# REAL-TIME RECOGNITION OF HANDPRINTED TEXT *

Gabriel F. Groner

*The RAND Corporation*
*Santa Monica, California*

## INTRODUCTION

During the past 10 years, there has been considerable interest in the automatic recognition of handwritten and machine-printed symbols.[1-7] Most of these studies have involved computer recognition of an already completed symbol; a few have involved special electronics for the analysis and recognition of characters [8] or words [9] as they are being written.

Only recently, with the advent of on-line computer systems, have computers been used for the real-time recognition of handprinted symbols.[10-13] Aside from Bernstein's recent work,[13] real-time schemes to date have been concerned only with the recognition of single, isolated symbols. None of the real-time schemes (to the author's knowledge) have been used in a problem-solving environment, or have utilized contextual information for recognition.

This paper describes a symbol recognition scheme which allows an on-line computer user to print text naturally, and have it recognized accurately, as he prints it. The scheme responds very quickly even though it recognizes a fairly large set of symbols. Moreover, it imposes few constraints on style, speed, or position of writing. It makes use of contextual information to distinguish symbols which cannot be distinguished by shape alone. This scheme has been used daily at The RAND Corporation for writing computer code, drawing flow charts, and editing. The symbol recognition scheme is written in IBM System/360 Assembly Language and runs on an IBM System/360 Model 40.

The symbol set consists of the upper-case Latin alphabet, the numbers, the symbols +, −, =, /, (, ), [, ], *, $, ., ,, ', ∧, >, <, and a scrubbing action which causes erasure of underlying symbols. The scheme also recognizes a set of flow-charting symbols, but these will not be discussed. When any other symbol is drawn, it is either identified as one of the above, or as a "cannot interpret." This symbol set is sufficiently large to enable a user to communicate data and directives to a computer by using *only* a pen-like instrument.

The scheme is designed so that the user can concentrate on his problem, rather than on extraneous operational mechanics. It therefore accommodates a variety of printing styles, allows for the printing of several symbols in quick succession, and responds quickly. Finally, any errors made by the scheme may be corrected easily.

## USER INTERACTION

A user communicates with the computer via a RAND tablet [14] in conjunction with a cathode ray tube (CRT). The tablet hardware consists of a

* This research is supported and monitored by the Advanced Research Projects Agency under Contract No. SD-79. Any views or conclusions contained in this paper should not be interpreted as representing the official opinion or policy of ARPA or The RAND Corporation.

horizontal 10-in square writing surface and a pen-like writing instrument. Figure 1 depicts a user interacting with this hardware. As the user moves the pen near the tablet surface, a dot on the CRT follows the pen motion—this direct feedback helps him position the pen for pointing or drawing. When he presses the pen against the tablet writing surface, a switch in the pen closes, thereby notifying the central processing unit (CPU) of a "pen-down" action. As he moves the pen across the tablet, the pen's track is displayed on the CRT—the pen "point" thus seems to have "ink." When the pen is lifted, the pen switch is opened, thereby notifying the CPU of a "pen-up" action, and "inking" ceases. A new user easily adjusts to watching the pen motion on the CRT while actually moving the pen off-screen on the tablet. Furthermore, he finds it convenient to have no part of the CRT "working surface" covered by his hand or pen.

The user must print plainly; i.e., not use curls or scrolls, and not drag the pen between strokes. The letter O must be written with a slash to distinguish it from the number 0, the letter I with serifs to distinguish it from 1, and the letter Z with a crossbar to distinguish it from 2 (these conventions are commonly followed when writing on coding forms). The user may otherwise print in his normal style. When the user's "inked" symbol is recognized, it is replaced by a hardware-generated version. He may change this symbol at any time, merely by writing over it.

Although the user may write anywhere on the tablet surface, he must follow some conventions of size and position just as he would when writing on a piece of paper. The display format subtly conditions him by displaying a recognized symbol in a standard size (full-size symbols are about $\frac{3}{16}$-in high) at the standard location (one of a possible 3570) nearest to the center of his "inked" version of the symbol. Since the user writes next to the standard-size symbols uniformly spaced along lines, he soon learns to adjust his printing accordingly. This symbol size and separation are about the same as those commonly required for printing on coding forms.

## ANALYSIS OF THE DATA

The purpose of data analysis is to extract those features most valuable for discrimination among the allowable characters; i.e., those features which have consistent values over variations of a particular symbol, yet which have different values for different symbols. Since a single scheme recognizes the printing of many users, any of whom may print sloppily, consistency is assured only if a symbol can be described by a few gross features.

The on-line nature of this recognition scheme enables processing of the data point-by-point as the pen is moved across the writing surface. In order to minimize time and storage requirements, therefore, the scheme extracts features as the data arrive.

### *The Data*

Pressing the pen against the writing surface activates the symbol recognition scheme by indicating the start of a stroke. As the pen is moved across the writing surface, the recognition scheme is notified of its position every 4 msec. Finally, when the pen is lifted off the surface, the recognition scheme is notified that the stroke is completed. Each pen position is accurate to about 0.005 in. The hardware thus provides a very detailed—about 100 data-points—time-sequential description of each stroke.

### *Smoothing and Thinning*

The scheme smooths the data by averaging a newly arrived data-point with the previously smoothed data-point, thus reducing the noise due to

Figure 1. A user interacting with a RAND tablet and CRT.

the discreteness of the pen location as measured by the tablet. Smoothing is based on the equations

$$X_{Si} = \frac{3}{4}X_{Si-1} + \frac{1}{4}X_{Ri}$$

and

$$Y_{Si} = \frac{3}{4}Y_{Si-1} + \frac{1}{4}Y_{Ri}$$

where $X_{Ri}$ and $Y_{Ri}$ are coordinates of the $i$th raw data-point, and $X_{Si}$ and $Y_{Si}$ are coordinates of the $i$th smoothed data-point.

Thinning is the process of removing some of the data-points from the pen track. This is accomplished by comparing the position of a new smoothed data-point with the position of the last point in a thinned track. If these points are sufficiently far apart, the analysis scheme accepts the smoothed point as part of the thinned track; otherwise, it is discarded. Thinning eliminates small perturbations in the track, and reduces the data processing requirements by drastically reducing (by a factor of seven or so) the number of data-points. Thinning is described by

$$X_{Tj} = X_{Si}, \; Y_{Tj} = Y_{Si}$$

if

$$|X_{Si} - X_{Tj-1}| \geq \triangle$$

and/or

$$|Y_{Si} - Y_{Tj-1}| \geq \triangle$$

where $X_{Tj}$ and $Y_{Tj}$ are the coordinates of the $j$th thinned data-point, and $\triangle$ is a parameter of the analysis routine. The recognition scheme, which expects symbols to be drawn about $\frac{3}{16}$-in high, uses $\triangle = 0.02$ in. This value of $\triangle$ is large enough to eliminate insignificant data-points, yet small enough to maintain the essential characteristics of the track.

Figure 2 is a photograph of a display generated by a program which did the following:

1. Replotted without any processing each sampled point of a figure drawn at the tablet.
2. Magnified the original figure eight times.
3. Smoothed the track.
4. Thinned the track with $\triangle = 0.01$ in.

### *The Curvature Feature*

Curvature is the most obvious track characteristic which is independent of position and size, and yet which describes the track's shape. Freeman [15] has suggested that a useful approximation to curvature is the sequence of quantized directional segments generated by the points in a thinned track. Kuhl [5]

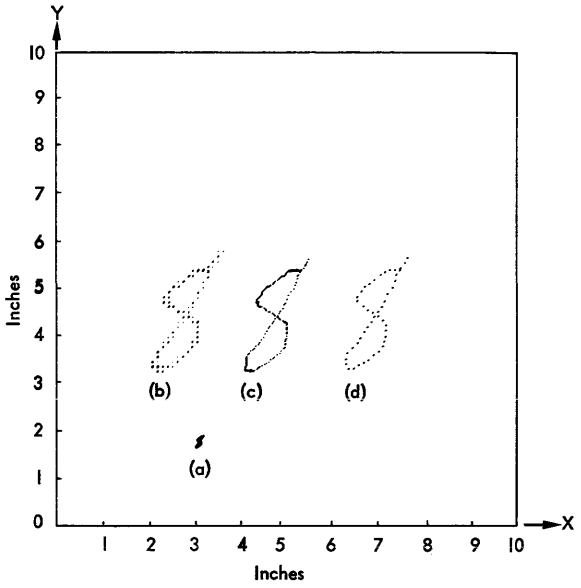


Figure 2. Display of a stroke—(a) as drawn, (b) magnified, (c) magnified and smoothed, (d) magnified, smoothed, and thinned.

and Bernstein [10] have used this approximation in their character recognition schemes. Bernstein, in fact, found it unnecessary to use the duration of each quantized direction but, rather, simply listed *changes* in quantized direction. Whereas Kuhl and Bernstein both used eight possible directions, the recognition scheme described here uses only four. Four directions, used in conjunction with other features, provide sufficient description for recognition, yet result in fewer symbol variations than do eight directions.

When a new point (the $j$th) is accepted in the thinned track, a quantized direction is computed using these inequalities: (1) if $|X_{Tj} - X_{Tj-1}| \geq |Y_{Tj} - Y_{Tj-1}|$,

(a) direction is right if $X_{Tj} - X_{Tj-1} \geq 0$
(b) direction is left if $X_{Tj} - X_{Tj-1} < 0$

or (2) if $|X_{Tj} - X_{Tj-1}| < |Y_{Tj} - Y_{Tj-1}|$,

(a) direction is up if $Y_{Tj} - Y_{Tj-1} \geq 0$
(b) direction is down if $Y_{Tj} - Y_{Tj-1} < 0$

If the same direction occurs twice in succession and is not the same as the last direction listed in the sequence, then it is added to the list; otherwise it is discarded. This requirement causes further thinning.

Small hysteresis zones (about 16° wide) around the quantized direction zone borders prevent the quantized directions in the approximate track description from switching back and forth—say from

right, to up to right—when part of a track is drawn along one of these borders. If, for example, the pen is moving to the right, it must next proceed at an angle steeper than 53° in order for the direction to change from right to up.

Applying these rules to the thinned track of Fig. 2 results in the direction sequence left-down-right-down-left-up.

*Inking and Corner Detection*

The CRT displays the pen track as "ink" constructed of a sequence of directional segments originating at the pen-down location. Sixteen possible directions are required to provide the user with sufficient "ink" detail. A segment is added to the "ink" each time a new thinned data-point arrives.

This same sequence of directions is used to detect sharp corners. A corner is detected whenever the pen moves in the same (±1) 16-direction for at least two segments, changes direction by at least 90°, and then proceeds along the new direction (±1) for at least two segments. The change in direction must take place either immediately or through a one-segment turn. Applying this test to the track in Fig. 2 detects corners in the upper-left and lowermost parts of the stroke.

*Size and Position Features*

As a stroke is drawn, its $x$ (horizontal) and $y$ (vertical) extremes are continuously updated. When the pen is lifted, thereby indicating the completion of the stroke, the analysis scheme uses these extremes to calculate the symbol's height and width in fractions-of-an-inch, its aspect ratio (ratio of height to width), and its center relative to the tablet origin. It divides the rectangular area defined by the symbol extremes into a 4 × 4 grid. The starting (pen-down) and ending (pen-up) points, as well as the corner locations, are then each encoded as lying in one of these 16 areas, thereby locating them relative to the symbol. Figure 3 shows the 4 ×4 grid defined by the track in Fig. 2. This track has the following description:

1. Six 4-direction segments (left-down-right-down-left-up) encoded as 2-3-0-3-2-1.
2. Corners at positions 7 and 15.
3. Starting point at position 0.
4. Ending point at position 0.

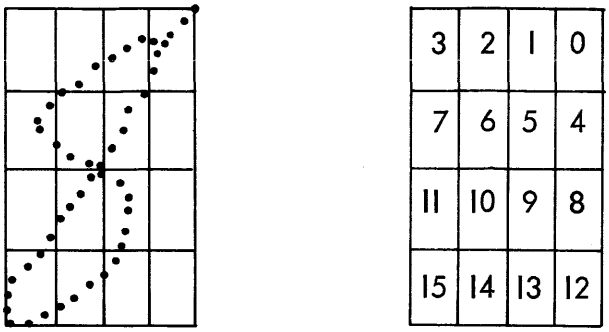


Figure 3. The 4 x 4 grid for the track of Fig. 2.

5. Height = 0.30 in.
6. Width = 0.22 in.
7. Aspect ratio = 1.36.
8. Center at $x$ = 3.15 in., $y$ = 1.75 in.

*Computing Requirements*

The data analysis described above takes up about 40% of the available computing time in the limiting case, where each data-point becomes a member of the thinned track. Smoothing and thinning requires about 20% of this analysis time, inking 30%, and corner detection 16%. The remainder of analysis time is for computing quantized directions, updating $x$ and $y$ extremes, and bookkeeping. The smoothing, thinning, and inking functions could be handled by hardware or by an I/O processor. Corner detecting could be deferred until stroke completion without requiring further storage (since it is based on the ink description, which is stored in any case), and without noticeably increasing the time between stroke completion and recognition. With these changes, a single user would require only about 15% of an IBM System/360 Model 40.

## DECISION-MAKING

Several characteristics of the decision-making scheme should be pointed out before it is described in detail. When a stroke is completed, its description is added to those of the other strokes belonging to the same symbol. The decision-making scheme identifies the partial symbol corresponding to this set of strokes but does not display its identification at this time. Such a partial symbol is considered complete when the user pauses, draws a somewhat distant stroke, or draws a stroke which cannot be combined with the partial symbol to form one of the 53 allowable symbols. This symbol is then displayed. The decision-making scheme thereby separates symbols

from one another and recognizes them as they are written.

The identification of a symbol is based on a data-dependent sequence of tests. At each step in the decision-making process there are several potential identifications. Some of these are eliminated by testing one of the features of the track. The particular test applied at any step depends on the set of possible identifications at that step, and on those characteristics of the track which have already been examined. The decision-making scheme thus has a tree structure. Its original design was based on an examination of the handwriting of four users. The author changes its structure, to accommodate additional symbol variations, as he acquires more experience.

*Identification of Single Strokes*

The first test groups the single-stroke partial symbols according to shape. This is accomplished by locating a sequence of direction segments in a table.

Only a few segments should be looked up in the table because each additional segment increases the table length by a factor of four (recall that there are four quantized directions). However, if too few are looked up, the test will not sufficiently reduce the number of possible stroke identifications in each shape group. A good compromise is to look up the first four direction segments of a track. If a track has fewer than four direction segments, it is encoded such that the last segment is repeated until there are a total of four. The table is therefore 256 entries long—160 ($4 + 4 \cdot 3 + 4 \cdot 3^2 + 4 \cdot 3^3$) of which correspond to allowable encodings. For this test, the track of Fig. 2 is encoded as 2-3-0-3 (left-down-right-down). The possible stroke identifications corresponding to this table entry are S,5,8,9, and "cannot interpret." Further tests, based on this particular set of possibilities, result in a single identification.

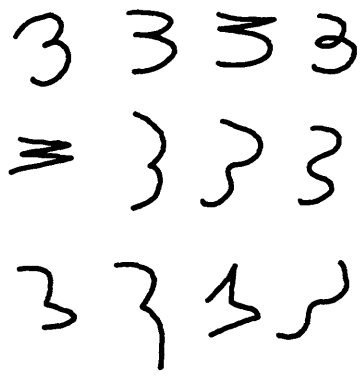

Figure 4. Some tracks recognized as the symbol 3.

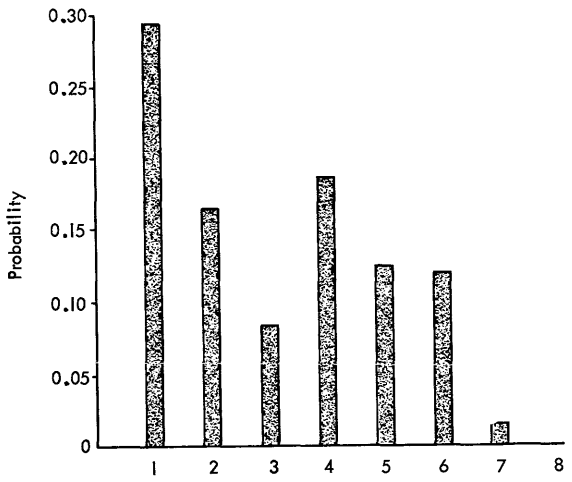


Figure 5. The probability of a number of symbols having the same first four direction segments.

The sequence of the first four direction segments of a track provides good first-level discrimination. In the present table, 45 of the 160 possible sequence encodings result in immediate identifications, with no further testing necessary. Another 50 sequences result in an immediate "cannot interpret."

To allow for variations in handwriting, each allowable symbol has many table entries. A track for the symbol 3— which has one of the most complex and varied shapes—may be drawn having any one of 26 different sequences of the first four direction segments. Figure 4 shows some of the tracks identified as the symbol 3.

Assuming that each of the single stroke partial symbols is equally likely, and that each four-direction segment description of any particular symbol is equally likely, the probability of having a number of symbols with the same table exit can be calculated. Figure 5 shows this probability of requiring further testing to discriminate among a number of symbols upon exiting from the currently used table. Approximately 30% of the direction sequences require no further testing. The probability of requiring further testing to discriminate among as many as four, five, or six symbols is fairly high, because several symbols might have the same first four directions in common, but other features which are different.

After this first test, the stroke is either recognized or is known to the one of a particular set of two-to-seven symbols. The second test is one which best (in the author's judgment) discriminates among these

particular symbols. Depending on the symbols, this test distinguishes differences in:

- The directions following the first four (e.g., to distinguish some 2's from some 3's)
- The number of corners (e.g., S vs 5);
- The positions of the starting or ending points relative to the symbol extremes (e.g., 0 vs 6);
- The aspect ratio (e.g., C vs ( );
- The size (e.g., , vs ) );
- The position relative to a line of text (e.g., , vs ′ ).

Testing continues until the number of possible stroke identifications is reduced to one. Figure 6 shows the sequence of testing required for the recognition of the track in Fig. 2. It is identified as the symbol 8.

The number of tests required for recognition is an interesting parameter because it measures the time and computer memory required for decision-making. The probability of requiring some number of tests for recognition can be calculated if it is assumed that each allowable single-stroke partial symbol is equally likely and that each decision tree-exit corresponding to a particular symbol is equally likely. Figure 7 shows the probability of requiring further testing after performing a number of tests in the existing decision tree. In more than half of all stroke recognition sequences, only two tests are required to arrive at a stroke identification. Figure 7 shows that ambiguity among several possible stroke identifications is always resolved after six tests.

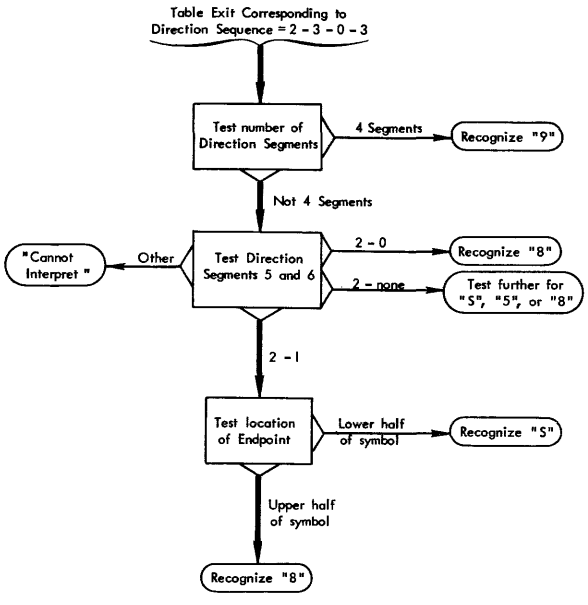


Figure 6. Recognition of the track of Fig. 2.

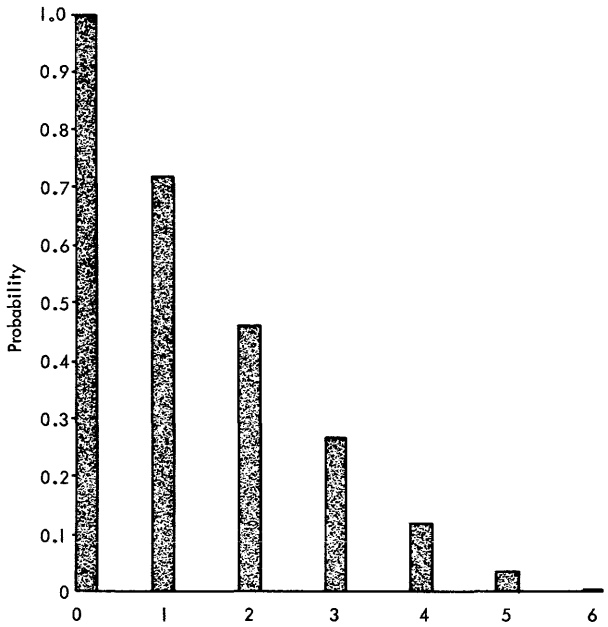


Figure 7. The probability of requiring further testing after $n$ tests.

*Recognition of Multiple-Stroke Symbols*

The recognition of a multiple-stroke symbol is based on the identifications and relative positions of its constituent strokes, but not on the order in which they are drawn. This is accomplished by cross-linking the tree structure. Each symbol has several descriptions, and therefore may be drawn in several ways. For example, a set of three horizontal strokes and one vertical stroke is recognized as the symbol E regardless of the writing order. The symbol E may also be written as an "L-like" stroke and two horizontal strokes, a "Γ-like" stroke and two horizontal strokes, a "Γ-like" stroke and one horizontal stroke, or a single "$\in$-like" stroke.

In many situations, each stroke requires only a general, rather than a precise, identification. For example, if a stroke known to be part of a multiple-stroke symbol is drawn vertically downward ($\pm 45°$), it need only be identified as a vertical stroke, rather than as a 1, ), (, or /. This reduces the compounding of errors that would otherwise result from incorrect identifications of single strokes, and also simplifies the decision-making.

Just as the starting and ending points of a single-stroke symbol are encoded according to their positions relative to the $x$ (horizontal) and $y$ (vertical)

extremes of that stroke, the starting and ending points of each stroke in a multiple-stroke symbol are encoded according to their positions relative to the $x$ and $y$ extremes of the whole symbol. These encoded positions differentiate those symbols which are comprised of the same combination of strokes. At most, four symbols have the same set of strokes.

A symbol comprised of two strokes identified as verticals is a K, V, X, or Y. This symbol is recognized as a K if one of the strokes has both its starting point and ending point in the leftmost quarter of the symbol. It is recognized as a V if both of its ending points are in the middle part of the lower quarter of the symbol. It is a Y if it has one ending point which is neither in the leftmost quarter, nor in the lower quarter. Otherwise, this symbol is an X. Other ambiguities among multiple-stroke symbols are similarly resolved.

*Segmentation into Symbols*

As the user writes a line of text, the recognition scheme decides when a partial symbol is completed, recognizes and displays it, and begins the analysis of the next symbol as it is being written. The scheme separates the set of strokes making up a completed partial symbol from the first stroke in the next symbol by considering timing, and the geometric extents and identifications of the partial symbol and the following stroke.

If a prespecified time elapses following the end of the most recent stroke, the corresponding partial symbol is considered completed regardless of what follows. This between-symbol time delay must be greater than the maximum expected time delay between two strokes belonging to the same symbol—0.3 sec has proven sufficient for experienced users. A sufficient pause between symbols eliminates the need to test for geometrical separation. This procedure reduces normal writing speeds by a factor of about one-half, but results in the most accurate segmentation.

A partial symbol is considered completed, regardless of timing or position, if it cannot be combined with the following stroke to form an allowable symbol. The symbols 8, Q, A, and E are examples of partial symbols which cannot be combined with *any* other stroke to form an allowable symbol. If a partial symbol can be combined with some strokes, but not with others, then the following stroke is potentially identified on the basis of its first four quantized direction segments. A test is then made to determine if the partial symbol can be combined with a stroke having this particular potential identification to form an allowable symbol. A partial symbol identified as a T, for example, may be combined with a vertical stroke to become an A, H, K, or *, or may be combined with a horizontal stroke to become an F or I, but cannot be combined with a potential C, U, or 2. In some situations, the following stroke must be identified more precisely. A circular stroke followed by a normal size "/-like" stroke, for example, may be identified as the letter O; but a circular stroke followed by a short "/-like" stroke will be identified as the number 0 followed by a comma.

Strokes which are written in quick succession, and which can be combined to form an allowable symbol are tested for geometric separation. Two such schemes for separating symbols geometrically have been investigated at RAND. One of these assumes that the writing surface is an 85 $\times$ 42 array of symbol spaces, and that each symbol is drawn in a different space. The other scheme assumes that the strokes in a symbol overlap (or, in some cases, are close to one another), but do not overlap with strokes belonging to another symbol.

*Fixed Symbol Spaces.* This scheme tests to see if the centers of a partial symbol and that of the next stroke lie in the same symbol space. If they do not, it separates them unless the partial symbol is a single vertical stroke lying just to the left of the following stroke. If they do lie in the same space, it considers them as part of the same symbol unless the stroke is a vertical in the rightmost quarter of the symbol space, *and* the partial symbol cannot be combined with a *right-hand* vertical to form an allowable symbol. These adjustments on the basic test allow for some misplacement of vertical strokes.

The user must be made aware of the symbol space positions so that he may space his printing accordingly. This is accomplished by displaying a series of dimly lit vertical bars, each at a border of a symbol space.

The primary disadvantage of this scheme is that small displacements of strokes relative to the symbol spaces cause segmentation errors. Such displacements may occur when the user writes quickly, or when the hardware displaces the "ink" slightly, thereby misinforming the user as to where he is writing. A further disadvantage is that the vertical bars are distracting.

*Relative Symbol Positions.* This scheme tests for the overlapping or adjacency of the partial symbol and

the following stroke. If the horizontal-extent of the stroke lies within the horizontal-extent of the partial symbol, it considers them parts of the same symbol. Partial-overlap causes the partial symbol and the following stroke to belong to the same symbol, unless their centers are farther apart than a maximum allowable distance. No-overlap separates the partial symbol and the following stroke, unless one (or both) is a single vertical stroke and their centers are not separated by more than a maximum allowable distance. The allowable separation distances are based on the normally expected symbol size.

The maximum-distance-between-centers test enables the separation of two symbols, such as T/ or R2, which happen to overlap because they are written close together. When symbols having a vertical stroke are written, the vertical frequently does not overlap the remainder of the symbol. The position test in the no-overlap case takes this into consideration. It is particularly useful when | | will be made an H or N by the next stroke, and also when two parts of the same symbol—e.g., the 1 and 3 of B—are just slightly disconnected.

The disadvantages of this scheme are that it may separate two parts of the same symbol which are not written close enough, and that it may not separate two different symbols which are written such that they overlap slightly. Its advantage is that it works about as well as the fixed-symbol-space scheme, yet does not *require* the display of symbol space separators. It uses the displayed symbols to cue the user about how large to write and about how far apart to place symbols. Since the symbols are displayed with fixed spacing, users tend to inadvertently imbed blank spaces unless they are provided with some strong indication of the location of the symbol spaces. This scheme therefore works best when the vertical bars are displayed.

## EVALUATION

One measure of the value of the recognition scheme is the accuracy with which it recognizes individual symbols. Since the goal of the scheme is to facilitate man-computer communication, a more direct measure of its value is the time required to perform some writing task on-line.

Three groups of users participated in the following two experiments. The first group consisted of six users who have had considerable experience with the symbol recognition scheme. The design of the decision-making scheme was, in fact, based on the handwriting of four of these users. The second group consisted of seven engineers and programmers who have had experience using the RAND tablet, but have not previously written with the symbol recognition scheme. The third group was made up of nine engineers, programmers, and secretaries who have had no previous experience with either the tablet or the recognition scheme.

### *Accuracy of Recognition*

Each user in the second and third groups participated in a half-hour training session prior to the testing phase of this experiment. The session began with an explanation of how to use the tablet, what symbols are recognized, the use of the scrub, etc. The user was then allowed to write whatever he wanted for 20 minutes—most users printed words, phrases, or the alphabet, usually repeating a symbol until it was recognized correctly. During the next 10 minutes, the user attempted to print each of the symbols while the author pointed out how he might achieve more success. The users in the first group did not receive any special training, since they were already familiar with the tablet and recognition scheme.

Since this experiment was concerned with *isolated* symbols, the decision-making scheme was adjusted (during the testing phase) to separate symbols on the basis of time only. Each user was instructed to carefully print the list of symbols five times, waiting for each symbol to be recognized before printing another. (The apostrophe was not included because the users were given no position cues, and therefore could not be expected to write an apostrophe different from a comma.) Each user thus printed each of 52 symbols five times; 53 responses were possible (including "cannot interpret"). All errors (including user errors such as writing 0 rather than ∅ for the letter O) were recorded.

The average recognition rate for the group with experience in using the symbol recognition scheme was 93% correct. The lowest "score" in this group was 90%; the highest, 96%. The average score among those experienced previously with only the RAND tablet was 88%, with a low score of 82% and a high score of 92%. The average score among those with no previous experience whatsoever was 87%, with a low of 81% and a high of 93%. The recognition rates were not only high, but quite uniform. In fact, the "scores" for users in the second and third groups are indistinguishable. The time it

takes to learn to use the hardware—with its separate tablet writing surface and CRT working surface—therefore appears to be less than one-half hour. The users in the first group probably scored somewhat higher than the others because they were more familiar with the scheme, and because it was designed to accommodate their writing styles.

Many of the errors were due to the misrecognition of (, ), [, and ]—(, for example, was frequently confused with 1, C, or L. The asterisk was also frequently missed because its shape is not well defined, and it therefore has many variations. Some users not familiar with coding form conventions found it difficult to learn to write the letter O with a slash to distinguish it from the number 0, yet learned to print Z with a crossbar, and I with upper and lower bars. The remainder of the errors were due to confusions between 7 and >, G and C, 5 and S, + and T, and a wide variety of other pairs of symbols.

*Ease of Operation*

Since this recognition scheme is designed for use in a problem-solving environment, an experiment was performed in which programmers used the scheme while solving simple coding problems. A user was given flow charts (Fig. 8), and was asked to write the corresponding computer code for one problem directly on-line (using only the recognition scheme) and that for the other off-line (using pencil and paper). The times required for solution—a measure of the usefulness of the recognition scheme versus pencil and paper—were recorded.

Since time was important, this experiment used the decision-making scheme which separates symbols according to identity and relative position as well as time. Users could therefore print quickly without pausing between symbols. Vertical bars were displayed to indicate symbol spacing. Users were allowed to make use of all editing facilities—i.e., change a symbol by writing over it; erase a symbol, space, or line with a scrubbing action; use the symbol $\wedge$ to insert symbols between others already printed on a line; and use the symbol > to obtain a blank line between two printed lines of text.

Eight programmers participated in this experiment. Four from the first group (described above) coded the problems in IBM System/360 Assembly Language. Each problem required the printing of approximately 200 symbols in this language. Two programmers each from the second and third groups coded the problems in FORTRAN—this required the printing of about 150 symbols for each problem. Half of the users in each group coded Problem A directly onto a coding form and Problem B directly on-line, using the tablet hardware and recognition scheme; the others coded Problem B onto a coding form and Problem A on-line.

Table 1. Times to Code the Problems of Figure 8

| Group | Problem | Coding Form Time | On-Line Time |
|---|---|---|---|
| 1 | A | 8¾ min | 12 min |
| | B | 9 " | 9½ " |
| 2 & 3 | A | 7 " | 12½ " |
| | B | 4 " | 9 " |

Table 1 summarizes the results of this experiment. Each entry is an average for two users. The inexperienced (second and third group) users spent a large part of their on-line problem-solving time trying to communicate with the recognition scheme. The experienced users, who had harder problems (assembly language versus FORTRAN) to solve, were not, however, slowed down very much by the recognition scheme.

The subjective findings of this experiment are, perhaps, more important than the numerical results. The accuracy of recognition here was much lower than in the previous experiment, partly because the scheme had to separate symbols geometrically, but also because the inexperienced users had forgotten some of what they had learned about the scheme, and because all of the users were less careful. All users found it difficult to print symbols as small and as closely packed as the hardware generated them—85 across a 10-in width. They tended to print oversize symbols, thus causing the scheme to introduce blank spaces or to recognize a single multiple-stroke symbol as two symbols.

The editing facilities helped to compensate for the difficulties with symbol recognition. The users found it convenient to change a symbol by writing directly over it, or to erase an entire line with a single scrubbing action—editing procedures which are much more difficult to do with pencil and paper than with the tablet hardware. The symbol $\wedge$ was frequently used to insert one or more symbols between two others. Although the symbol > was used on only two occasions to insert a new line between two others, this editing feature is potentially valuable for composing solutions to more difficult problems.

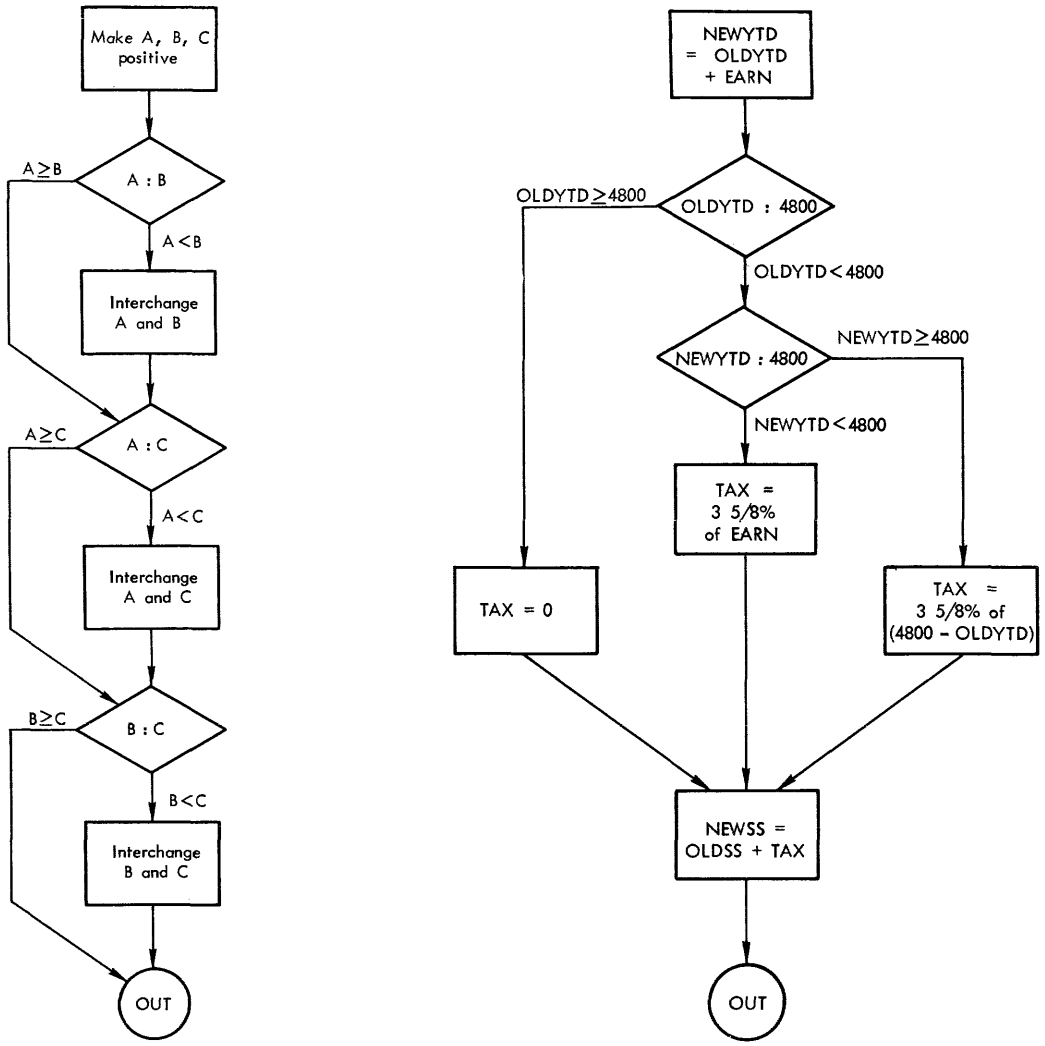


Figure 8. Flow charts of the problems coded in the experiment—(a) Problem A: a procedure for computing social security tax; (b) Problem B: a method of sorting three numbers into a descending sequence of their magnitudes.

Thus, this on-line recognition scheme with editing facilities appears to be a useful problem-solving aid, particularly as the users become more experienced, and the problems become more difficult. In a problem-solving situation, the editing facilities give the user much greater flexibility than pencil and paper. The accuracy of the recognition scheme is high enough that it is not distracting and the hardware is natural to use.

## DISCUSSION

This recognition scheme meets its primary objective of enabling any user to communicate naturally with a computer. A user is not distracted by any operational mechanics but, rather, may concentrate on his problem. All communications are made with a single device—a pen-like instrument. A user may write anywhere on a horizontal writing surface. The recognition scheme has been extended to recognize such flow-charting symbols as rectangles, circles, triangles, and diamonds. Thus, the user may draw free-form flowcharts as well as text, using only a pen.

The recognition program responds quickly and is efficient in storage. When the time-delay normally used to separate symbols is set to zero, the lifting of the pen and the display of a recognized symbol are apparently simultaneous. The recognition program—

including the data analysis and decision-making routines and data storage but not display or editing routines—requires about 2400 32-bit words of memory.

The major shortcoming of the present scheme is its difficulty in recognizing quickly written text: it has difficulty in separating overlapping symbols and in combining disconnected parts of a single symbol. Furthermore, since quickly written symbols tend to be distorted, they are misrecognized more often than carefully drawn symbols. These problems are due, in part, to the small size and close packing of the symbols. They can be relieved by allowing larger between-symbol spaces.

Another difficulty with the recognition scheme is that only a person familiar with the computer program can add a new symbol or new symbol description. Such changes are complicated because they frequently require several coding changes in the cross-linked tree structure of the decision-making scheme. A useful variation of this scheme would therefore be based on a data-dependent sequence of tests, but would permit automatic changes.

Finally, the advantages and disadvantages of pen-location-by-pen-location feature extraction should be clarified. This procedure is useful for real-time recognition because it minimizes the time delay between symbol completion and identification, yet produces a valuable set of features. It may, however, result in symbol variations not introduced by a scheme which extracts features after a symbol is completed. Such variations arise because some symbols may be drawn either clockwise or counterclockwise, portions of some symbols may or may not be retraced, some symbols may be constructed of various numbers and sequences of strokes, etc. Perhaps some other set of dynamically extracted features would prove as useful for discrimination as the present set without introducing as many variations.

## ACKNOWLEDGMENTS

I would like to thank T. O. Ellis, J. F. Heafner, and W. L. Sibley, all of The RAND Corporation, for many profitable discussions, and for designing and implementing the graphical hardware-software without which the recognition scheme could not exist.

## REFERENCES

1. M. E. Stevens, *Automatic Character Recognition—A State-of-the-Art Report,* National Bureau of Standards NBS Tech. Note 112 (May 1961).

2. E. E. David, Jr., and O. B. Selfridge, "Eyes and Ears for Computers," *Proc. IRE,* vol. 50, no. 5, pp. 1093–1101 (May 1962).

3. E. E. Graziano, *Automatic Pattern Recognition During the Period 1961–1962: An Annotated Bibliography,* Lockheed Missiles and Space Company, Sunnyvale, Calif. Report 6-90-63-16/SB-63-13 (May 1963).

4. E. C. Greanias, et al, "The Recognition of Handwritten Numerals by Contour Analysis," *IBM J. of Res. and Dev.,* vol. 7, no. 1, pp. 14–21 (Jan. 1963).

5. F. Kuhl, "Classification and Recognition of Hand-Printed Characters," *IEEE International Convention Record,* 1963, part 4, pp. 75–93.

6. T. Marill, et al, "Cyclops-1: A Second-Generation Recognition System," *AFIPS Conference Proceedings (FJCC),* vol. 24, Spartan Books, Baltimore, 1963, pp. 27–33.

7. H. A. Glucksman, "A Parapropagation Pattern Classifier," *IEEE Trans. on Elec. Computers,* vol. EC-14, no. 3, pp. 434–43 (June 1965).

8. T. L. Dimond, "Devices for Reading Handwritten Characters," *Proc. Eastern Joint Computer Conference,* Dec. 1957, pp. 232–37.

9. L. D. Harmon, "Handwriting Reader Recognizes Whole Words," *Electronics,* vol. 35, no. 34, pp. 29–31 (Aug. 24, 1962).

10. M. I. Bernstein, *Computer Recognition of On-Line, Hand-Written Characters,* The RAND Corporation, Santa Monica, Calif., RM-3753-ARPA (Oct. 1964).

11. R. M. Brown, "On-Line Computer Recognition of Handprinted Characters," *IEEE Trans. on Elec. Computers,* vol. EC-13, no. 6, pp. 750–52 (Dec. 1964).

12. W. Teitelman, "Real-Time Recognition of Hand-Drawn Characters," *AFIPS Conference Proceedings (FJCC),* vol. 26, part 1, Spartan Books, Baltimore, 1964, pp. 559–75.

13. M. I. Bernstein, *An On-Line System for Utilizing Hand-Printed Input,* System Development Corporation, Santa Monica, Calif., TM-3052, July 11, 1966.

14. M. R. Davis and T. O. Ellis, "The RAND Tablet: A Man-Machine Graphical Communication Device," *AFIPS Conference Proceedings (FJCC),* vol. 26, part 1, Spartan Books, Baltimore, 1964, pp. 325–31; also, The RAND Corporation, Santa Monica, California, RM-4122-ARPA (Aug. 1964).

15. H. Freeman, "On the Encoding of Arbitrary Geometric Configurations," *IRE Trans. on Elec. Computers,* vol. EC-10, no. 2, pp. 260–68 (June 1961).