Hypothesis Generation in a Computational Model for Visual Word Recognition

Jonathan J. Hull
State University of New York at Buffalo

Reading machines must convert pages of written text into a form that can be interpreted by a computer. Algorithms that carry out this process usually handle the task in either of two basic ways.[1] The first method is to segment words of text into isolated characters and recognize the individual characters. This technique is most successful if the input text is made up of characters that do not touch or overlap one another. However, the method is prone to errors when words cannot be segmented or when characters are broken because of noise in the image. The second, more general method for reading images of text is to recognize entire words as individual units. This method is more difficult because of the need to recognize a larger number of symbols, so it is typically used when it is difficult to segment a word into its constituent characters—for example, in cursive script.[2] The success of both techniques is considerable in some domains, but neither has resulted in algorithms that can read unconstrained text, instantaneously switching between text printed in different fonts and scripts, as a human reader can.[3]

To attain some of the human ability to recognize text printed in an almost innumerable variety of fonts and scripts, some researchers have adapted relevant portions of the human reading process to the development of reading algorithms. Shillman,[4] for example, determined the parameters of procedures for recognition of isolated characters from psychological experiments. The success of his methodology was demonstrated in experiments in which hand-printed examples of the letters *U* and *V* were distinguished by machine with even better accuracy than by human subjects.[5]

The approach discussed in this article also seeks to adapt relevant portions of the human reading process to the machine reading problem. An objective of this work is the development of a computational word recognition model that reflects the essence of the processes a person consciously and unconsciously executes while reading a running text. It is hoped that the text-reading methodology and algorithms discussed here will eventually possess human fluency. A complete solution to this problem will require many components, from input-level image processing to syntactic and semantic analyses, and expert levels of performance probably will not be achieved until most of these components are in place. The techniques described here, however, should provide the basis for continued development of these components.

The psychological investigation of the human reading process has been under way for over 100 years. One of the first observations made about the way people read was that our eyes do not move smoothly from left to right across a line of text. Rather, our eyes move in jumps from one fixation point to the next. It was also discovered that there are anywhere from 0 to 3 fixations per word. Other early work rejected the conjecture that we synthesize word identities from the recognition of individual characters; experiments demonstrated that human subjects recognized a four- or five-letter word in about the same amount of time it took to recognize an isolated character.[6] Subsequent work stressed the importance of the whole word to the recognition process and showed that important features are extracted on the word level and do not depend on the segmentation of a word into characters.[7] Contemporary theories of word recognition in fluent reading

0885-9000/86/0800-0063 $01.00©1986 IEEE

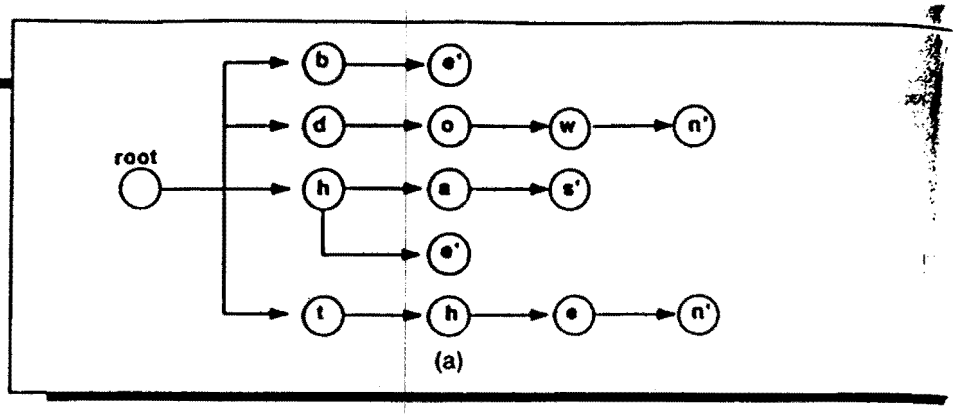


Figure 1. A trie for the dictionary *be, down, has, he, then* (a) and a letter-feature table (b). Terminal nodes in the trie are indicated by an apostrophe. The features for each letter are represented by the digits 0 through 5.

involve multiple, interacting processes that carry out tasks ranging from feature extraction to high-level goal analysis.[8] According to one theory, information at various distances from the current fixation point is processed in different ways as a person reads from left to right across a line of text.[9] Preliminary processing of words to the right of the current fixation point provides expectations about the identity of the initial letters in upcoming words. More precise visual processing at the current fixation is integrated with these expectations and with information from other sources to recognize the fixated word. This is similar to using information to the right of a fixation to generate hypotheses about the upcoming word and using information from a fixation to test the hypotheses.

Interesting parallels are apparent when the human reading process is compared to developments in algorithms for reading digital images of text. For example, the observation that fluent reading is not done on a character-by-character basis suggests that to achieve human competence in an algorithm, some word-level processing should take place before characters are identified. Also, an explicit identification of individual characters may not be an essential part of a reading algorithm. Another interesting observation is that the hypothesis generation and testing that is an implied part of some theories of human reading is similar to the pattern recognition technique of hierarchical classifier design.[10]

The strategy of hypothesis generation followed by hypothesis testing is adopted here as the basic design of an algorithm for reading digital images of text. The objective of hypothesis generation is to follow the human methodology, using features extracted from a word without first segmenting it into characters. These features suggest a *neighborhood*—a group of words that are hypotheses about the identity of the input word. These word hypotheses then drive a hypothesis-testing technique that matches the input to one of the hypotheses.[11] The result is a recognition of the input word image.

A neighborhood is a subset of a fixed vocabulary of words called the *dictionary* of the system. The method used to compute a neighborhood is similar to a previous technique for cursive script word recognition.[12] This article presents a generalized technique that uses sequences of discrete features and an efficient dictionary representation to compute neighborhoods. A statistical analysis determines the effect of any feature set on the performance of the technique, and a specific feature set for lowercase text is shown to produce good results. Image-processing experiments further demonstrate the feasibility of this feature set. The extension of this technique to uppercase and mixed-case text is discussed elsewhere.[13]

Neighborhood computation

The neighborhood of an input word is computed by a two-step process of feature extraction followed by dictionary lookup. The feature extraction phase determines a small number of discrete features that are easy to extract from an image, ensuring efficiency and accuracy. The features should be common to as many printed font styles as possible so that a wide variety of text can be read. The output of feature extraction is the left-to-right sequence of features present in an input word.

The dictionary lookup process uses the sequence of features output by the feature extraction phase to determine the neighborhood of words with the same sequence of features. A letter tree, called a *trie,* is used for this purpose because of its flexibility and the natural way that it can take advantage of the sequential nature of the input. A trie is a data structure that is essentially a finite automaton representation for a fixed dictionary of words. The usefulness of the trie has been demonstrated in algorithms for cursive script recognition[14] and contextual postprocessing.[15]

An example of a trie for the dictionary *be, down, has, he, then* is shown in Figure 1a and a data structure called the *letter-feature table* is shown in Figure 1b. Each letter in the input alphabet (the lowercase letters *a* to *z* in this example) is described by a sequence of features in the letter-feature table. The features shown in Figure 1b are numbered with the digits 0 through 5. The specific meaning of such features will be made clear later. The letter-feature table is used during the lookup process to map the features detected in an image onto nodes in the trie. The lookup process uses a simple postorder traversal in combination with the letter-feature table to return all the words that match the input feature string. These words are the neighborhood of the input feature string. For example, if the string of features numbered 2110 is given to the lookup process, and the trie and the letter-feature table in Figure 1 are used, the neighborhood containing *he* and *be* is returned.

Statistical evidence

The potential usefulness of the neighborhood computation as a generator of hypotheses is measured by the number of dictionary words in the neighborhood of an input word. Ideally, the neighborhood of any input word should contain a small number of dictionary words, to simplify the work of any subsequent recognition procedure. This goal must be

ter	features	letter	feature	letter	features
a	01	j	5	s	0
b	21	k	21	t	1
c	10	l	2	u	11
d	12	m	111	v	0
e	10	n	11	w	0
f	20	o	11	x	0
g	11	p	31	y	0
h	21	q	13	z	0
i	4	r	10		

(b)

balanced against the feature detection requirement of a small number of features that occur in many fonts. The following study uses five statistics to project the performance of different aspects of the neighborhood calculation technique when input words are drawn from a large corpus of text. The statistics and the methodology of the study apply to text printed in any font, as well as handwritten script; however, results are presented only for lowercase printed text.

Definitions of statistics. For a given set of features numbered 0, 1, . . . , $n-1$, the *shape number* of a word is defined as the concatenation of the numbers of the left-to-right sequence of features that occur in the word. If n is greater than 10, a separator is placed between feature numbers to preserve uniqueness.

The *neighborhood size* (ns) of a dictionary word is the number of dictionary words with the same shape number. If $s(w_i)$ is a function that computes the shape number of the ith dictionary word, and the following predicate is defined:

$$eqshape\ (w_i, w_j) = \begin{cases} 1 \text{ if } s\ (w_i) = s\ (w_j) \\ 0 \text{ otherwise,} \end{cases}$$

then the neighborhood size of the ith dictionary word w_i is

$$ns\ (w_i) = \sum_{j=1}^{N_d} eqshape\ (w_i, w_j),$$

where N_d is the number of words in the dictionary.

We measure the average neighborhood size (ANS) by partitioning the dictionary into neighborhoods that contain words with the same shape number. If N_n is the number of neighborhoods that occur in a given dictionary, if NS_i is the number of words in the ith neighborhood, and if N_d is the number of words in the dictionary, then ANS is computed as

$$ANS = \frac{\sum_{i=1}^{N_n} NS_i}{N_n} = \frac{N_d}{N_n}$$

This statistic measures the number of words that can be expected to occur in a neighborhood in the dictionary.

The average neighborhood size per dictionary word (ANS_d) is given by the following formula:

$$ANS_d = \frac{\sum_{i=1}^{N_d} ns\ (w_i)}{N_d}$$

This quantity measures the expected number of words output by the dictionary under the assumption that there is a uniform probability of encountering any word contained therein.

Another statistic that considers the frequency of occurrence of words in the source text is the average neighborhood size per text word (ANS_t), which is computed by

$$ANS_t = \frac{\sum_{i=1}^{N_t} ns\ (tw_i)}{N_t} = \frac{\sum_{i=1}^{N_d} ns\ (w_i) * n\ (w_i)}{N_t}$$

where $ns(tw_i)$ is the neighborhood size of the ith word in the running text, $ns(w_i)$ is as defined before, $n(w_i)$ is the number of times the ith dictionary word occurs in the source text, and N_t is the total number of words in the source text. For example, if an entire source text consisted of the phrase "never say never," N_t would be 3, and N_d would be 2; tw_1 = "never," tw_2 = "say," and tw_3 = "never." Also, $ns(tw_1) = ns(tw_2) = ns(tw_3) = 1$, and $ANS = ANS_d = ANS_t = 1$. ANS_t measures the projected performance of the neighborhood calculation as it affects the hypothesis-testing portion of the computational model. If the probability of encountering any word image is proportional to the number of times that word occurs in the text, then this quantity expresses the average neighborhood size that will be encountered if a text is processed in a word-by-word fashion by a reading algorithm.

These three measures of average neighborhood size illustrate different aspects of the performance of the neighborhood calculation. ANS is a static measure that gives information about the breakup of the dictionary but does not account for any dynamic characteristics of the procedure. ANS_d is a dynamic measure of the "stress" expected to be placed on a dictionary structure during the running of the algorithm, but it assumes only a uniform probability of encountering any word and thus embodies little knowledge about the source text. ANS_t is a dynamic measure of the average number of words that would be presented to an algorithm carrying out complete recognition of the source text. As such it is the most interesting of the three since it best shows the effect of the neighborhood computation routine on the overall reading algorithm.

Two other statistics that project the performance of various aspects of this algorithm are the percentage of the dictionary uniquely specified by shape and the percentage of text uniquely specified by shape. These are the percentages of words in the dictionary and text that fall in a neighborhood containing only those words. These statistics indicate how much of the dictionary and text are recognized by the neighborhood calculation alone. In the "never say never"

example, 50 percent of the dictionary and 33 percent of the text are uniquely specified by shape.

Experiment on lowercase text. In an experimental study the five statistics defined above were determined from a large body of text known as the Brown corpus.[16] Containing over 1,000,000 words (500 sample passages of about 2000 words each), the text was designed by a group of linguists at Brown University to represent edited modern American English. For this study the entire corpus was converted to lowercase and six features (listed here by their numerical representations) were used to define the shape of each word:

(0) A significant area at the beginning or end of a word filled with one or more nonvertical strokes (e.g., the area to the right of the vertical part in a *c*).
(1) A short, vertical part (e.g., the leg of an *r*).
(2) A long, high, vertical part that extends above the main body of the word (e.g., the ascender portion of a *b*).
(3) A long, low, vertical part that extends below the main body of the word (e.g., the descender in a *p*).
(4) Dots over short, vertical parts (occurs in an *i*).
(5) Dots over long, vertical parts (occurs in a *j*).

These features were selected because of their simplicity and the relative ease with which they can be computed from the image of a word. Many other features could be used, but they would probably not be as simple and easy to determine as those listed above. For example, the holes that occur in *a, b, d, e, g, o, p,* and *q* are an obvious feature to consider, but their detection in the presence of image noise such as broken or touching characters is notoriously difficult. Another feature that could be used is the slanted parts that occur in *k, v, w, x, y,* and *z*. To simplify feature analysis, this feature was not used.

The *shape number*—that is, the left-to-right sequence of feature numbers—for each character is shown in the letter-feature table in Figure 1. The shape number of a word is constructed by appending the shape numbers of its characters, deleting any zero that appears between two non-zeros, and retaining at most one zero at the beginning or end. For example, the shape number of *dog* is 121111, the shape number of *cat* is 111, and the shape number of *tie* is 1410. The zeros are deleted in the middle and retained at the ends because it is easier to recognize filled spaces at the ends of words.

The Brown corpus was designed to be representative of the written English language; thus experimental results derived from it should be extensible to a more general situation. The corpus is divided into 15 subject or genre categories:

(A) Press reportage
(B) Press editorial
(C) Press reviews
(D) Religion
(E) Skills
(F) Popular lore
(G) Belles lettres
(H) Miscellaneous
(J) Learned
(K) General fiction
(L) Mystery/detective fiction
(M) Science fiction
(N) Adventure/western fiction
(P) Romance/love story
(R) Humor

The five statistical measures defined earlier were computed for the entire corpus as well as for each of its 15 genres, thus providing information about the projected performance of the neighborhood computation technique in a general situation as well as in situations more like a typical reading environment, where text covers a single general subject. The results of this experiment are presented in Table 1, which shows that 28 percent of the dictionary and 9 percent of the entire corpus are uniquely specified by shape. The value of ANS is only 2.5 over the whole text, and ANS_d and ANS_t are 22.9 and 38.4, respectively. When measured over the genres, the smallest ANS_d value is 4.2 and the highest is only 10.8. This trend is repeated in the ANS_t figure, where the smallest value is 5.8 and the highest is 16.4. Thus, restricting the subject matter of the source text reduces the values of ANS_d and ANS_t by about half what they are in the entire corpus.

The same statistical measures were also computed over subsets of the corpus that contained words with the same number of characters. The results, shown in Table 2, indicate the potential performance of the neighborhood computation technique under the assumption that the number of characters in a word can be accurately estimated. (Note that this assumption can be different from the assumption that a word can be segmented into characters.) Table 2 shows that the highest values for ANS, ANS_d, and ANS_t are now 3.7, 16.0, and 15.6, whereas their previous values were 2.5, 22.9, and 38.4. Therefore, the average number of words presented to any subsequent recognition routine by the neighborhood computation is reduced by a factor of more than two when the number of letters in the input word is known. The table also shows the effect of estimating the number of letters in an input word when words of length greater than or equal to 14 characters are considered. Here, more than 95 percent of all words are uniquely specified by shape.

Experimental simulation

A series of experiments was carried out on digital images of lowercase words to show that the features used to define neighborhoods in the statistical study could be computed from words printed in a variety of fonts. The objective was to extrapolate the results of the statistical study to real images of text. Although further development of the image-processing portions of the algorithm might be necessary to realize a commercially viable algorithm, the experiments demonstrate the feasibilty of such an endeavor.

The image database for these experiments contained 10 complete 24-point font samples (see Figure 2 for examples from each font) that were digitized on a laser scanner at a resolution of 500 binary pixels per inch. These data were segmented into characters and stored in individual files. Words

Table 1. Summary of the projected performance of a procedure that computes the shape of lowercase words. Results are broken down in terms of the genres as well as the entire text of the Brown corpus.

Genre	N_d	N_t	% dict. unique	% text unique	ANS	ANS_d	ANS_t
A	11,807	88,051	35	19	2.0	10.1	14.3
B	8,653	54,662	39	20	1.9	7.6	11.6
C	7,751	35,466	41	24	1.8	7.2	9.9
D	5,733	34,495	47	25	1.7	4.9	7.1
E	9,846	72,529	37	21	2.0	8.6	12.9
F	12,678	97,658	36	19	2.0	9.6	14.6
G	16,218	152,662	36	15	2.0	10.8	16.4
H	6,692	61,702	40	23	1.8	5.3	8.0
J	14,244	160,920	39	21	1.9	9.0	13.4
K	8,554	58,650	37	18	2.0	7.9	12.1
L	6,315	48,462	37	19	1.9	6.7	9.9
M	3,001	12,127	48	30	1.6	4.2	5.8
N	8,065	58,790	34	18	2.1	8.3	12.2
P	7,713	59,014	37	19	2.0	7.9	12.3
R	4,693	18,447	45	30	1.7	5.0	7.4
Corpus	43,264	1,013,549	28	9	2.5	22.9	38.4

Table 2. Summary of the projected performance of a procedure that computes the shape of lowercase words as a function of word length. The input texts are subsets of the Brown corpus that contain words of the same length.

w1	N_d	N_t	% dict. unique	% text unique	ANS	ANS_d	ANS_t
1	26	33,244	35	91	1.9	3.0	1.3
2	253	172,064	5	7	3.0	6.1	5.6
3	1,028	215,282	13	16	3.7	12.2	11.1
4	2,624	161,528	13	5	3.7	15.1	15.6
5	4,478	112,229	17	17	3.2	16.0	14.9
6	6,132	86,432	25	23	2.5	10.9	11.3
7	7,034	79,119	35	29	2.0	6.9	7.8
8	6,430	56,413	48	44	1.6	3.9	4.1
9	5,347	40,053	61	56	1.4	2.4	2.5
10	4,032	26,959	72	69	1.2	1.7	1.8
11	2,557	14,927	82	79	1.1	1.3	1.4
12	1,578	8,137	88	85	1.1	1.2	1.3
13	908	4,208	93	88	1.0	1.1	1.2
14	455	2,055	96	95	1.0	1.0	1.1
15	216	589	99	98	1.0	1.0	1.0
16	83	164	100	100	1.0	1.0	1.0
17	48	71	96	97	1.0	1.0	1.0
>=18	35	75	100	100	1.0	1.0	1.0

were then generated by appending the appropriate character images. Although these fonts are a small subset of the thousands of fonts available for typesetting and they are all the same point size, they vary enough in visual characteristics to test the performance of the neighborhood calculation. For example, there is wide variation in size within the 10 fonts: height ranges from 154 pixels for the Caslon font to 216 pixels for the Bembo font; width, as measured by the *w*, varies from 94 pixels for the Caslon Antique to 161 pixels for the Americana. Stroke thickness also varies widely: the thickness of the long, high, vertical stroke in the *b* ranges from 10 pixels in Americana to 20 pixels in Bodoni Bold. Considered together, these are size variations (from the minumum value) of 40 percent in height, 71 percent in width, and 100 percent in stroke thickness.

The shape number of the image of a word was determined by the following algorithm:

(1) Adjust for variation in stroke thickness by thinning or thickening until the width of the thickest stroke is within a tolerance. The width of the thickest stroke is determined from measurements on the vertical density projection histogram. The tolerance is a range about the mean thickness of the widest strokes in the 10 fonts.

(2) Find all dots.

(3) Find all vertical parts and classify each as a spurious response; a short, vertical part; a long, high, vertical part; or a long, low, vertical part.

(4) Match any dots determined in step 2 with the nearest vertical part. If that vertical part was classified as a short, vertical part, then classify the dot and this vertical part as an *i*. If that vertical part was classified as a long, low, vertical

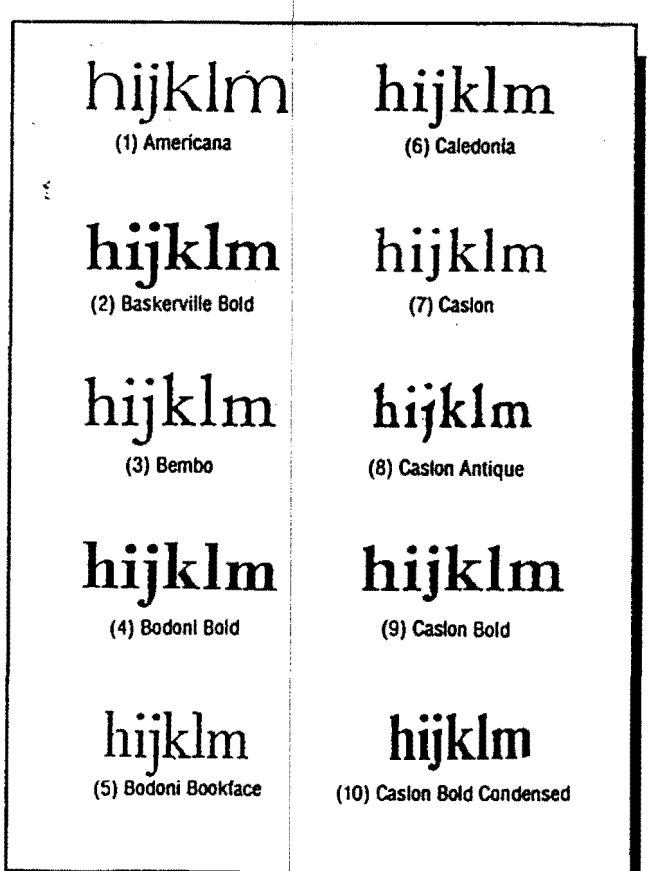
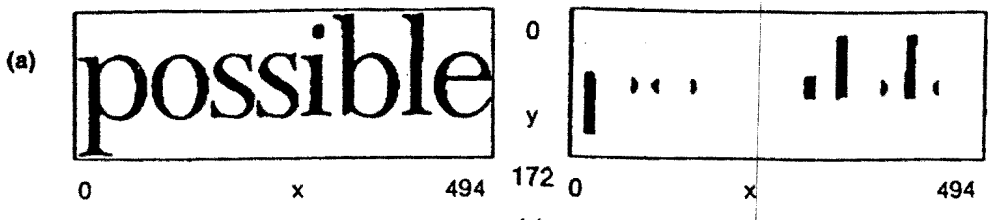


Figure 2. Examples of characters from each font used in the recognition tests.



(c)

comp.	height	width	size	min. y prop.	max. y prop.	y of cog	y of coa	dist. from cog to coa	hgt*wid size	P2A
vert 1	75	14	1028	0.42	0.85	109	109	1.0	22	2.3
vert 2	21	8	116	0.45	0.56	87	87	2.0	52	1.9
vert 3	20	8	112	0.45	0.56	87	88	1.0	48	1.8
vert 4	17	7	97	0.45	0.56	86	87	1.4	36	1.7
vert 5	29	14	392	0.42	0.58	86	86	1.0	14	1.4
vert 6	78	14	1047	0.14	0.59	61	63	2.2	45	2.5
vert 7	21	8	114	0.45	0.56	87	87	2.0	54	1.9
vert 8	78	15	1041	0.13	0.58	62	62	1.0	129	2.5
vert 9	20	7	106	0.45	0.56	87	88	1.0	34	1.8

(d) 3111421210

Figure 3. Program operation example: (a) an input word in the Caledonia font; (b) the vertical bars in the input word that are detected by the convolution; (c) table of features used to differentiate the four classes of vertical parts; (d) the shape number for the input word. Notation in the feature table: *min. y prop.* and *max. y prop.* are the minimum and maximum y locations on a scale of zero to one for each component; *y of cog* and *y of coa* denote the actual y values of the center of gravity and the center of area; and *dist. from cog to coa* is the Euclidean distance between the center of gravity and the center of area.

part, then classify the dot and the vertical part as a *j*.

(5) Determine if there is a significant area at the beginning or end of the word that contains one or more nonvertical strokes.

The thinning in step 1 is done with a standard, pixel-based thinning algorithm.[18] The thickening method changes any white pixel to black if the white pixel is connected to a black component. This is repeated until the desired thickness is reached. The location of dots in step 2 is done by finding large (greater than 60 pixels), connected, black components in the upper third of the image. Step 4, the matching of dots with vertical parts, is done by measuring the horizontal distance between the centers of area of a dot and the vertical parts in the image. The vertical part closest to the dot is the one matched to it. If the dot is beyond a threshold distance from any vertical part, it is classified as a spurious response. The determination of significant areas containing nonvertical strokes at the beginning of a word in step 5 is done by thresholding the horizontal distance from the first black pixel in the original image to the first vertical part. The value of the threshold depends on the width of the thickest stroke in the original image. An analogous process is applied at the end of a word.

Step 3, the location and classification of vertical parts, has two stages. The first stage locates the vertical parts by convolving the input image with a vertical bar mask (55 rows by 1 column) and thresholding the output (at a value of 22). This operator effectively smooths the image and removes many of the slanted portions of letters such as *x*. The classification stage determines whether a vertical part is a spurious response; a short, vertical part; a long, high, vertical part; or a long, low, vertical part.

The classification of vertical parts is carried out with a polynomial discriminant function.[19] Ten features were calculated for each connected component: height, width, size (in pixels), minimum *y* value as a proportion of height (or min. *y* prop.), maximum *y* value as a proportion of height (or max. *y* prop.), the *y* value of the center of gravity, the *y* value of the center of area, the Euclidean distance between the center of gravity and the center of area, the difference between the size and the product of the height and width, and perimeter^2/4 * π * size (called P2A). Eight combinations of these features were also used: P2A^2, P2A^3, P2A * height, P2A * width, P2A * size, P2A * min. *y* prop., P2A * max. *y* prop., and size^2. P2A is a shape measurement that helps discriminate between spurious responses and vertical parts. P2A was used as a feature after it was observed that a large number of spurious responses had a shape distinctly different from the shape of vertical parts.

The coefficients of the classifier were learned during a training phase. The data for this phase were the components that result from convolving the isolated characters in each font with the 55 × 1 mask. The classifier was tested on the training data and yielded a correct classification of 98 percent of the 26 × 10 = 260 input characters.

It is interesting to note that even though this is a high rate of performance, the actual recognition rate for words may vary widely depending on how many words contain the erroneously classified characters.

Figure 3 shows an example of the image processing carried out by this algorithm. Figure 3a shows the original image of the word *possible*. Figure 3b shows the thresholded output produced by the algorithm after convolution with the vertical mask. The complete set of features for each component in Figure 3b is given in Figure 3c. The shape number for *possible* shown in Figure 3d is computed from the information in Figure 3c by the classification algorithm.

Table 3. Accuracy of the shape number computation for the 630 words in sample G60 of the Brown corpus.

Font	Characters not touching		Characters touching	
	% correct	% error	% correct	% error
Americana	99.8	0.0	84.2	5.2
Baskerville Bold	100.0	0.0	93.0	3.1
Bembo	100.0	0.0	86.0	2.0
Bodoni Bold	100.0	0.0	99.0	0.2
Bodoni Book	99.7	0.3	79.6	7.4
Caledonia	100.0	0.0	75.0	8.2
Caslon	100.0	0.0	80.3	4.6
Caslon Antique	100.0	0.0	94.3	2.5
Caslon Bold	99.4	0.3	96.5	1.7
Caslon Bold Condensed	93.7	4.6	89.8	1.5

Several experimental runs were made to test the performance of the image-processing and classification algorithms. One of the 500 samples of text in the corpus was randomly chosen and used to generate images of words. The sample, number 60 in genre G (*belles lettres*), is a 2003-word excerpt from *Molly and Me* by G. Berg and C. Berg. The dictionary of this sample contains 630 words. Although this is a small sample, 54 percent of the words in the running text of the entire corpus occur among the 630 words in the dictionary.

Test images were generated from the 630 words in two ways. In the first method the appropriate character images were appended and the characters were not allowed to touch. This method is designed to test the general-purpose performance of the recognition algorithm on good-quality input. In the second method the character images were appended and moved horizontally until their black portions touched. This tests performance under a condition easy for humans to compensate for but difficult for a recognition algorithm that requires topologically distinct characters.

The results of these tests are presented in Table 3. Under the first input condition, a better than 99 percent correct classification rate was achieved in 9 out of 10 cases. The Caslon Bold Condensed font has a 93.7 percent correct classification rate. When the second condition (characters touching) was tested, the best performance was 99 percent for the Bodoni Bold and the worst-case performance was 75 percent for the Caslon. This indicates several areas for improvement. Many of the errors were due to the erroneous detection of slanted parts, which could be corrected if slanted parts as well as vertical parts were detected. Other errors could be corrected in the trie search if the tests performed on a component were specialized on the basis of the letter in which the component could occur. For example, one set of tests could be used if the component was probably a spurious response in a *w*; another set could be used if the component was a short, vertical part in an *n*.

An examination of the errors committed in the experiment on touching characters showed that many of the vertical parts were misclassified as spurious responses. The proximity of other letters caused the convolution to produce components less rectangular than the components used in developing the classifier. The problem could be corrected by training the classifier on such components or by designing a classifier less dependent on measures of component shape than this one.

The statistical study and experimental simulation show that the neighborhood calculation technique described here has an initial level of feasibility. Experiments with more fonts and further work on the image-processing routines are needed to achieve commercially valid levels of performance.

Extensions of the neighborhood calculation technique are under way, including the incorporation of uppercase and mixed-case text. This requires the definition of a feature set for the uppercase alphabet and the development of image-processing routines to detect those features, coupled with a preprocessing routine that locates uppercase letters in the input image and detects uppercase features at the indicated locations. A similar procedure could be used for italics. Statistical studies of an uppercase feature set have shown the feasibility of computing neighborhoods of uppercase and mixed-case text.[13]

The use of the neighborhood calculation technique on handwritten cursive script would require the definition of a feature set that could be extracted from a wide variety of scripts. Features such as ascenders and the number of cross-

ings of a line through the middle of a word have been used for this purpose.[12] It remains to be seen how well these features would perform on cursive script from a large number of writers. However, if such features can be computed with enough accuracy, the statistical analysis presented here could be used to project performance in this domain.

The work discussed in this article is the first step in the development of a computational model for visual word recognition that possesses the human adaptability to different input formats. A complete methodology, which will incorporate this work in the first stage of its processing, is now being developed. Other portions of the algorithm will use the neighborhood of an input word to direct the detailed analysis of an image. The use of constraints between words of text is also under study.[20]

Acknowledgments

The author greatly appreciates valuable discussions with Radmilo Bozinovic as well as the advice and guidance of Sargur Srihari and the comments of Don D'Amato, Anthony Ralston, Stuart Shapiro, John Tan, and Deborah Walters. He also gratefully acknowledges the support of the United States Postal Service under the BOA program.

References

1. J. R. Ullmann, "Advances in Character Recognition," in *Applications of Pattern Recognition*, K. S. Fu, ed., CRC Press, Boca Raton, Fla., 1982, pp. 197-236.
2. R. F. H. Farag, "Word-Level Recognition of Cursive Script," *IEEE Trans. Computers*, Vol. 28, 1979, pp. 172-175.
3. J. Schurmann, "Reading Machines," *Proc. Sixth International Conf. Pattern Recognition*, Munich, W. Germany, Oct. 1982, pp. 1031-1044.
4. R. J. Shillman, *Character Recognition Based on Phenomenological Attributes: Theory and Methods*, PhD dissertation, Massachusetts Institute of Technology, Aug. 1974.
5. C. Y. Suen and R. J. Shillman, "Low Error Rate Optical Character Recognition of Unconstrained Handprinted Letters Based on a Model of Human Perception," *IEEE Trans. Systems, Man, and Cybernetics*, June 1977, pp. 491-495.
6. J. K. Cattell, "The Time It Takes to See and Name Objects," *Mind*, Vol. 11, 1886, pp. 63-65.
7. R. N. Haber and L. R. Haber, "Visual Components of the Reading Process," *Visible Language*, Vol. XV, No. 2, 1981, pp. 147-181.
8. M. M. Taylor, "The Bilateral Cooperative Model of Reading: A Human Paradigm for Artificial Intelligence," in *Artificial and Human Intelligence*, A. Elithorn and R. Banerji, eds., North-Holland, 1984.
9. K. Rayner, G. W. McConkie, and D. Zola, "Integrating Information Across Eye Movements," *Cognitive Psychology*, Vol. 12, 1980, pp. 206-226.
10. J. Schuermann and W. Doster, "A Decision Theoretic Approach to Hierarchical Classifier Design," *Pattern Recognition*, Vol. 17, No. 3, 1984, pp. 359-369.
11. J. J. Hull and S. N. Srihari, "A Computational Approach to Visual Word Recognition: Hypothesis Generation and Testing," *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, Miami Beach, Fla., June 1986, pp. 156-161.
12. L. D. Earnest, "Machine Recognition of Cursive Writing," *Information Processing 1962, Proc. IFIP Congress 62*, Munich, W. Germany, Aug. 27-Sept. 1, 1962, pp. 462-466.
13. J. J. Hull, "A Computational Theory of Visual Word Recognition," Technical report, SUNY at Buffalo, Dept. of Computer Science, 1986.
14. R. Bozinovic and S. N. Srihari, "Knowledge-Based Cursive Script Interpretation," *Proc. Seventh International Joint Conf. Pattern Recognition*, Montreal, Canada, July 30-Aug. 2, 1984, pp. 774-776.
15. S. N. Srihari, J. J. Hull, and R. Choudhari, "Integrating Diverse Knowledge Sources in Text Recognition," *ACM Trans. Office Information Systems*, Vol. 1, No. 1, Jan. 1983, pp. 68-87.
16. H. Kucera and W. N. Francis, *Computational Analysis of Present-Day American English*, Brown University Press, Providence, R.I., 1967.
17. R. Shinghal and G. T. Toussaint, "A Bottom-up and Top-down Approach to Using Context in Text Recognition," *International J. Man-Machine Studies*, Vol. 11, 1979, pp. 201-212.
18. T. Pavlidis, *Algorithms for Graphics and Image Processing*, Computer Science Press, Rockville, Md., 1982.
19. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Addison-Wesley, New York, 1973.
20. J. J. Hull, "Inter-word Constraints in Visual Word Recognition," *Proc. Sixth Canadian Conf. Artificial Intelligence*, Montreal, Canada, May 21-23, 1986, pp. 134-138.

Jonathan J. Hull is a research associate in the computer science department of the State University of New York at Buffalo, from which he received the BA degree in computer science and statistics in 1980 and the MS degree in computer science in 1983. He expects to receive the PhD in computer science in September 1986.

Hull's address is State University of New York at Buffalo, Dept. of Computer Science, 226 Bell Hall, Buffalo, NY 14260. His ARPAnet address is hull%buffalo-cs@CSNET-RELAY.ARPA.