A Multi-Level Perception Approach to Reading Cursive Script[1]

Sargur N. Srihari and Radmilo M. Bozinovic

Department of Computer Science
State University of New York at Buffalo

ABSTRACT

Reading cursive script involves elements of visual perception at one level of processing and those of language perception and understanding at a higher level. The problem is approached as one in which a word image is transformed through a representational hierarchy of levels based on descriptions that use points, contours, features, letters, and words. Global control is hierarchical until an intermediate level after which it is heterarchical. Two modes of learning are defined: supervised training with user feedback and unsupervised adaptation to the writer.

I. INTRODUCTION

Reading cursive script is the problem of transforming text from the iconic form of cursive (continuous or running) writing into symbolic representation. There exist several early approaches to the problem (Sayre 1973). The goal of this work was to develop a modern framework for reading a well-formed, prerecorded binary valued word image which is assumed to represent a word in a given lexicon. The problem is approached as of signal to symbol transformation between a series of representation levels: points, contours, shapes, letters, and words. The approach here lies in between two different methods of using lexical context in word recognition: one is segmentation followed by classification of segments using a lexicon to constrain alternatives (Srihari 1983) and the other is to classify words based on global word shape without any segmentation (Hull 1986).

II. SYSTEM OPERATION

The system processes the input data hierarchically until a certain level, and the processing is heterarchical (interaction between levels) after that (Erman and Lesser 1980) (Figure 1).

The raw image, after initial preprocessing of smoothing and slant removal is brought to the *I(mage)-level.* Slant removal and smoothing operations eliminate any significant overall slant of the strokes, and remove minor contour discontinuities and roughness, bringing input to the *I*-level. An input word, *might*, is depicted in it's *I*-level form in Figure 2. The three main vertical zones of script — the middle one, where the "bodies" of all letters reside, and upper and lower ones, corresponding to ascenders and descenders are found by a reference-line-finding operation.

A. Presegmentation

The next preprocessing step is to do loose segmentation, i.e., identify each *potential* presegmentation point (PSP). The task is to find as small a PSP set over all words as possible, such that a subset SP is the set of true segmentation points between letters. A candidate

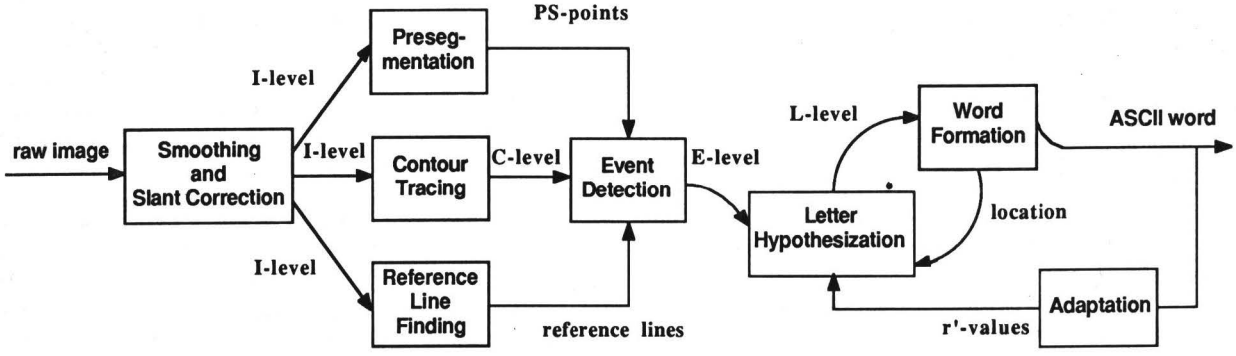

Figure 1. Cursive Script Recognition System

[1] This work was supported in part by the Office of Advanced Technology of the United States Postal Service under contract BOA 104230-84-0962 and the National Science Foundation Grant IRIS-86-13361.

PSP may or may not be the actual PSP, e.g., the partial inscription corresponding to the first letter *might* could represent an *m*, *ui*, or *in* sequence, and we would like to have each of these potential letter borders marked as a PSP. This operation results in the first letter, *m*, spanning three presegments (0-4), *h* spanning two presegments (6-8) and all the others one each (Figure 2). The fourth presegment (3-4) corresponds to no letter, and will be interpreted as such and indicated with a quote (') symbol.

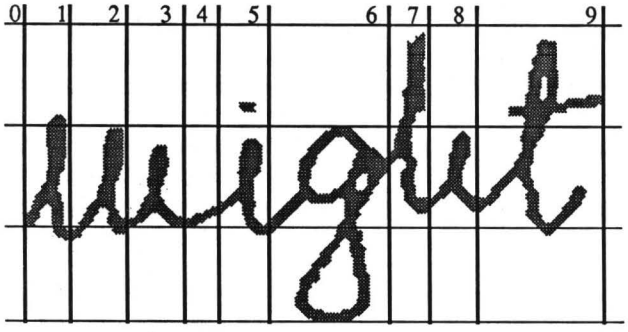

Figure 2. Example of word *might* with reference lines and PSP.

B. Contour Tracing

Preliminary processing yields reference lines, PSPs, and a smoothed and slant-corrected image at the *I*-level. The next task is to transform this to the *C(ontour)-level* which describes the image in terms of contours and their topology. The *C*-level representation of *might* is:

```
( ( (19 59) (0 0 2 1 ..) (( (96 83) (1 0 0 -1 ..) )
  ( (96 85) (1 0 0 -1 ..)) ( (148 101) ( 1 -1 -2 -2 ..) )
  ( (98 56) ( 1 -1 -2 -2 ..) ))
( (73 92) (- 0 0 2 ..) )).
```

There are six contours present: the main outside contour, followed by four inside contours ("holes") within it (three in *g* and one in *t*), and another outside contour (the dot over *i*).

C. Event Detection

The *C*-level is then transformed into the *E(vent)-level* which is a description in terms of features, or *events*, and their locations (Bouma 1971). This description of *might* records that there exists an *i*-shape (peak) between PSP 0-1, a lower curve (1-2), a dot (4-5), upper curve (5-6), etc. The actual representation is a binary matrix of events and locations. The columns represent sixteen events, and rows represent their location in terms of presegments — either single ones or sequences of two. The *L(etter)-level* consists of a series of alternative letter strings, or prefixes, that account for parts of the original *I*-level. The top level output is an ASCII word formed from a lexicon of legal possibilities, and such that it corresponds to the entire *I*-level representation.

D. Letter Hypothesization

Until the *E*-level, successively more refined *unique* descriptions of the *whole* word image are generated. At the *L*-level there are multiple representations competing with each other, which are not all generated at once. Competing letter hypotheses can account for exactly the same portion of the image, in which case there is only a classification difference between them; or, they can partially overlap thereby displaying segmentation differences. After a series of strict hierarchical transformations, letter hypothesization is the first global module that interacts with higher ones in a heterarchical manner, and as such, when called upon might not produce a set of full word representations on the *L*-level; rather, it generates partial representations of the size and at the positions that are required by higher-level modules.

Letter hypothesization consists of two parts: *letter selection* and *rating computation*. Letter selection is an operation in which certain letter hypotheses are generated based on partial *E*-level data. In rating computation all of the *E*-level evidence is taken into account while generating an overall likelihood score for each hypothesis. For each selected letter l we compute the *a posteriori* probability $P(l|\bar{e})$, where $\bar{e}$ is the vector of all (abstract) events which have occurred.

Knowledge about letter formation is specified in three places: rules of presegmentation, which are procedurally fixed; tables of r'-values, one per context (total of nine), each one 27 × 16 (number of events) in size, each element of which is a real number measuring the dependency between a letter and event in that context; and rules for event compatibility, which are built into the event detection process. The letter hypothesization scheme has both statistical and syntactic elements. Within each PS, letters (or parts) are matched statistically. The syntactic process is reflected by the varying number of PS that different letters can span, since separate statistics are compiled for each of the different contexts.

Once events are detected, a search procedure examines the *E*-level matrix and proceeds in parallel through a lexicon represented as a trie (Srihari 1983). In doing so it calls the letter hypothesization module. Given a PSP, the latter generates an *L*-level description, one letter long, of the portion of the word from that point to at most three points to its right. Thus, starting from the left end of the word, or point 0, it generates the sequence of letter hypotheses. Thus *might* is

represented by the list:

((m .84 3) (n .73 2) (u .73 2) (w .69 3)
(w .51 1) (c .30 1) (j .30 1) (s .25 1)
(e .21 1) (g .16 1) (v .10 2) ...)

Each entry contains the letter, its rating on a (−1, 1) scale, and the number of presegments it covers. Expansion of the best current hypothesis is done by making its rightmost presegmentation point the starting point for new letter hypothesizing.

Since the current best hypothesis is *m*, which covers three presegments, the next letter hypothesization starts from point three with the L-level description

(('.60 1)(u .46 2)(y .66 2)(v .73 2)(n .74 2)(h .82 2)
(k .83 2)(x .83 2)(x .83 2)(w .85 2) ...)

All these letter hypotheses are attached to the current top solution, *m*, from the right, and sent for lexicon lookup and rating recomputation, giving a list of prefix hypotheses. The representation here has the form (prefix, rating, PS length, letter PS length(s)). Actual hypotheses in iteration 2 for our example are

(('n .73 2 2) ('u .73 2 2) ('m' .72 4 (3 1)) ('w .60 3 3)
('w .51 2 2) ('c .30 1 1) ('j .30 1 1) ...)

Therefore, the top hypothesis has been replaced by its high-rated descendants, and the whole list reordered according to descending score values. In this case, one of them, *m'*, is the correct interpretation, but has fallen from first place, and will have to wait until it regains it to get expanded further. This happens as the hypotheses above it get expanded, and as the evidence supporting their extensions diminishes, they drop in ranking or disappear. By iteration 4, the correct candidate resumes top position, and retains it until iteration 6, as *m'ig* as seen below:

(('m'ig .77 6 (3 1 1 1)) ('un .71 4 (2 2))
('nu .71 4 (2 2))('w .60 3 3)('us .49 3 (2 1))
('ne .47 3 (2 1))...)

Letter *h* does not get a high rating, and *m'igh* drops again in iteration 7. Finally, by iteration 20, all the other alternatives have diminished in rating themselves, the correct interpretation is topmost once again, and in the next step it yields a hypothesis that is a legal lexicon word (i.e. not only a prefix), and also accounts for the whole word (9 presegments), thus allowing the algorithm to return it as the answer. The final rating for *might* is of the form:

('m'ight .48 9 (3 1 1 1 2 1)).

Two questions that need further explanation are: (1) How do certain combinations get preferred over others in a quest for a unique solution? This is done by way of direct comparison between these combinations — and preferably in a way that does not depend on their length or position. Given an E-level description, the goal of the recognition procedure is to search the trie and come up with a full word as an answer. The search strategy employed here is a modified stack decoding algorithm (Jelinek 1969). (2) When and where to ask for how many more letter hypotheses to be generated? This is solved by specifying an appropriate control mechanism. Word hypothesization takes place left-to-right. The computation of ratings for prefix hypotheses is based on two requirements: (i) length-uniformity of values, allowing for comparison on equal basis of hypotheses of different length, and (ii) length-independence of the computation, which is to be performed as a function of only the parent hypothesis' rating.

III. TRAINING AND ADAPTATION

Several parameters concerning the contour and feature levels are learned in an initial *training* phase. These parameters can be gradually altered with change of writers and styles in a process of *adaptation*.

In training, the user provides feedback to the system in order to facilitate the learning of r-values. Thus it is a case of supervised learning. For a given letter l and event e the estimation of these values is based on the formula:

$$r(l, e) = \frac{p(e|l)}{p(e)} = \frac{n(l, e) / n(l)}{n(e) / n} = \frac{n(l, e) \times n}{n(e) \times n(l)}$$

This training scheme, while requiring the training set to contain all letters that are to be recognized, and in sufficient quantities to make statistical judgements about them, the selection of words themselves can be arbitrary, and isolated letter prototypes are not necessary. Segmentation of a word into letters through presegmentation and feedback allows information from each letter to be gathered independently from each other.

In the case of adaptation the identity of the letters has to be decided without outside feedback. Such information is not present at the time of letter hypothesization, but it is available at the end of the recognition procedure in the form of the claimed word identity, the only remaining problem being the reconstruction of sets of relevant events for each occurred letter. Such problems of learning from solution paths are not uncommon (Sleeman 1982). Its p string is always retained, and is thus available also for the final word answer. This way, the increments for all the n-

counters can be precisely identified, and their values updated. An identical procedure is used in the training part, except that the word identity comes from a different source. In cases when a word is deemed unrecognizable, no counter updating takes place. When a new writer is installed, the counter values are reset at the original "unadapted" level, and subsequent adaptation for that writer takes place from there.

IV. PERFORMANCE

Two lexicons were used in the experiments: most experiments were based on a small lexicon of 710 words and the rest on a lexicon of 7,800 words (Kucera 1967). In the first experiment the training set consisted of 66 words, where the writing naturally conformed with the constraints of being horizontal and not slanted. The test set consisted of another 64 words. With the small lexicon 77% were correctly recognized (first choice), 9% incorrectly recognized, and 14% rejected. When the search procedure was extended to consider the first two returned words for the correct answer, the results were 81% correct, 5% error and 14% rejection. The second experiment involved learning. Latent learning (counter updating) took place during the first experiment, and upon its completion the acquired knowledge was invoked. After that, the same set of 64 words was run over again. In this case it is justified to run the *retraining set* as a test set, since it is combined with the original training set and in that sense not the only source of knowledge, and the feedback comes from inside. Thus no improvement is guaranteed in advance (e.g., *wrong* learning, in case of misrecognized words, could solidify the misrecognition of this input in the second pass, unless heavily counterbalanced by other evidence). The results: 78% correct, 8% incorrect, and 14% rejected. In the third experiment, a second writer and the small lexicon were used. No slant restrictions were placed on the writing since the slant removal module was used. A total of 53 word samples was taken, eight of which were used for *retraining*, and the remaining 45 as the testing set. The retraining was performed in the same way as the original training, except that the counters obtained from this pass were used to update (in an unweighted manner) those from experiment 1, thus just slightly modifying the original r'-values. The results: 64% correct answer in top two, 11% incorrect, and 24% rejected. The results were similar when a third writer was used with 67 word samples and 18 used for retraining.

V. SUMMARY

The main results of this enterprise of developing a cursive script word recognition system are: (i) a scheme of knowledge interaction, with the appropriate control and data structures, and clearly identified representation levels was defined; (ii) a method for hypothesizing and rating certain objects from their (possibly unrelated) constituent features was developed — it is uniform over all features, objects, their locations and sizes and is computationally simple; (iii) several image level operations were introduced/refined; and (iv) learning/adaptation capabilities for cursive script were distinguished.

REFERENCES

1 Bouma, H., "Visual recognition of isolated lower case letter," *Vision Research*, 11, 1971, 459-474.

2 Erman, L. and Lesser, V., "The Hearsay-II speech understanding system: a tutorial," in W. Lea (ed.), *Trends in Speech Recognition*, Prentice-Hall, 1980, 361-382.

3 Hull, J.J. and Srihari, S.N., "A computational approach to word recognition: hypothesis generation and testing," *Proc. of IEEE-CS Conf. Computer Vision and Pattern Recognition*, June 1986, 156-161.

4 Jelinek, F., "A fast sequential decoding algorithm using a stack," *IBM J. Res. Develop.*, vol.13, Nov. 1969, 675-685.

5 Kucera, H. and Francis, W.N., *Computational Analysis of Present Day American English*, Brown University Press, 1967.

6 Sayre, K.M., "Machine recognition of handwritten words," *Pattern Recognition J.*, 1973., 213-228

7 Sleeman, P., Langley, P. and Mitchell, T., "Learning from solution paths: an approach to the credit assignment problem," *AI Magazine*, 1982, 48-52.

8 Srihari, S.N., Hull, J.J., and Choudhari, R., "Integrating diverse knowledge sources in text recognition," *ACM Transactions on Office Information Systems*, 1(1), 1983, 58-87.