

SPEED, ACCURACY, AND FLEXIBILITY TRADE-OFFS IN ON-LINE CHARACTER RECOGNITION

C. C. TAPPERT

IBM Research Division

T. J. Watson Research Center, P. O. Box 704

Yorktown Heights, NY 10598, USA

An on-line character recognizer was improved by making trade-offs on computation speed, recognition accuracy, and flexibility. Firstly, the error rate was halved by increasing computation precision at the expense of speed, achieving a character recognition accuracy of 97.3 percent. Secondly, without loss of accuracy, computation speed was increased by an order of magnitude over the original speed, to 85 characters/second on the IBM System 370/Model 3081. This was done by using a fast linear match to narrow the character choices for a slow but accurate elastic (non-linear) match. Some loss in flexibility resulted, however, because the linear match requires the same number of handwritten strokes for input and prototype characters. Also, the error rate of elastic matching was found to be about half that of linear matching.

Keywords: On-line handwriting recognition; Real-time character recognition; Tablet digitizers.

1. INTRODUCTION

Ideally, a handwriting recognizer can be designed for maximum speed, accuracy, and flexibility. However, when used in a workstation, it may have to be designed to compromise among these properties. Firstly, the recognition algorithms must be fast enough to be usable when implemented on a microprocessor. This means that they must operate at writing speed, i.e. about two characters per second for handprinted alphanumeric characters. Secondly, the recognizer must be accurate enough to be acceptable to the user. And thirdly, the recognizer must be flexible, i.e. it must handle a reasonably wide range of handwriting input.

For the purpose of improving our on-line character recognition system for use in a prototype of a spreadsheet application, two sets of modifications were made to the system, resulting in improved recognition accuracy and computation speed. This paper reports on the evolution or development of the system through these modifications, particularly from the point of view of compromises on speed, accuracy, and flexibility. A complete description of the system's algorithms is also presented.

The original system was designed to operate on a general English alphabet of printed characters, including lower- as well as upper-case characters.¹ Also, to achieve a reasonable response time on an IBM PC/AT, computational shortcuts had been implemented at the expense of recognition accuracy. In order to determine its useability in a spreadsheet application this system was evaluated on an alphabet not including the lower-case letters. The evaluation, performed by a group independent of

the one that had designed and implemented the system, provided the data for the present studies.²

The system was first modified with the aim of maximizing recognition accuracy at the expense of computation speed. Therefore, modifications included removal of computational shortcuts and sacrificing alphabet generality. This system is described in the first experiment. It was also evaluated separately.²

The system was further modified to achieve greater speed of computation at the expense of some loss in flexibility, but with no loss of recognition accuracy. This was done by constraining the stroke number in matching and using stroke-by-stroke linear and elastic matching. This system is described in the second experiment.

2. EXPERIMENT 1: STROKE-NUMBER-FREE RECOGNITION

The purpose of this development phase was to improve the recognition accuracy of the system without regard for speed of computation. The reader is advised to read the study by Schoonard *et al.*,² which fully describes the data collection procedure and the recognition accuracy results. Here, the system algorithms are described in detail together with the modifications made to increase accuracy.

2.1. Procedure

Handprinted character data were collected by Schoonard *et al.*² The data collection procedure will be summarized briefly. Nine volunteers from our research laboratory served as participants. Each participant sat at a table containing an opaque digitizing tablet (a Penpad tablet manufactured by Pencept) connected to a PC/AT. This tablet had a resolution of 200 points/inch and a sampling rate of 100 points/second. The participants were asked to print characters in boxes appearing on the paper attached to the tablet. The character set contained 44 characters and consisted of the upper-case alphabet, the eight symbols () . * + - / = and the digits. The output of the digitizing tablet was then processed by a character recognizer and the recognized characters were output to the participants via the display of the PC/AT.

The data collected from the copy task of the experiment described above was sent to the mainframe IBM System 370 for the processing described in the present experiment. For this experiment the recognition software was improved. The full set of algorithms used in this system are described together, in detail, for the first time. Many of the algorithms were previously described separately and appropriate references will be cited below.

2.1.1. Preprocessing

Several techniques, essentially algorithms from signal processing, were applied with the aim of reducing the noise originating from the limiting accuracy of the tablet, the digitizing process, erratic hand motion, and the inaccuracies of pen-down indication. These techniques were applied separately to each stroke, where a stroke is the writing from pen-down to pen-up. They were also applied in the order described unless otherwise indicated.

Wild point correction was used to correct for an occasional spurious point, usually caused by hardware problems. This algorithm is based on the acceleration of the movement of the stylus tip. If the magnitude of either the x or y acceleration exceeds a threshold, the point is declared to be wild. The accelerations are computed simply as the rate of change of velocity, and velocity as the rate of change of position. Thus, the acceleration in the x dimension at point i is $a_i = (x_{i+1} - x_i) - (x_i - x_{i-1})$. Acceleration is used because it is limited by the mass of the hand and pen and the muscular forces driving the pen motion. Wild points are corrected by replacement with the average of the preceding and following points. This method was described previously in more detail.³ Examples of wild point correction by this method are shown in Fig. 1. Functionally similar procedures have been reported previously.⁴⁻⁷

Smoothing was performed by averaging a point with its neighbors, a technique often used.⁸⁻¹¹ A frequently used technique averages a point only with previous points so that the computation can be performed as each point is received.^{8,11} We replaced a point with a symmetric weighted average of its neighbors—a 5-point least-squares fit recommended by H. Ellozy, i.e. $x_i = (-3x_{i-2} + 12x_{i-1} + 17x_i + 12x_{i+1} - 3x_{i+2})/35$. This smoothing was successfully applied to the data of each stroke n times, where n was empirically determined.

Filtering was performed to eliminate duplicate data points and to reduce the number of points. The technique used forces a minimum distance between consecutive points and has been used in other systems.^{4,5,8-12} This method tends to produce equally spaced points, suitable for subsequent elastic matching.

Merely forcing a minimum distance between consecutive points has the drawback of not always capturing the full extent of pen movement at cusps and corners. This is illustrated in Fig. 2. Here, points h and i are retained points and point k is the next point sufficiently far from point i to be retained. Point j is between points i and k but would be skipped because it is not sufficiently far from point i . Therefore, an enhancement was made to this algorithm in order to capture cusps and corners more precisely. For significant direction changes (ϕ_k greater than an empirically determined threshold), the intervening points between i and k are examined to determine whether there are any in an acceptance region (Fig. 3). If there are, the one farthest from point h is retained. This enhancement improves the writing shape representation while increasing the number of points retained by only about three percent. This filtering method has been described in more detail.¹³ Another filtering method with similar properties has also been reported.¹⁴

Dehooking algorithms were used to eliminate extraneous data having the appearance of 'hooks'. Hooks can occur at stroke edges, that is, the beginnings and ends of strokes. They are due to inaccuracies in pen-down detection and rapid and sometimes erratic motion on the part of the writer in placing the stylus on, and lifting it off, the surface of the tablet. Two algorithms are used to eliminate most hooks with minimal elimination of true data.

The first procedure searches the edges of strokes (up to a specified number of points from the edge and a specified distance relative to character height) for a steady state,

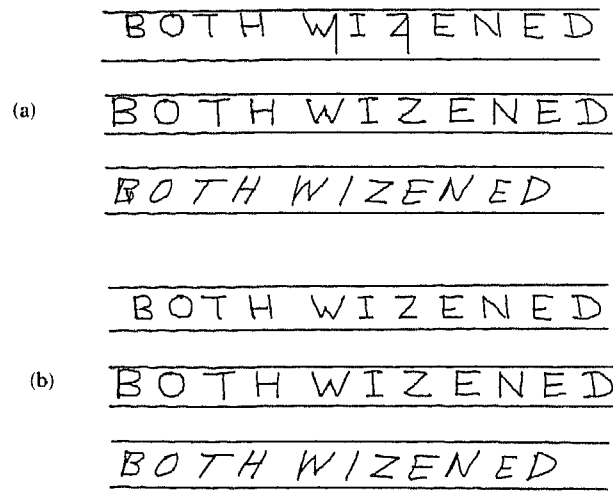


Fig. 1. Wild point correction, (a) before and (b) after.

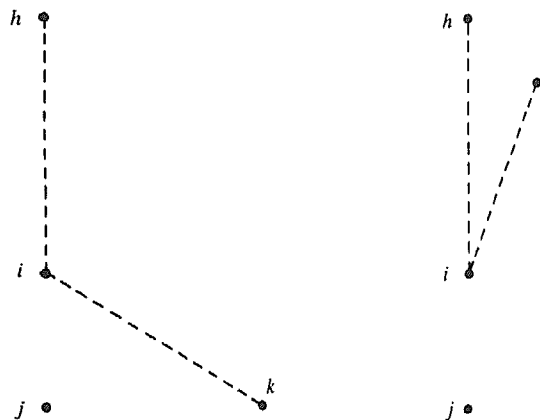


Fig. 2. Problems with usual filtering.

that is, a sequence of points in which little movement occurs. Since such steady states generally occur between the hook and what is considered the stroke data itself, a hook can be deleted by locating the steady state near the edge of a stroke and deleting the data points from the edge to the steady state. This first procedure is invoked before filtering since it depends on timing to detect a steady state.

The second procedure was designed to eliminate any remaining hooks at stroke ends. Hooks at stroke ends are more frequent and require this additional procedure to enhance hook removal. The second procedure, invoked after filtering, looks for a significant direction change (greater than a threshold) at the end of the stroke, and if

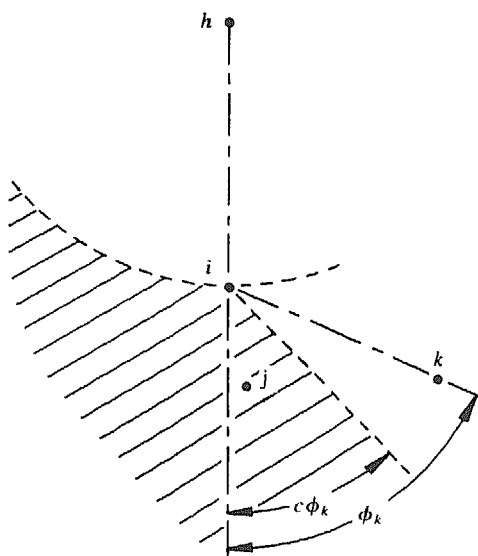


Fig. 3. Acceptance region for improved filtering.

the size of the hook is small relative to character height (by a threshold), the hook is eliminated. Figure 4 illustrates a hook that can be removed by this procedure. These methods have been described in more detail.¹⁵ Functionally similar methods have been reported.^{16, 17}

Dot reduction was performed to reduce dots to single points. Two procedures were used to deduce the data of most dots to single points with minimal reduction of non-dot data. The first procedure reduces a small stroke to a point if the stroke fits within a box that is small relative to normal character height. Specifically, if the maximal extent (in the x or y direction) of the stroke is within a threshold, then the stroke is reduced to a single point. This procedure is done before filtering.

Some writers make scribbled dots. The second procedure, invoked after filtering, is aimed at reducing such dots to single points. Three conditions must be met. Firstly, the maximal extent of the stroke must be within a threshold. This threshold is larger than that of the first procedure, and it is more stringent for a stroke of a multi-stroke character than for a character consisting of a single stroke. Secondly, the ratio of the number of stroke points to the maximal extent must exceed a threshold (ensuring a high density of points). And thirdly, the average direction change in going from point to point within the stroke must exceed a threshold (scribble characteristic). If these conditions are met, the stroke is reduced to a single point.

2.1.2. Pruning

Pruning is the elimination of some prototypes from contention prior to elastic matching. Since the elastic matching procedure used can stretch or compress a prototype character up to double or half its length, minimal pruning eliminates those

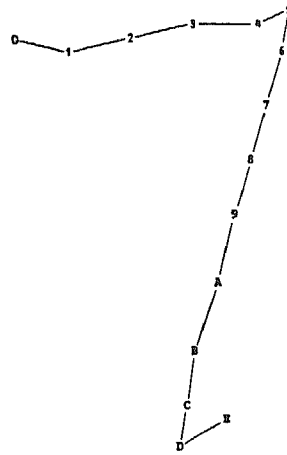


Fig. 4. Illustration of a hook.

prototypes whose lengths are not within this range. This minimal pruning was used to ensure that loss of accuracy could not be attributed to pruning. We have previously investigated feature analysis techniques for pruning.^{18,19}

2.1.3. Parameter extraction

From the points remaining after preprocessing, three parameters were extracted to characterize each coordinate point. The first and second parameters were the normalized x and y offsets, x_i and y_i , from the center of gravity. The third parameter was the slope angle, ϕ_i , of the tangent to the curve at point i , approximated by the slope angle of the line segment from one point to the next. The original system used four parameters, these three and the y -offset from the baseline.¹

The normalization of the x and y offsets was such that the y difference of the box height, corresponding roughly to the height of most characters, was forced to the greatest possible slope-angle difference (180 degrees). A character is represented by a sequence of parameter vectors $S = V_1, V_2, \dots, V_N$, where $V_i = (x_i, y_i, \phi_i)$ and N is its length in number of points.

2.1.4. Elastic matching

In matching an unknown against a prototype, a distance metric is used to compare an arbitrary point i in the unknown with a point j in the k th prototype. The point distance is

$$d(i, j; k) = (x_i - x_{j,k})^2 + (y_i - y_{j,k})^2 + c|\phi_i - \phi_{j,k}|$$

where the first two terms are squares of the x and y differences and the third term is a constant (empirically determined) multiplied by the slope-angle difference.

The point distance is used to obtain an overall distance for comparing an unknown character against a prototype. The overall distance permits non-linear elasticity on the part of the prototype, stretching up to a factor of two and compressing up to a factor of one half. This distance is computed efficiently by dynamic programming using the recursion relation

$$D(i, j; k) = d(i, j; k) + \min \begin{cases} D(i-1, j; k) \\ D(i-1, j-1; k) \\ D(i-1, j-2; k) \end{cases}$$

where $D(i, j; k)$ is the cumulative distance from $(1, 1)$ to (i, j) . Starting with $D(1, 1; k) = d(1, 1; k)$ and with $D(i, j; k)$ large (infinite) elsewhere, the cumulative distance recurrence relation is used to obtain, for all possible paths through the model, the minimum distance $D_k = D(N, M_k; k)$ between prototype k and the unknown, where N is the length of the unknown and M_k is the length of prototype k . Recognition is performed by matching the unknown character against each prototype (remaining after pruning) and choosing that character yielding the smallest overall distance. This form of elastic matching was described previously¹ and has been used in speech recognition.²⁰ An illustration of this elastic matching is shown in Fig. 5.

2.1.5. Prototype establishment

Each participant wrote 965 characters during a training phase.² Prototypes were established from these characters. The system processed each of these characters sequentially and a character was added to the prototype set if it was either a new character or did not closely resemble previous prototypes for the same character. Whether a character resembled previous prototypes was determined by applying the same procedure that was used during recognition. Specifically, if a character was not recognized correctly or if an incorrect second choice was close to a correct first choice (the ratio of the elastic matching score of the second choice to the first was less than a specified threshold), then that character was added to the prototype set.

2.1.6. Recognition

The system classified the 476 characters of each participant, obtained from the copy task of the recognition phase,² by comparing each to the prototypes obtained from the participant.

2.1.7. Algorithm improvement

The essential ideas used to improve the algorithms were determined at the outset. Experimental verification of these ideas and empirical determination of some of the thresholds was made using the recognition data from writers 1, 4, and 7.

Recognition accuracy was improved in four ways. Firstly, floating-point arithmetic was used in preprocessing and parameter extraction, except for a few cases where scaled-up integers were used. Secondly, Euclidean distance calculations were used in place of absolute x and y differences. (Actually, squared Euclidean distance was used

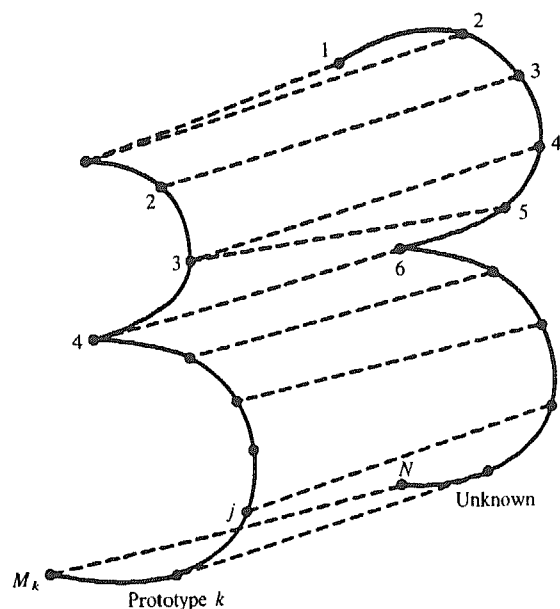


Fig. 5. Elastic matching.

for efficiency of computation.) Thirdly, a term previously used in the point-distance metric was omitted. This term was the y offset from the baseline which permitted distinction between certain upper- and lower-case letters such as between P and p , where the critical difference is the position of the character relative to the baseline.¹ However, since this distinction was not necessary for the alphabet used, this term would only add noise to the metric. Finally, the use of greater precision in specifying the coordinate values permitted the addition of smoothing.

2.2. Results

The results of this experiment are shown in Table 1. The average error rate was cut in half, from 5.6% in the original experiment of Schoonard *et al.* to 2.7% as reported here and also by Schoonard *et al.*² There was an average of 135.4 prototypes per writer, or about 3 prototypes per alphabetic character.

The data of writers 1, 4, and 7 were used to arrive at the improved algorithms. Although it is not usual to include such data in reporting recognition accuracy, it was done to be fair, because to exclude them would result in higher recognition accuracy (97.8%). Determination of the value of the various modifications was difficult primarily because their order of implementation influences the degree of improvement in accuracy. However, significant increases in recognition accuracy were obtained by eliminating the y offset term from the point-distance metric and by using squared Euclidean distance in trimming.

Table 1. Stroke-number-free recognition accuracy.

Writer										
	1	2	3	4	5	6	7	8	9	Average
Accuracy	98.5*	98.5	99.0	96.6*	96.6	97.3	93.9*	97.5	98.1	97.3

* Algorithms tailored to data upon which these values are based.

In order to achieve this increase in recognition accuracy, computation time increased by a factor of two. Also, because the changes involved conversion from fixed- to floating-point arithmetic in elastic matching, a PASCAL elastic match was used rather than the previous machine-language match. Pruning limited the number of prototypes matched to about 75% of the total number of prototypes. The speed of computation was about 9 characters/second on the IBM System 370/Model 3081, with most of the computation time (over 90%) being used for elastic matching.

3. EXPERIMENT 2: STROKE-NUMBER-CONSTRAINED RECOGNITION

The purpose of this development phase was to improve the computation speed of the system without decreasing recognition accuracy. Computational speed is important because, although the recognition system was developed on the IBM System 370, it is being downloaded to smaller, less-powerful machines.

3.1. Procedure

The procedure was identical to that of Experiment 1, except for the modifications described below. The basic changes were matching on a stroke-by-stroke basis and adding a fast first-stage linear match.

3.1.1. Preprocessing

Wild point correction was not performed. In the previous experiment it was observed that no wild points were found in the data. Since the wild point correction algorithm was not utilized, it was disabled.

3.1.2. Pruning

Since matching was done on a stroke-by-stroke basis, those prototypes having a different number of strokes compared to the unknown were pruned. Also pruned were prototypes in which the length of any stroke was not within the elasticity range of half to double the length of the corresponding stroke of the unknown.

3.1.3. Parameter extraction

Although parameter extraction was unchanged, it should be noted that the x and y offsets were still computed from the center of gravity of the whole character, even though matching was performed on a stroke basis.

3.1.4. Linear matching

A linear match was performed to further reduce the number of prototypes to be elastically matched against the unknown. This is essentially an additional method of pruning.

This linear match was performed on a stroke-by-stroke basis. Within each stroke, the points of the unknown were linearly matched against points of the prototype (using the point-distance metric) as follows. The first point of the unknown stroke was matched to the first point of the prototype, and the last to the last. Each intermediate point of the unknown was matched against the linearly nearest point of the prototype. This is illustrated in Fig. 6.

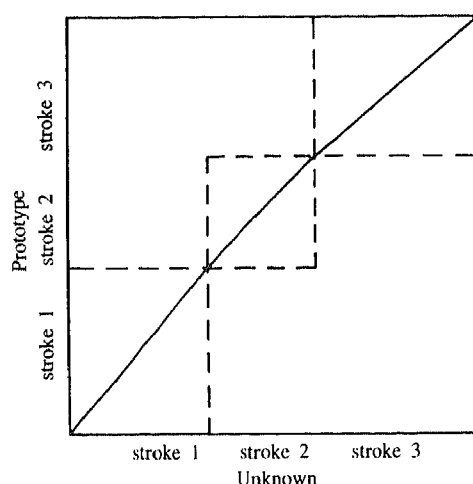


Fig. 6. Linear stroke-by-stroke matching.

The minimum linear-matching score was found and a *cutoff-threshold multiplier* (constant) multiplied by that minimum was used as a cutoff threshold. That is, all prototypes with linear-matching scores greater than this cutoff threshold were pruned (Fig. 7).

Figure 8 shows the percentage error rate (the average of writers 1, 4, and 7) as a function of the cutoff-threshold multiplier. The multiplier controls the amount of elastic matching performed. A multiplier of 1.0 represents a pure linear matcher since all but the prototype with the minimum score is pruned and no elastic matching is required. On the other hand, a high multiplier represents a pure elastic matcher since no prototypes are pruned and all are elastically matched. The cutoff-threshold multiplier was set (at 1.7) to yield the low error rate of elastic matching, while providing a reasonable increase in the speed of computation.

3.1.5. Elastic matching

The elastic matching procedure in Experiment 1 was used on a stroke-by-stroke basis. This is illustrated in Fig. 9 which shows the reduced computation region for

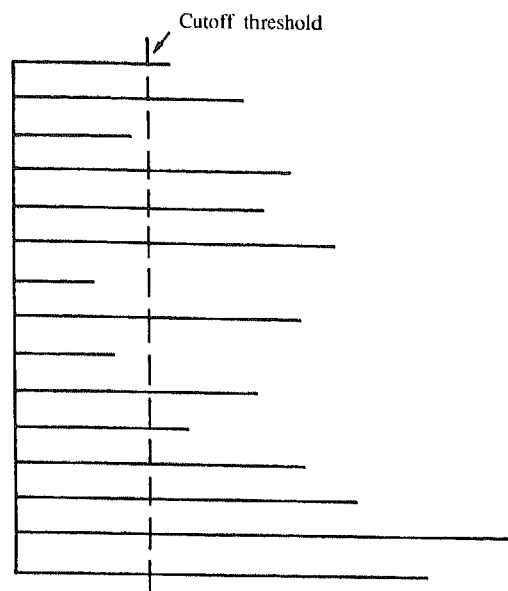


Fig. 7. Pruning by linear matching. Horizontal line lengths represent prototype matching scores. Prototypes whose match scores exceed a threshold (constant multiplied by the minimum match score) are pruned.

stroke matching compared to that of the original character matching; computation regions are bordered by lines of slope 2 and slope 1/2. (Note: for n equal-length strokes the area of the computation region for stroke matching is $1/n$ that for character matching.) The matching score for a character is merely the sum of the matching scores for each stroke of the character.

3.1.6. Computation time analysis

A program profiler (EPSCA) recorded the percentage of time used by the program statements during execution.²¹

3.2. Results

The results of this experiment are shown in Table 2. The recognition accuracy is essentially unchanged. Minor differences are due to fluctuations in the discrimination of similarly shaped characters. However, the speed of computation increased by an order of magnitude, from 9 to 85 characters/second on the IBM System 370/Model 3081. Initial pruning and the smaller computation region of stroke-by-stroke matching increased computation speed by a factor of four. The linear-match pruning increased speed by an additional factor of two.

In this experiment initial pruning reduced the number of prototypes to about 29% for linear matching, and the fast linear match further reduced this number to only about 3.8% of the total number of prototypes. Thus, with an average of 135.5 prototypes per writer, only about five prototypes were elastically matched against an unknown.

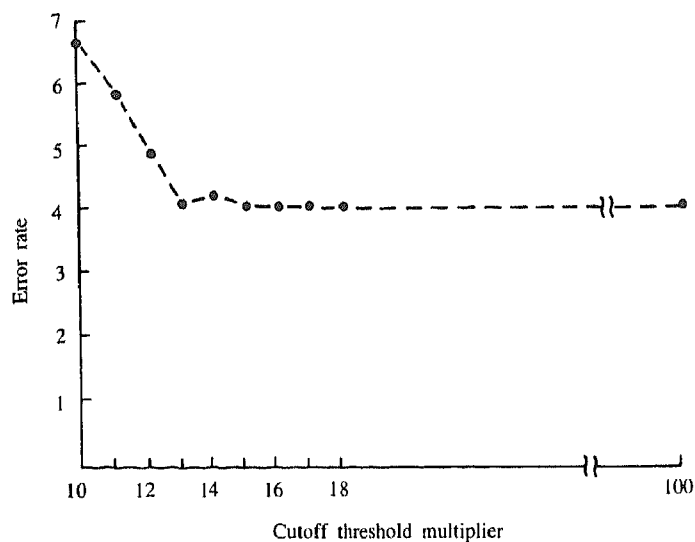


Fig. 8. Linear versus elastic matching.

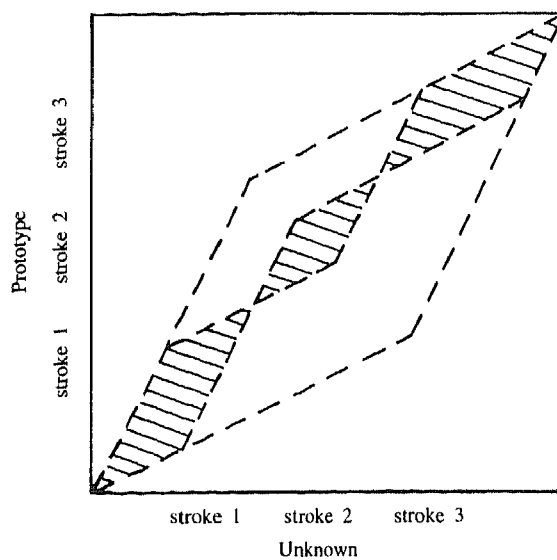


Fig. 9. Stroke-by-stroke elastic matching showing the computation region for stroke matching (hatched) and that for stroke-number-free matching (large parallelogram).

The distribution of the number of prototypes remaining at elastic matching is shown in Table 3. Over a third of the time, linear matching reduced the number of prototypes (choices) to one so that elastic matching was actually not necessary, although it was done to obtain comparable scores from the matching process. The number of occurrences of ten or more prototypes is indicative of the frequency of characters for which there were similar or unusual prototypes.

Using the EPSCA program,²¹ a breakdown of the computation time by function during recognition for writer 1 was obtained (Table 4). The matching process, linear and elastic, used 43.8% of the computation time.

Table 2. Stroke-number-constrained recognition accuracy, with prototypes initially, and at linear and elastic matching.

	Writer									
	1	2	3	4	5	6	7	8	9	Average
Accuracy	98.3*	97.9	98.3	95.2*	96.8	97.7	94.3*	97.7	99.0	97.2
Number of prototypes	92.	103.	106.	177.	114.	141.	191.	162.	134.	135.5
At linear matching	23.5	33.9	39.3	52.2	23.5	30.8	72.2	38.9	37.7	39.1
At elastic matching	3.1	4.3	4.2	7.4	3.6	3.8	7.2	8.5	4.4	5.2

* Algorithms tailored to data upon which these values are based.

Table 3. Frequency distribution (percent) of prototypes at elastic matching.

Number of prototypes	Writer									
	1	2	3	4	5	6	7	8	9	Average
1	45.6	44.9	33.3	17.8	41.1	40.5	32.7	30.4	38.8	36.1
2	17.7	10.8	12.4	12.7	10.4	13.1	9.3	21.7	13.5	13.5
3	10.1	7.2	13.3	8.0	8.0	8.6	7.2	14.3	9.1	9.5
4	5.9	5.5	11.0	10.2	7.4	6.1	9.6	4.9	5.1	7.3
5	5.1	5.3	6.8	7.6	6.6	10.3	5.3	4.9	4.0	6.2
6	3.6	7.2	4.6	6.1	9.7	4.4	4.7	1.5	4.2	5.1
7	3.0	3.4	4.9	4.0	6.6	3.0	3.0	1.7	4.6	3.8
8	1.3	3.2	3.6	4.7	4.7	2.5	2.8	2.1	3.2	3.1
9	1.9	4.0	1.1	2.8	0.9	3.0	2.6	3.0	2.3	2.4
10 or more	5.9	8.6	9.1	26.2	4.7	8.4	22.9	15.6	15.2	13.0

Finally, it was found that elastic matching reduced the error rate of linear matching by about 40% (Fig. 8). A similar figure was also obtained for all nine writers. This quantifies the superiority of elastic over linear matching.

Table 4. Computation time (percent) per function.

Function	Time
Smoothing	7.4
Filtering, dehooking, dot reduction	9.2
Pruning	10.0
Parameter extraction	3.8
Linear matching	27.8
Elastic matching	16.0
Storage allocation and disposal	9.4
Miscellaneous	16.4

4. DISCUSSION

4.1. Speed and Accuracy

In the two experiments described above, compromises were made among the properties of speed, accuracy, and flexibility. It was found that high recognition accuracy can be achieved together with high computation speed. In particular only 16% of the time (Table 4) was required for elastic matching to decrease the linear-match error rate by about 40% (Fig. 8). Also, the additional computation required for greater precision is also small on most contemporary machines. Therefore, the only possible disadvantage in the trade-offs implemented may be flexibility.

4.2. Flexibility

Flexibility is an important property of handwriting recognition systems. Writing on a tablet with no constraints can be varied as writing with pen and paper. At the other extreme, the writing can be limited to characters which are printed carefully, one to a box, and each character can further be limited to a specific number, order, and direction of strokes. Generally speaking, flexibility is concerned with where the characters or words are written, such as on lined paper or spaced one from another, and how they are written, such as with variation in number, order, and direction of strokes.

Our approach has been to have each writer establish his own prototype set. All intra-writer variation is handled by the system merely by adding prototypes, thus increasing flexibility; however, the amount of variation handled is limited because it is restricted to a single writer. Significant intra-writer variation, such as differing stroke number, order, or direction, is handled merely by storing additional prototypes. While this approach is flexible in that the writer is not constrained in his character production, the writer must take time to create prototypes. For a relatively small alphabet (such as the Latin alphabet, including digits and special symbols) this approach is feasible because of the relatively small number of characters and also the few strokes per character. For Chinese characters a more sophisticated approach is needed to handle both stroke number and order variation without a corresponding increase in

prototypes. This is accomplished by matching each stroke of the unknown against each stroke of the prototype, finding the best stroke correspondence, and, when the stroke numbers differ, using linkage rules to combine strokes appropriately.^{22,23}

The system modifications described above resulted in a system which is more accurate and faster at the expense of some loss in flexibility in stroke number. This loss in flexibility will now be examined. Since our approach has been to add prototypes when necessary, there should be little or no decrease in the ability of the system to handle writer variation in stroke number. Although there was some concern that the number of prototypes might increase in going to stroke-by-stroke matching, this was not the case. A final consideration is that extraneous pen-lifts are a source of error for matching at the stroke level but not for matching at the character level. This was the primary reason for matching at the character level in the original system. However, another way to handle extraneous pen-lifts is to connect successive strokes when the distance between a pen-up and a subsequent pen-down is small relative to character size, a technique used previously.¹⁴ Thus, any loss in flexibility from matching at the stroke rather than the character level is minimal.

Finally, the omission of the y offset from the baseline is considered (Experiment 1). This change reduced the generality of the recognizer, since it could no longer distinguish between characters whose essential difference is the position of the character relative to the baseline. However, discrimination based on character position relative to the baseline can be added in a post shape-recognition phase, which seems more appropriate. Furthermore, this change resulted in a matching procedure which responds more accurately solely to the shape of a character. Therefore, there is no loss in flexibility from this change.

4.3. Future Accuracy Improvement

Further improvement in recognition accuracy can be anticipated. Firstly, the prototype establishment procedure can be improved. For example, the writer could be given the capability to interactively review machine-drawn plots of prototype characters and to directly insert and delete prototypes.²⁴ Furthermore, the serial prototype establishment procedure could be replaced by a global one which examines a set of candidates and derives a set of prototypes from them. One such procedure promotes the identification and usually the elimination of mavericks,²⁵ where a *maverick* is a character that is not written as intended, inadvertently not made differently from other alphabet characters, not captured accurately due to hardware problems, or distorted in some fashion due to other problems. In these experiments maverick prototypes were an important cause of error since, for example, some writers wrote different characters identically, such as 'O' and '0' and 'I' and 'l'.

Secondly, postprocessing can be performed to distinguish between similarly shaped characters. One such approach is based on *functional attributes*.²⁶ For example, 'A' versus 'H' and 'O' versus 'C' can be discriminated by the attribute of closure, the first of each pair having, and the second lacking, the attribute.

Thirdly, improvements to preprocessing are possible. The dehooking and dot

reduction procedures merit review since they were designed using a relatively small sample of data from a different tablet. Also, changing to float-point arithmetic in the few cases where scaled-up integers are still used may further increase accuracy.

4.4. Future Speed Improvement

Additional improvement in computation speed can be anticipated. The storage allocation and disposal can be eliminated by creating a free list for the structures. The linear and elastic matching procedures are small and could be converted to machine language. Also, some speed could be gained by not doing elastic matching when only one choice remains from linear matching. Finally, the speed of many functions could be increased. For example, a function like pruning is particularly slow because the program was designed for pruning by feature analysis.

ACKNOWLEDGEMENTS

Deserving of thanks are T. Fujisaki and H. A. Ellozy for their support, the late Shawhan Fox for valuable discussions during the implementation and testing of the improved algorithms, and J. S. Lipscomb who read the manuscript and offered helpful suggestions.

REFERENCES

1. C. C. Tappert, "Adaptive on-line handwriting recognition", *Proc. 7th Int. Conf. on Pattern Recognition*, 1984, pp. 1004-1007.
2. J. W. Schoonard, J. D. Gould, M. Bieber and A. Fusca, "A behavioral study of a computer hand print recognition system", IBM Research Report RC 12494, Feb. 1987.
3. J. Kim and C. C. Tappert, "Correction of wild points within an electronic tablet data", *IBM Tech. Disclosure Bull.* **29** (1987) 4042-4045.
4. S. Hanaki, T. Temma, and H. Yoshida, "An on-line character recognition aimed at a substitution for a billing machine keyboard", *Pattern Recogn.* **8** (1976) 63-71.
5. M. Hosaka and F. Kimura, "An interactive geometrical design system with handwriting input", *Proc. IFIP Symposium*, 1977, pp. 167-172.
6. K. Odaka, H. Arakawa and I. Masuda, "On-line recognition of handwritten characters by approximating each stroke with several points", *IEEE Trans. Syst. Man Cybern.* **12** (1982) 898-903.
7. P. J. Pobjee and J. R. Parks, "Application of a low cost graphical input tablet", *Inf. Process.* **71** (1972) 738-741.
8. H. Arakawa, K. Odaka and I. Masuda, "On-line recognition of handwritten characters - Alphanumerics, Hiragana, Katakana, Kanji", *Proc. 4th Int. Joint Conf. on Pattern Recognition*, Nov. 1978, pp. 810-812.
9. M. I. Bernstein, "A method for recognizing handprinted characters in real-time", *Pattern Recognition*, Ed. L. N. Kanal, Thompson, 1968, pp. 109-114.
10. B. Blesser, "Multistage digital filtering utilizing several criteria", United States Patent, 4,375,081, Feb. 1983.
11. G. F. Groner, "Real-time recognition of handprinted text", *Proc. FJCC*, 1966, pp. 591-601.
12. E. F. Yhap and E. C. Greanias, "An on-line Chinese character recognition system", *IBM J. Res. Dev.* **25** (1981) 187-195.

13. J. Kim and C. C. Tappert, "Tablet image filtering for recognition purposes", *IBM Tech. Disclosure Bull.* **28** (1985) 1158-1161.
14. M. K. Brown and S. Ganapathy, "Preprocessing techniques for cursive script word recognition", *Pattern Recogn.* **16** (1983) pp. 447-458.
15. C. C. Tappert, "Dehooking procedure for handwriting on a tablet", *IBM Tech. Disclosure Bull.* **27** (1984) 2995-2998.
16. E. Mandler, "Advanced preprocessing technique for on-line script recognition of nonconnected symbols", *Proc. 3rd Int. Symp. on Handwriting and Computer Applications*, July 1987, pp. 64-66.
17. J. R. Ward, "Method and apparatus for removing noise at the ends of a stroke", United States Patent, 4.534.060, Aug. 1985.
18. M. Karnaugh, J. M. Kurtzberg and C. C. Tappert, "Prototype pruning for elastic matching symbol recognition", *IBM Tech. Disclosure Bull.* **24** (1982) 4522-4523.
19. J. M. Kurtzberg, "Feature analysis for symbol recognition by elastic matching", *IBM J. Res. Dev.* **31** (1987) 91-95.
20. F. Itakura, "Minimum prediction residual principle applied to speech recognition", *IEEE Trans. Acoustics, Speech, Signal Processing* **23**, Feb. (1975) 67-72.
21. L. R. Power, "EPSCA Distribution for VM/CMS", IBM Internal Report, 1985.
22. T. Wakahara and M. Umeda, "Stroke-number and stroke-order free on-line character recognition by selective stroke linkage method", *Proc. 4th ICTP*, 1983, pp. 157-162.
23. T. Wakahara and M. Umeda, "On-line cursive script recognition using stroke linkage rules", *Proc. 7th Int. Conf. on Pattern Recognition*, 1984, pp. 1064-1068.
24. A. S. Fox, J. Kim and C. C. Tappert, "Prototype Management and Error Correction Scenarios", IBM Research Report RC 14054, Sept. 1988.
25. A. S. Fox and C. C. Tappert, "A global prototype establishment procedure for handwritten character recognition", *IBM Tech. Disclosure Bull.* **24** (1989) 114-116.
26. B. Blesser, R. Shillman, T. Kuklinski, C. Cox, M. Eden and J. Ventura, "A theoretical approach for character recognition based on phenomenological attributes", *Proc. 1st Int. Joint Conf. on Pattern Recognition*, 1973, pp. 33-40.

Received 15 December 1989.



Charles C. Tappert received a B.S. degree from Swarthmore College, and M.S. and Ph.D. degrees from Cornell University. He joined IBM in 1967 and has been a Research Staff Member at the IBM T. J. Watson Research Center since 1972. His research

has primarily been in the area of pattern recognition. He worked on automatic speech recognition and speech coding for ten years. Since 1973 he has been working on tablet terminals, his main interest being computer recognition of hand-printing and cursive writing.