HYPOTHESIS TESTING IN A COMPUTATIONAL THEORY OF VISUAL WORD RECOGNITION

Jonathan J. Hull

Department of Computer Science
State University of New York at Buffalo
Buffalo, New York 14260
hull@cs.buffalo.edu

ABSTRACT

A computational theory of reading and an algorithmic realization of the theory is presented that illustrates the application of the methodology of a computational theory to an engineering problem. The theory is based on past studies of how people read that show there are two steps of visual processing in reading and that these steps are influenced by cognitive processes. This paper discusses the development of a similar set of algorithms. A gross visual description of a word is used to suggest a set of hypotheses about its identity. These then drive further selective analysis of the image that can be altered by knowledge of language characteristics such as syntax. This is ***not*** a character recognition algorithm since an explicit segmentation of a word and a recognition of its isolated characters is avoided. This paper presents a unified discussion of this methodology with a concentration on the second stage of selective image analysis. An algorithm is presented that determines the minimum number of tests that have to be programmed under the constraint that the minimum number of tests are to be executed. This is used to compare the proposed technique to a similar character recognition algorithm.

1. Introduction

The fluent reading of text by computer without human intervention remains an elusive goal of Artificial Intelligence research. Fluent reading is the transformation of an arbitrary page of text, that could contain a mixture of machine-printed, hand-printed, or handwritten text, from its representation as a two-dimensional image into a form understandable by a computer, such as ASCII code. The current lack of a technique with these capabilities is interesting in light of the relative ease with which people read and the many years of investigation into computer reading algorithms, the methods people use to read text, and the long history of Artificial Intelligence research into computer vision [12].

The parallel between algorithms for reading text and explanations for human performance is most interesting. With some notable exceptions, most reading algorithms use a *character recognition* approach in which words are segmented into isolated characters that are individually recognized. For these algorithms reading is equivalent to a sequence of character recognitions.

The way people read is significantly different from character recognition. We bring to reading a wealth of information about the world and expectations about what we will read. This is mixed with knowledge about how text is arranged on a page, knowledge of the syntax and semantics of language, and visual knowledge about letters and words. The recognition processes that take place during fluent reading use visual information from much more than just isolated characters. Whole words or groups of characters are recognized by processes, that in some cases, do not even require detailed visual processing. This is because fluent human reading uses many knowledge sources to develop an understanding of a text while it is being recognized. This integration of understanding and recognition is responsible for human performance in fluent reading.

The fact that few reading algorithms have utilized the many disparate knowledge sources or the recognition strategy of a human reader might explain the gap between the reading proficiency of algorithms and people. Although some character recognition techniques have been augmented with knowledge about words, no reading algorithm has been proposed that fully utilizes the sorts of knowledge routinely employed by a human reader [11]. Such an algorithm would have the potential of yielding substantial improvements in performance.

2. A Computational Theory and Algorithm for Reading

The mechanism of a computational theory and its algorithm are chosen as the vehicle for the present investigation of reading because reading is an information processing task to which this mechanism applies [9]. The proposed computational theory of reading is based on previous studies of human performance. It shows *what* is computed by people when they read, *why* this is important, and general guidelines of how this should be carried out. Since reading is a complex information processing task involving interactions of knowledge from many different sources, algorithms are developed that implement only a subset of these interactions. However, these algorithms are sufficient to illustrate that if the complete version of the theory were implemented, a robust "reading machine" would result.

The computational theory of reading proposed here is derived from work on human reading that includes studies of human eye movements [10]. To a person who reads a line of text, it seems to them as if their eyes move smoothly from left to right. However, this is not completely true. In reality, our eyes move in ballistic jumps called *saccades* from one *fixation point* to the next. During a saccade the text is blurred and unreadable. (This is not apparent to the reader.) Therefore, most of the visual processing of reading takes place during the fixations. Usually there are about one to three fixations near the beginning of the word. However, interestingly enough, some words are never fixated. The sequence of fixations is approximately from left to right across a line of text, however, regressions do occur frequently. Figure 1 shows the sequence of fixations in a line of text [1].

There are two types of visual processing in reading. In the first type of processing, information from peripheral vision provides a gross visual description of words to the right of the current fixation point. This information is used to form expectations about the words. The second stage of processing occurs on a subsequent fixation when these expectations are integrated with other visual information.

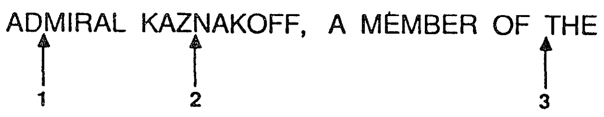

Figure 1. Sequence of fixations in a line of text [1].

The visual processing is influenced by many high-level factors that include the reason a person is reading the passage of text, as well as the familiarity of the reader with the subject and his or her skill level. A more skilled reader uses visual information more economically than a less skilled one [3]. That is, a skilled reader uses less visual processing than a poor reader. Recent work has also shown that syntactic processing also influences the visual processing of a text [2].

The proposed computational theory of reading contains three stages that are similar to those of human reading. The first stage generates hypotheses about words from a gross visual description. This is similar to the visual processing of words to the right of a fixation point and is an essential component of one theory of human reading [4]. The second stage uses these hypotheses to determine a feature testing sequence that can be executed on the image to recognize the word. This sequence is adaptable to different high-level influences and can be executed at different physical locations in the word. This stage is similar to the detailed visual processing that takes place at a fixation. The third stage of the theory concerns high-level processing. This stage captures the influence of the various non-visual processes that influence reading such as syntax and semantics. These processes remove word-hypotheses from consideration that do not agree with the high-level constraints. This is a way to represent the influence of many high-level knowledge sources.

The remainder of this paper discusses algorithms that implement two of the three stages outlined above. A hypothesis generation procedure is briefly presented. A global contextual analysis procedure is not fully discussed here. Instead, the reader is referred to a technique that uses knowledge about transitions *between* words to improve the performance of the hypothesis testing portion of this algorithm [5]. A hypothesis testing component is fully presented and an algorithm is discussed that determines the minimum number of feature tests needed by this component.

3. Hypothesis Generation

The hypothesis generation component of the algorithm uses a description of the gross visual characteristics of a word image to index into a dictionary and retrieve a subset of words called a *neighborhood* that have the same description. The description is the left-to-right sequence of occurrence of a small number of features. The features are simple and easy to extract to increase the reliability of the technique in the presence of noise. This approach is suitable for generating hypotheses about an input word since a small number of features can partition a large dictionary into a limited number of small neighborhoods [6]. This is less error-prone than using many features to carry out complete recognition.

4. Hypothesis Testing

The hypothesis testing component of the algorithm uses the words in a neighborhood to determine a feature testing sequence. The testing sequence is structured as a tree that specifies the order and locations in which tests are performed. The result of a test determines successive tests and reduces the number of words that could match the input image.

It is assumed that the input image contains a word in the neighborhood. This constrains the features that could occur at locations in the image. If a set of features are defined a-priori, this constraint determines the subset that could be present. The features used in this paper for hypothesis testing are shown below:

feature code	description
E	empty space;
1	closed at both the top and bottom, e.g. "o";
2	closed at the top, e.g. "n";
3	closed at the bottom, e.g. "u";
4	left of a short vertical bar in an "a"
5	right of a short vertical bar in a "c"
6	right of the short vertical bar in "e"
7	right of a long vertical bar in an "f"
8	between two short vertical bar in a "g"
9	right of a long vertical bar in a "k"
10	right of a short vertical bar in an "r"
EE	large empty space containing one of { s,v,w,x,y,z }.

A discrimination test decides which member of a subset of these features is present at a given location. A list is used to show the features discriminated by a test. For example, (1 2) is a test that discriminates between feature 1 (closed at both the top and bottom) and feature 2 (closed at the top). The locations used by hypothesis testing are the areas *between* the features discovered by the neighborhood calculation. Several of the hypothesis testing features can be adjacent to one another in these locations. For example, in the sequence "ba", the area between the short vertical bar in the "b" and the short vertical bar in the "a" contains the hypothesis testing feature E4.

An example is shown in Figure 2 of how the discrimination tests are arranged in a tree. The hypothesis generation procedure determined that there were four features in the input word and four locations (1 through 4) between those features at which a discrimination test could be applied. The features of the hypothesis generation stage are numbered 2110 in the second line of Figure 2. The 2 refers to an ascender, the 1's to short vertical bars, and the 0 the a significant vertical space that does not contain a vertical bar. These are all present in the same sequence in the neighborhood { *be*, *has*, *he* }.

The nodes at the first level of the tree are the tests that could be applied at each of the four locations. The result of a test either determines a recognition or the next set of tests that could be applied. In Figure 2, if location 3 is considered and the discrimination between features E and E4 is performed, *has* is recognized if E4 is present. Otherwise, if E is present, the choices are narrowed down to *be* or *he*. If feature 1 is then found in position 2, *be* is recognized, Otherwise if feature 2 is found in position 2, *he* is recognized.

Of particular interest are the *shortest paths* in a hypothesis testing tree. These are paths from a top-level node to a terminal node (all of its descendents are word decisions). A shortest path contains the fewest tests needed to recognize the words in a hypothesis testing tree. There are four shortest paths in Figure 2. They are from position 2 to position 3 (contains tests (1 2) and (E E4)), position 2 to position 4 (contains tests (1 2) and (6E EE)), position 3 to position 2 (contains tests (1 2) and (E E4)), and from position 4 to position 2 (contains tests (1 2) and (6E EE)). Therefore, there are two different mimimum sets of tests that can be used to recognize the words in this tree. They are { (1 2), (E E4) } and { (1 2), (6E EE) }. Henceforth, a shortest path will also mean the set of tests it contains.

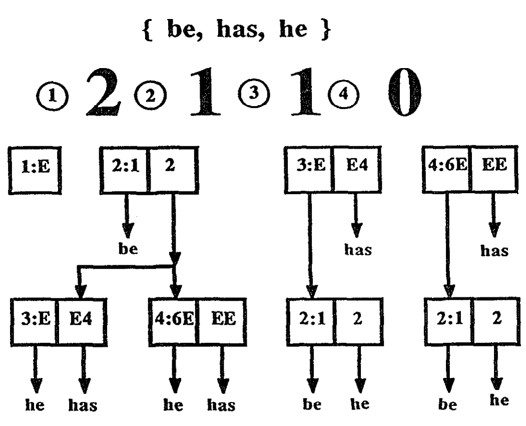

Figure 2. An example search tree for the neighborhood { be, has, he } with visual description "2110" Each node contains a position indicator "n:" where n is one of 1 through 4. The test at each node is carried out by discriminating between the features that follow the position indicator.

5. Minimum Number of Tests

It is of interest to determine the minimum number of different tests that would have to be executed to recognize every word in a given dictionary. (N.B. A dictionary is partitioned into neighborhoods by the hypothesis generation stage. Each neighborhood is in one-to-one correspondence with a hypothesis testing tree.) This set of tests (called MT) contains the smallest set of tests that would have to be programmed under the constraint that the minimum number are to be executed. As such, it indicates the computational effort needed to recognize the words in the dictionary and gives us a way to compare this methodology to other reading algorithms. The more efficient technique would require fewer tests and thus would have a smaller MT.

Several of the issues involved in finding MT are illustrated by the following example. The top 100 most frequent words in the Brown Corpus were chosen and the testing trees for these words were determined. The Brown Corpus is a text of over 1,000,000 words that was designed to represent modern edited American English [8]. Of the 100 most frequent words in the Corpus, 55 are uniquely recognized by the hypothesis generation computation. The other 45 words are contained in 16 trees. These words are shown in Figure 3 along with the tests on the shortest paths in their trees.

A shortest path is shown as a list of tests and a tree is represented by a list of the shortest paths it contains. In this example, there are as few as one (trees 5, 8, 10-13, and 16) and as many as eleven different shortest paths (tree 7) in a testing tree. Overall, there are 25 different tests on the 16 shortest paths in Figure 3. They are:

(1 2) (E E4) (6E EE) (EE E4) (1 E E4) (6E EE E) (1 E) (6E E) (EE EE4 E4) (6E 6E4) (10E E) (10E 6E) (10E 2) (6E 6E4 E) (10E 2 E) (10E 6E EE) (6EE 6E) (EE E) (10E 6EE EE) (1 2 5E4 6EE) (1 6E E E4) (1 3) (2 3) (1 3 E) (10E 6E)

The objective is to choose the smallest subset of these tests that contains the tests on at least one of the shortest paths in each tree.

tree	words	shortest paths
1	be, has, he	(((1 2) (E E4)) ((1 2) (6E EE)))
2	have, her, for	(((1 E E4)) ((6E EE E)))
3	what, who	(((E E4)) ((1 E)))
4	well, all	(((EE E4)) ((6E E)))
5	was, way, we, as	(((EE EE4 E4)))
6	so, at	(((EE E4)) ((1 E)))
7	years, were, are, any	(((6E 6E4) (EE E4)) ((10E E) (EE E4)) ((10EE 6E) (EE E4)) ((10E 2) (EE E4)) ((6E EE) (EE E4)) ((10E 2) (6E 6E4 E)) ((6E 6E4 E) (6E EE)) ((10E 2 E) (EE E4)) ((10E 2 E) (6E E)) ((10EE 6E EE) (EE E4)) ((10EE 6E EE) (6E Z)))
8	they, the	(((6EE 6E)))
9	down, than, then	(((1 E E4)) ((6E EE E)))
10	two, to	(((EE E)))
11	or, my, new	(((10E 6EE EE)))
12	on, no, can, even	(((1 2 5E4 6EE)))
13	me, may, now, over	(((1 6E E E4)))
14	out, not	(((1 2)) ((1 3)))
15	one, our	(((2 3)) ((10E 6E)))
16	man, most, must	(((1 3 E)))

Figure 3. The shortest paths in the testing trees for the 16 neighborhoods computed from the top 100 most frequent words in the Brown Corpus.

The brute force algorithm for this is to evaluate every possible subset and determine if the tests in it can solve every tree. However, this is obviously unsuitable because of an exponential complexity. Therefore, an approach is needed that reduces the computation. An algorithm is proposed here that achieves this objective.

The algorithm takes the shortest paths from a number of trees as input (Figure 3 is an example). It iteratively adds the tests on a shortest path to the solution set (initially empty) until it contains the tests that can be used to recognize the words in every tree. The solution set is then output. It should be noted that all the tests on one of the shortest paths in each tree must be in the solution. Otherwise, the words in that tree could not be recognized. For example, in tree one of Figure 3, either both (1 2) and (E E4) or both (1 2) and (6E EE) must be in the solution. A more precise description of the algorithm is:

Algorithm:

(1). Take the union of the shortest paths;

(2). Determine the number of trees that are solved by the tests in each shortest path. (Solving a tree is equivalent to having the tests on one of its shortest paths in the solution.)

(3). Find the shortest paths that contain the most tests and solve the maximum number of trees; in the above example there are eight shortest paths that contain two tests and solve three trees. For example, tests (1 2) and (E E4) solve trees 1, 3, and 14; (6E 6E4) and (EE E4) solve trees 4, 6 and 7, etc.

(4). Add the tests from one of the shortest paths discovered in step 3 to the solution. Remove the trees that are solved from consideration. Derive a reduced set of shortest paths by removing tests that are in the solution.

(5). If such a reduction yields an empty set of shortest paths, output the tests in the solution. Stop only if one of the possibly many solutions is desired. Otherwise continue with step one.

This algorithm must terminate because every reduction is guaranteed to produce a smaller set of shortest paths. There are 16 different solutions to the example set of shortest paths in Figure 3. Each solution contains 13 tests. One solution is:

(1 2) (E E4) (EE EE4 E4) (EE E4) (6E 6E4) (1 E E4) (2 3) (6EE 6E) (EE E) (10E 6EE EE) (1 2 5E4 6EE) (1 6E E E4) (1 3 E)

The 16 solutions to the example set of equations illustrates that there may be many solutions that contain the minimum number of tests. Choosing the "easiest" among them is largely a matter of judgement and intuition. The algorithm could be tuned to expand only the most promising paths, i.e., those that early-on are found to contain "easy" tests.

A version of the above procedure was implemented that found the first set of tests on a shortest path. This shows the smallest number of tests that must be executed to recognize a given vocabulary. This was applied to two vocabularies. The first were subsets of the Brown Corpus determined by frequency. The second were the subject categories or genres that make up the Brown Corpus. These results are shown in Table 1.

These results are interesting because they show at most 347 different tests are needed to recognize the text in any subject category of the Brown Corpus. The experiments that increased dictionary size show a linear effect of the number of dictionary words on the number of tests needed to recognize those words. Usually, doubling the dictionary size increases the number of tests by fifty percent. Another interesting result of this study is the difference between genres. Only 184 tests are needed to recognize the 34,495 words in genre D. This is very interesting in comparison to the 220 tests needed to recognize nearly the same number of words (35,466) in genre C, i.e., 36 more tests are needed to recognize only 1000 more words. This could be attributable to the difference in subject categories (genre C contains press reviews and genre D contains religious material). However, it is more likely due to the difference in the number of words in the dictionaries of the genres. The dictionaries for genres C and D contain 7751 words and 5733 words, respectively. Overall, the results indicate that the hypothesis testing methodology is economical and requires much fewer than the theoretical maximum of 2^{12} tests to be programmed. In fact, so few tests are needed that an implementation with high degrees of performance should be achievable.

6. Comparison to Character Recognition

The reading algorithm proposed in this paper has been compared to a character recognition algorithm of similar design [7]. This was done by designing a character recognition algorithm that was comparable to the proposed reading algorithm and determining the minimum number of tests needed by both approaches.

The character recognition algorithm was designed to contain both a hypothesis generation and a hypothesis testing phase. The hypothesis generation stage was the same as that of the proposed technique. Because a character recognition algorithm requires that individual characters be isolated, an additional assumption was made that the characters in the input words could be perfectly segmented and, thus, the number of characters in each word could be determined. Therefore, the contents of each neighborhood were constrained to be words with the same number of characters.

The hypothesis testing phase of the character recognition algorithm used the same design as the proposed technique. The only difference was in the discrimination tests. They were between-characters rather than between-features. This is the point where the two methods differed. For example, in the neighborhood { may, now } three character-discriminations were possible: (m n), (a o), and (y w). In contrast, the proposed technique would discriminate among the features that occur between the vertical bars.

The character recognition algorithm can be summarized:

1. Assume that an input word can be perfectly segmented and that it comes from a given, fixed vocabulary. Identify the locations of each character.
2. Determine the neighborhood of the input word in the given vocabulary. The same features and feature extraction procedure are used as in the proposed methodology except that the neighborhood must contain words with the same number of characters.
3. Use the proposed method of hypothesis testing to recognize the input. Because this is a character recognition approach, the discriminations are between characters that occur in the positions specified by step 1.

One modification to the proposed method was needed to make the neighborhoods calculated by the two methods the same. The hypothesis generation algorithm used an additional parameter of the number of characters in each word. Note that the modification requires only that the number of characters in a word be determined. This can be much easier than segmenting a word into isolated characters.

dict. size	words of text	no. of tests	dict size	words of text	no. of tests
50	413,565	6	5000	880,802	172
100	483,355	13	6000	898,145	197
250	566,288	27	7000	912,068	205
500	632,693	42	8000	923,358	223
750	674,112	51	9000	932,831	246
1000	705,045	61	10,000	940,871	252
2000	781,581	92	15,000	968,004	331
3000	827,178	128	20,000	984,105	368
4000	858,306	155	25,000	994,105	425

genre	words of text	no. of tests	genre	words of text	no. of tests
A	88,051	297	J	160,877	303
B	54,662	225	K	58,650	226
C	35,466	203	L	48,462	219
D	34,495	184	M	12,127	127
E	72,529	256	N	58,790	235
F	97,658	294	P	59,014	238
G	152,662	347	R	18,447	158
H	61,659	206			

Table 1. The minimum number of tests that must be executed to recognize the indicated vocabularies.

The minimum number of tests needed by the algorithm proposed in this paper and the character recognition technique were determined for the genres of the Brown Corpus. It was discovered that the proposed algorithm would have to execute from one percent to fourteen percent fewer tests to recognize the same texts. However, from three percent to seven percent of the running text could not be completely recognized by the proposed technique. These were words where the hypothesis testing algorithm could not reduce the neighborhood to a unique choice. Usually, there were only two or three choices for such a word and these choices were from different syntactic or semantic categories. It is hoped that future work with such higher level knowledge will allow these ambiguities to be reduced. The conclusion of this experiment was that the proposed algorithm would execute significantly fewer tests than a similar character recognition technique at a tolerable cost in text that could not be completely recognized.

7. Experimental Results

The viability of the hypothesis testing strategy and its ability to adapt to different input conditions was demonstrated by its implementation for a dictionary of 630 words from a randomly-selected sample of 2003 running words from the Corpus. Each of these words was generated in ten different fonts. The fonts were 24 pt. samples digitized at 500 pixels per inch binary on a laser drum scanner. Word images were generated by appending the images of the appropriate characters. Word images were also generated by appending the characters and moving them horizontally until they touched. This is very difficult for some techniques to compensate for. This resulted in 12,600 input images.

Five of the ten fonts were used as training data. The other fonts were subject to no image processing before they were used for recognition testing. The 6300 words in the training data were recognized correctly 98% of the time when the characters were not touching and 95% of the time when the characters were touching. The other cases were errors. The 6300 words not in the training data were correctly recognized in 95% of the cases when the characters were not touching and 92% of the time when the characters were touching.

8. Discussion and Conclusions

A computational theory and algorithm for fluent reading was presented. The work presented in this paper sought to bridge the gap between theory and methods and to bring to reading algorithms the benefits of many years of psychological investigation of human reading. The mechanism used to effect this transfer was a computational theory and its related algorithms. This is an example of applying the theoretical constructs of Artificial Intelligence to an engineering problem.

It was seen that people do not read by recognizing isolated characters as do most current techniques. Instead, people recognize larger groups of letters or words. This recognition process uses at least two stages. One uses a gross visual description to develop expectations about words in a running text. The other integrates these expectations with detailed visual processing to form complete perceptions of the words in the text. This stage of processing is very individualized and subject to change based on many external factors. Another process that occurs during reading uses high-level knowledge to affect the visual processing.

This paper discussed algorithms that performed the two stages of visual processing. A method for hypothesis generation was presented that extracted a gross visual description of words and used it to return a number of hypotheses from a dictionary that contained the word in the input image. A technique for hypothesis testing was also presented. This method was structured as a tree search of discrimination tests. The result of a discrimination test was either a recognition of the input word or another set of tests that could be applied to the image. The tree search methodology was set up so that different testing strategies could be used to recognize a word, as a human reader is capable of doing.

An algorithm was presented that determined the minimum number of different discrimination-tests needed to recognize the words in a large vocabulary. It was shown that this technique requires a small number of tests to recognize any word in large subsets of text. Recognition experiments on many different fonts showed that about 95% correct recognition was achieved on 12,600 word images. This demonstrates the ability of this methodology to tolerate different formats and its potential to reach high levels of performance.

ACKNOWLEDGEMENTS

The author is grateful to Sargur N. Srihari for valuable consultations. We acknowledge the support of the Office of Advanced Technology of the United States Postal Service under contract BOA 104230-84-0962.

REFERENCES

1. W. K. Estes, "On the interaction of perception and memory in reading," in *Basic Processes in Reading: Perception and Comprehension*, D. LaBerge and S. J. Samuels (editor), Lawrence Erlbaum Associates, Hillside, New Jersey, 1977.
2. L. Frazier and K. Rayner, "Making and correcting errors during sentence comprehension: eye movements in the analysis of structurally ambiguous sentences," *Cognitive Psychology 14* (1982), 178-210.
3. R. N. Haber and L. R. Haber, "Visual components of the reading process," *Visible Language XV*, 2 (1981), 147-181.
4. J. Hochberg, "Components of literacy: Speculations and exploratory research," in *Basic Studies on Reading*, H. Levin and J. P. Williams (editor), Basic Books, Inc., New York, 1970, 74-89.
5. J. J. Hull, "Inter-word constraints in visual word recognition," *Proceedings of the Conference of the Canadian Society for Computational Studies of Intelligence*, Montreal, Canada, May 21-23, 1986, 134-138.
6. J. J. Hull, "Hypothesis generation in a computational model for visual word recognition," *IEEE Expert*, August, 1986, 63-70.
7. J. J. Hull, "A computational theory of visual word recognition," Technical Report, SUNY at Buffalo, Department of Computer Science, 1987.
8. H. Kucera and W. N. Francis, *Computational analysis of present-day American English*, Brown University Press, Providence, Rhode Island, 1967.
9. D. Marr, *Vision*, W.H. Freeman and Company, San Francisco, 1982.
10. K. Rayner, *Eye movements in reading: perceptual and language processes*, Academic Press, New York, 1983.
11. J. Schurmann, "Reading machines," *Proceedings of the 6th International Conference on Pattern Recognition*, Munich, West Germany, October 19-22, 1982, 1031-1044.
12. L. G. Shapiro, "The role of AI in computer vision," *The Second IEEE Conference on Artificial Intelligence Applications*, Miami Beach, Florida, December 11-13, 1985, 76-81.