The system monitor assumes that the script has been decomposed into a set of basic symbols before presentation. Although it is possible to construct PLANG descriptions for composite characters (as in 3) or complete words of the script, this would necessitate storage of PLANG descriptions for all possible composite characters or words and devising algorithms for dealing with complex frame functions used in construction of the descriptions.

The system assumes only one type of frame function named "superimpose" and is denoted by an asterisk. The partitioning functions and primitives used are the same as outlined in Section II earlier. PLANG descriptions for some of the symbols of Devanagari script are given in (4).

```
DEVSYM    Begin
Describe A(अ)  (*((DE (LT Ω)(LM Ω))(DE(LM Ω)(LB Ω)) (HRZ (LM Ω)(MM Ω)) (TE((MT) U (LP) Ω))))
Describe E(इ)  (*((HRZ (LT Ω)(RT Ω))(VERT(MT Ω)(MM Ω)) (YES ((LM) U (RB) Ω))(VAE ((LM) U (MB) Ω))))
               and so on
               (PLANG sentences for all other symbols of Devanagari script)
Compose   ((TE (R))(*((HRZ (LT R)(RT R)) (VERT (MT R)(MB R)))))
Compose   ((YES (R))(*((CE (RT R)(MM R))(DE (MM R) (LB R)))))
Compose   ((VAE (R)) (*((RTD (LT R)(RB R)) (CRL (LT R)))))
               and so on
               (definitions of all other compose macros used in the description of symbols of Devanagari script)
               End                                                                          (4)
```

Functions of different blocks of the system are outlined below.

a) Preprocessing: The digitized input pattern is presented to the preprocessor. Preprocessing consists of five operations: cleaning, thinning, framing, finding the core strip, and segmenting the input frame.

The cleaning operation removes isolated points which carry no information. The operation of thinning extracts a skeleton of the pattern. The thinning algorithm used consists of step by step stripping off the boundary points without loss of connectivity. This process of stripping off the boundary points is continued until the thickness of the pattern reduces to one or two points.

The operation of framing evaluates the rectangular boundary dimensions of the actual pattern by scanning from all directions.

The "core strip" in Devanagari script is defined as the strip containing the flow of the main characters, that is, the strip obtained after deleting lower and upper modifier symbols and diacritical marks. The upper and lower modifier symbols and diacritical marks are separately framed and recognized. In a typical Devanagari word these upper and lower signs do not occur on all characters and a horizontal line runs at the top of the core strip. Thus it is easy to extract the core strip by counting black point densities of the lines along horizontal direction. It will indicate a sharp peak and a sharp fall at the two ends of the core strip. For example, the core strip for the word पहाड़ is पहाड़.

The core strip thus obtained is segmented into composite characters or constituent symbols as far as possible. Since a word is formed by joining composite characters through a top horizontal line, segmentation is a straightforward process if the constituent composite characters are nontouching. First of all, a preliminary segmentation is attempted by counting black point densities of the lines along vertical direction from below the top horizontal line of the core strip to the bottom of the strip. Partitioning is done in the middle of the area where black point density is zero. For example, the core strip पहाड़ after segmentation would give rise to segments प, ह, ा, and ड़. Such a segmentation procedure fails if the constituent characters are under the shadow of each other as is encountered in situations like प्र where the half character ् is in the shadow of character र. Thus, after the preliminary segmentation, if for any of the frames containing the segments, the breadth, and height ratio exceeds 1.7, a complex segmentation is attempted. A curve drawn through a continuity of white points from below the top horizontal line of the core strip to the bottom of the strip is used for complex segmentation. If this procedure fails to get further segments, it is concluded that the constituent characters are touching. For the touching characters, approximate segmentation could be obtained through aspect ratio information. However, it has not been tried in the present implementation. It should be noted that PLANG depicts relationships among picture fragments in terms of "regions" and it has inherent capability of providing tolerance for inappropriate segmentation.

b) Labeling: Labeling is a local feature extraction operation. Every point of the pattern is assigned a label depending upon the local property it exhibits with respect to the neighboring points. Primitive recognition algorithms primarily rely on the labels.

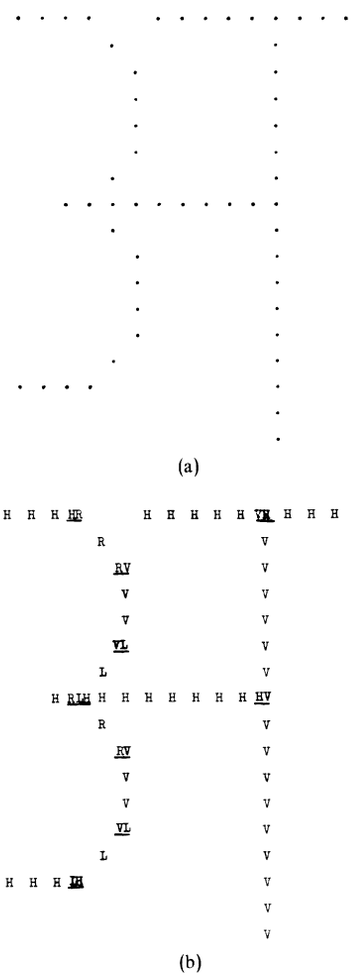


Fig. 6. (a) Thinned pattern. (b) Labeled pattern.

The system SPAS looks for five kinds of local properties, namely: whether the point in question forms the part of a vertical line (V label), a horizontal line (H label), a right-going diagonal (R label), a left-going diagonal (L label) or none of these (do not care label–D label) at the local level. The local spread in which these properties are looked for, is specified in terms of the frame dimensions. A pattern point may be assigned more than one label if it satisfies more than one property. Thus a point having label (VL) denotes that a vertical line and a left-diagonal at the local level meet at that point denoting a junction of the type (VL).

Fig. 6 shows an example. A labeling threshold of four points is assumed for hand V labels; and three points for R and L labels.

c) Primitive Recognition: All primitives of the picture class are described in terms of a spatial distribution of H, V, R and L labels in the regions of their spread. Do not care labels are not used in the description. D labels could be interpreted as V, H, R, or L labels as and when needed.

For example the primitive VERT is described as "continuity" of V labels from the initial region to the terminal region. (Initial and terminal regions define the spread of the primitive.) One may allow a certain amount of break in the continuity depending upon the acceptable deformation in the primitive. The flowchart of Fig. 7 explains the algorithm for recognition of the primitive VERT. The algorithms for the other primitives have been devised in a similar fashion.

d) Call to Look for Primitive: A call to look for a primitive in a specified region of the frame is made to the primitive recog-

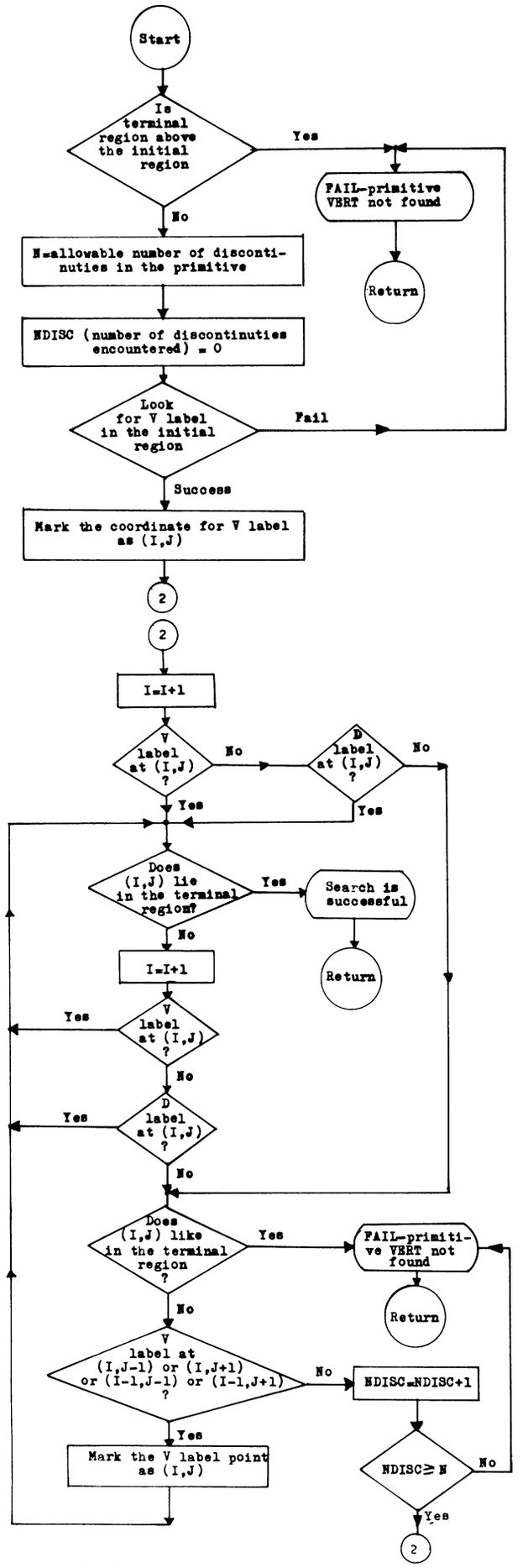


Fig. 7. Flowchart for recognition of primitive VERT.

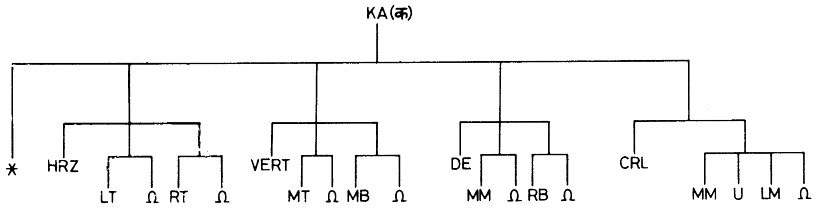


Fig. 8. Parsing tree for symbol "KA".

nition routine on the basis of the parse of the stored descriptions. The system stores PLANG descriptions for all symbols of the script. In Fig. 8 a hierarchic representation of the PLANG description for the symbol KA (क) is shown. A goal oriented top-down parser looks for a frame function and finds an asterisk which signifies that all subtrees having the same root as the asterisk, represent picture fragments which are superimposed. A call to look for primitive HRZ in the "left-top" to "right-top" region of the picture frame is made to the primitive recognition block. If it is successful, the parser moves on to other subtrees towards the right. If at any stage the goal is unsuccessful, penalty is entered and if the penalty exceeds a preset threshold value, the parser picks up the description for the next probable symbol. A parse is said to be successful if all the goals while spanning the tree from left to right are successful.

e) Feeding Contextual Information: The system is equipped with contextual information about Devanagari script to aid in correct interpretation. Some of the diacritical marks or vowel-modifier symbols are never attached to some of the characters of the script, and some of the symbol sequences are illegal for the script. Rules depicting these restrictions form major components of the contextual information.

These rules, in addition to pointing out anomalies, might give alternatives which, in turn, might force re-entry into the system. At this stage some heuristics might be used to make a guess among the alternatives. For example, if the interpretive device gives the recognized symbol(s) as क्अ (a composite character in Devanagari script), one can be sure that the system has not been able to recognize the symbol(s) correctly as we know that no vowel can follow a half letter in Devanagari script (क् is a half letter and अ a vowel in Devanagari). Now there are two alternatives, either the recognized half letter is an actual full letter or the recognized vowel is actually a consonant. If we take the nearest symbol for each of the symbols taken to be wrongly interpreted (based on heuristics), more likely, in the former case the recognized symbol(s) would be कअ, and in the latter क्स (क is a full letter and स a consonant in Devanagari). The routine might re-enter at an appropriate place to find whether the half letter क् is actually a क or the letter अ is an actual स. In another example, suppose the recognized symbol is एा, i.e., a vowel (ए) followed by a vowel modifier (ा). This is not permitted in Devanagari script. The possible resembling symbols as alternatives to एा are स or रा. Based on heuristics the system might look for the presence or absence of the middle horizontal line.

The interpreted symbol string, prior to its entry into the contextual information feed block, is converted to machine oriented Devanagari script (MODS) symbol string [7]. This facilitates further processing.

IV. Results and Discussions

The recognition system has been implemented on an IBM 7044 using Fortran and IITPAX [6]. The implementation is aimed at demonstrating a methodology for recognition of Devanagari script and testing viability of the approach. Only a small subset consisting of resembling characters and upper and lower signs were included to form the samples. The system has been tested over a small sample size and no attempt has been made to make it a commercial product.

The chosen test symbols are: अ, उ, ए, र, रा, शा, य, स, त, र, श, [illegible], and [illegible]. This is approximately 50 percent of the total set of Devanagari script symbols. The criteria for choice has been to include a few basic characters with their derived forms and all possible modifier symbols. The enlarged contours of some of the actual symbols used are shown in Fig. 9.

The experiment was carried out in two parts. In the first part individually typed as well as constrained handwritten characters of varying size, thickness, and shape were used as test patterns. In the second part only typed composite characters or words of the script, formed out of the chosen test symbols, have been used as test patterns. The results obtained have been depicted in Tables I and II, respectively. As only typed script has been used in the second part, the system's ability to perform segmentation and extraction of core strip has been examined only with respect to the typed script.

A mechanical method of scanning the input pattern was employed for its digitization due to lack of facilities for automatic scanning. The input patterns were blown up 7 to 10 times bigger than their normal size and superimposed on a millimeter grid. A millimeter square is taken to represent a black point if it is black for more than 50 percent of its area.

The PLANG descriptions for the symbols used in the test were generated by mere consideration of their shapes and variability. As an example, description for character उ would be corresponding to two *D*-shaped curves placed one above another with a top horizontal bar. These descriptions are not modified once planted into the system. In the first part of the experiment, different PLANG descriptions were needed for typed and handwritten characters, and they were stored as logical OR in the system. No training set has been employed in arriving at a suitable PLANG description or for its improvement. The description is close to the natural description which one could possibly use in describing the shape of a character say to a child.

As evident from the results shown in Tables I and II, more than 90 percent correct recognition has been achieved. Only two characters have been misinterpreted because of their high resemblance. "Unable to recognize" exits have been primarily due to the mixing up of upper symbols and the failure of primitive recognition routines on heavy distortions. Contextual constraints have been found to be extremely useful in interpreting "matra" symbols.

Although the experiment has been conducted over a small sample size, the results obtained are quite encouraging. It demonstrates viability of an approach for recognition and analysis of adequately complex patterns made up of line-like elements. Besides using a simple descriptive schema, for the first time contextual constraints have been employed explicitly to resolve ambiguities and help in predictive parsing. The recognition scheme seems to be applicable to almost all Indian scripts. Besides

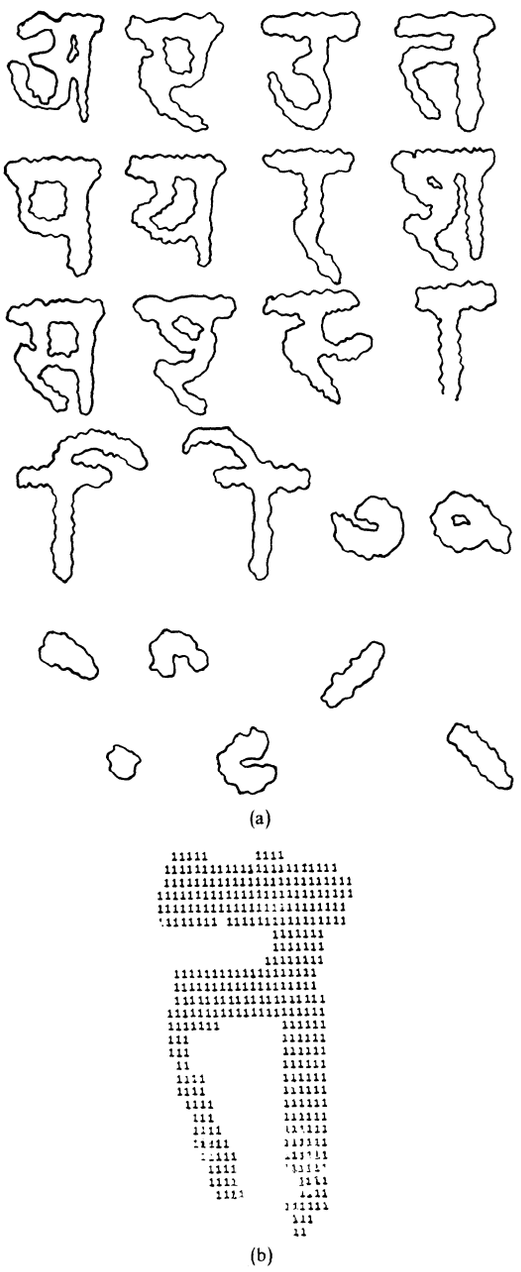


Fig. 9. (a) Some samples of symbol contours. (b) Digitized character.

TABLE I
TEST RESULTS FOR SOME INDIVIDUAL CHARACTERS

Test Symbol	Sample size (typed+handwritten)	Correct recognition
अ	7 (4+3)	5 (3+2)
उ	7 (4+3)	7 (4+3)
ए	7 (4+3)	7 (4+3)
र	7 (4+3)	7 (4+3)
ष	4 (2+2)	3 (1+2)
ध्र	6 (3+3)	5 (2+3)
य	3 (1+2)	2 (1+1)
प	4 (2+2)	4 (2+2)
त	4 (2+2)	4 (2+2)

TABLE II
TEST RESULTS FOR SOME COMPOSITE CHARACTERS OR WORDS

Test Sample	No. of Symbols	Recognition status
ओस	4	Correct
ओर्रे	6	Unable to recognize 'matra' symbol
श्रेश	4	Correct
उसे	3	Correct
सर्प	3	Correct
शेप	4	Correct
प्रति	4	Correct
प्रीत	4	Correct
स्ति	3	Correct
त्यो	4	Correct
स्यु	3	Correct
तृयां	5	Correct
सूर्य	4	Correct
श्पि	3	Correct
सैर	3	Unable to recognize upper 'matra' symbol
शोर	5	Correct
प्सूत	4	Correct
शैंपू	6	Unable to recognize upper 'matra' symbol

recognition, the system's inherent capability of analyzing input patterns of the class and utilizing contextual constraints, interesting applications like computer assisted teaching of scripts can emerge [8].

However, there are many aspects which need further exploration, the most important being the machine generation of PLANG description. This would enable the system to "learn" about a pattern during the training phase and improve its performance. Most of the descriptive approaches for pattern recognition, have ignored or not utilized this aspect of the problem primarily because it is difficult to devise criteria for measuring similarities or dissimilarities among descriptions. Winston's work [9] is of relevance in this context.

REFERENCES

[1] R. Narasimhan, "Picture languages," Computer Group, TIFR, Bombay, Tech. Rept., no. 75, 1969.

[2] R. Narasimhan and V. S. N. Reddy, "A syntax-aided recognition scheme for hand-printed English letters," *Pattern Recognition*, Nov. 1971.

[3] A. C. Shaw, "The formal description and parsing of pictures," Stanford Univ., Tech. Rept., no. CS84, 1968.

[4] K. S. Fu and P. H. Swain, "On syntactic pattern recognition," *Software Engineering*, J. T. Tou, Ed. New York: Academic, 1971.

[5] A. Guzman, "Some aspects of pattern recognition by computer," M.S. thesis, Dep. of Elec. Eng., M.I.T., Cambridge, 1967.

[6] R. M. K. Sinha, "A syntactic pattern analysis system and its application to Devanagari script recognition," Ph.D. dissertation, Elec. Eng. Dep., I.I.T., Kanpur, 1973.

[7] R. M. K. Sinha and H. N. Mahabala, "MODS—machine oriented Devanagari script," *J. Inst. Telecom. Eng.*, India, pp. 623–628, Nov. 1973.

[8] R. M. K. Sinha, "Teaching script on a digital computer," *J. Inst. Telecom. Eng.*, India, pp. 720–722, Nov. 1976.

[9] P. H. Winston, "Learning structural description from examples," M.I.T., Cambridge, Tech. Rept. AI TR-231, 1970.