Programming Pattern Recognition*

G. P. DINNEEN†

EVERYONE LIKES to speculate, and recently there has been a lot of talk about reading machines and hearing machines. We know it is possible to simulate speech. This raises lots of interesting questions such as: If the machines can speak, will they squawk when you ask them to divide by zero? And can two machines carry on an intelligent conversation, say in Gaelic? And, of course, there is the expression "electronic brain" and the question, Do machines think? These questions are more philosophical than technical and I am going to duck them.

Over the past months in a series of after-hour and luncheon meetings, a group of us at the laboratory have speculated on problems in this area. Our feeling, pretty much unanimously, was that there is a real *need* to get practical, to pick a real live problem and go after it. As the title of this session suggests, we picked pattern recognition. Selfridge has already explored some of the implications of this problem and indicated at least three levels of complexity.

Selfridge and I have begun an investigation of one aspect of pattern recognition, that is, the recognition of simple visual patterns. Consider, for example, how many different representations of the block capital A we recognize as A. A great number of variations in such things as orientation, thickness, and size can occur without loss of identity. Our real live problem then is to design a machine which will recognize patterns. We shall call any possible visual image, no matter how cluttered or indistinct, a configuration. A pattern is an equivalence class consisting of all those configurations which cause the same output in the machine. For example, if the machine had just two outputs, yes and no, then one pattern would be all those inputs which caused the machine to say no.

The visual patterns we have used in our experiments have been mostly block capital A's and O's, and some squares and triangles. Why did we choose letters? Probably because of the interest in reading machines. Why A and O? Perhaps because O upside down is still O, like radar spelled backwards. Capital letters and geometric figures seem like a good start. Straight lines are too simple and most other things, such as faces, are too complicated. Let me emphasize again that we are not trying to build an efficient reading machine, but rather to study how such a machine might work.

* The research in this document was supported jointly by the Army, Navy, and Air Force under contract with the Massachusetts Institute of Technology.

† Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Mass.

Our theory of pattern recognition is that it is possible to reduce by means of a sequence of simple operations a configuration to a single number, or by means of a set of such sequences to a set of numbers. We believe that for the proper sequences, almost all of the configurations belonging to a given pattern reduce to the same number or set of numbers. At least one of the operations of this machine must be a counting operation.

A machine for pattern recognition should place a configuration, such as one representation of the block capital A, into its proper equivalence class. The decision as to what things belong to an equivalence class or pattern is made at this stage by the designer. The machine must extract the essential characteristics of the configuration in order to recognize the pattern to which it belongs.

One of the first steps is the design of basic operations which the machine uses to reduce the input image. Although this is a preliminary and experimental phase of the study, the results obtained have been very interesting and extremely valuable in formulating future studies. For this reason, it seems worthwhile to discuss these results now.

The problem we have stated is not basically arithmetic, and it is not even clear that a machine for pattern recognition should be completely digital. However, one of the very important applications of a high-speed digital computer is the *simulation* of just such problems. We have used MTC—the Memory Test Computer—at Lincoln Laboratory for our investigation.

The Image

From now on, we are considering the simulation of this problem on MTC. The first problem is "how does MTC see the image?" The field of view is a 90×90 array (Fig. 1). A picture is constructed by making each of the 8,100 cells black or white, similar to a newspaper photograph, except that there are only two gradations in intensity, zero to one. If black is interpreted as one and white as zero, and storage registers are assigned to the 90×90 array, six to a row, as in Fig. 2, a simple image may be described by giving the contents of the 540 register as octal numbers. A flexowriter tape giving the contents of 540 registers is read into the machine and the visual image is thus stored by the computer. We chose a 90×90 array because we wanted the image to be as large as was practical in terms of computer storage. The larger the image, of course, the greater the resolution and the variety of inputs.

For purposes of pattern recognition, certain basic

operations will be performed on the input image. Each operation performs a mapping of the input image, or working image. The result of this mapping is also stored in the machine. Thus another bank of 540 registers is required for storage of this new image. If a sequence of operations is to be performed, then the transformed image is transferred back to the input storage or working image after each operation. If it is desired to retain the original image, then three banks are required, as shown in Fig. 3.

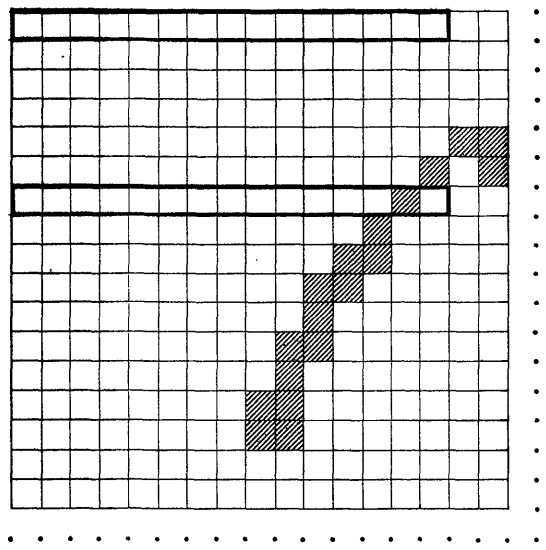

Fig. 1—90×90 input array.

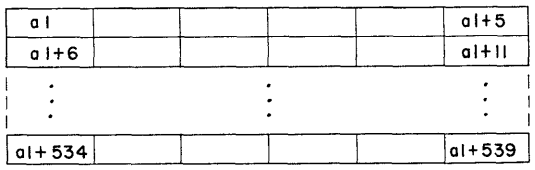

Fig. 2—The input image.

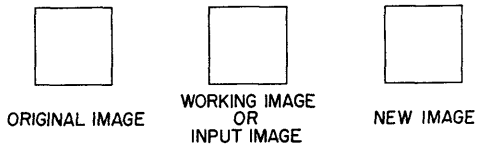

Fig. 3

Averaging Operation

The first of the basic operations is an averaging operation which was designed to eliminate spurious noise and to smooth irregularities in the image. The point of this operation is that those elements or cells which agree with their neighbors are unchanged, but those which do not are changed. The notion is that the resulting image is more homogeneous. The averaging operation is performed by observing the contents of all the elements in some $n \times n$ square surrounding an element. In Fig. 4 we are considering element a_{ij}, thus we look at the elements in the 5×5 square surrounding it. Let us agree to call this square a "window." To complete the operation we count the number of ones in this window and compare the total with some threshold. If the count is greater than or equal to the threshold then the corresponding element in the new image, a_{ij}' is "one," otherwise $a_{ij}' = 0$. For example, a zero surrounded by all zeros remains zero, but a zero surrounded by some ones may become zero or one, depending on the threshold.

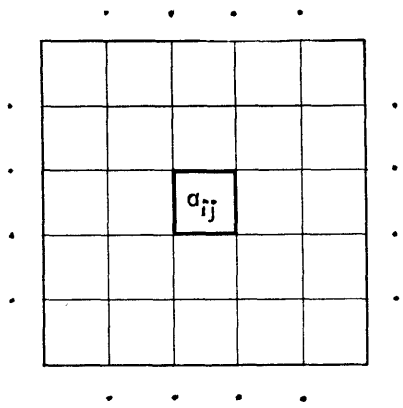

Fig. 4—Averaging operation.

The first averaging operation uses a 5×5 square. A count of the number of ones inside this square is made (Fig. 4).

$$N = \sum_{m=-2}^{+2} \sum_{n=-2}^{+2} a_{i+m,i+n}.$$

The N is compared with the threshold T and

$$N \geqq T \qquad a_{ij}' = 1$$
$$N < T \qquad a_{ij}' = 0.$$

The threshold T is held constant over the entire working image.

Results of the Averaging Operation

A large number of experimental runs were made using the computer to test the operation. As predicted, averaging with a 5×5 window and a threshold of 5 eliminated scattered "ones" and filled in holes. Fig. 5 is an input image, and Fig. 6 is the same image after this averaging operation. These are photographs of one of the MTC scopes.

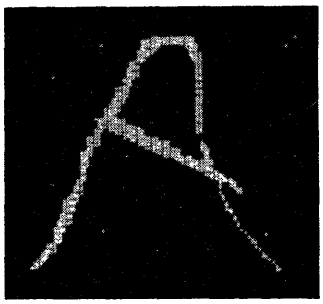

Fig. 5—A3, input image.

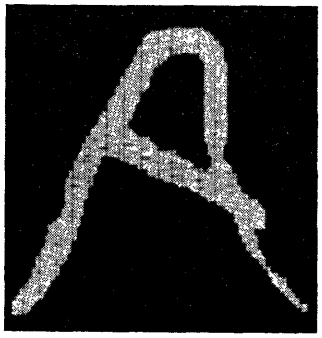

Fig. 6—A3, after averaging with threshold 5.

For a low threshold, such as 5 for a 5×5 window, the image will be thickened. As the threshold is raised a thinning takes place. This is evident in Figs. 7 through 11. It is particularly significant that for a threshold of 15, the corner point and two junction points are isolated. The same phenomenon is shown in Figs. 12 through 17. The thick A of Fig. 18 has a blank strip and one small hole. For the low thresholds these irregularities are removed and for the high thresholds they are emphasized, as shown in Figs. 18 through 25.

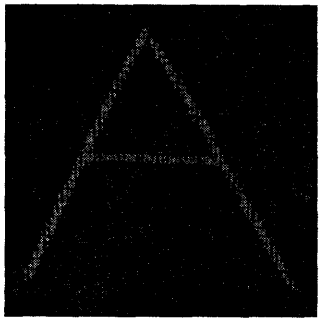

Fig. 7—A4, input image.

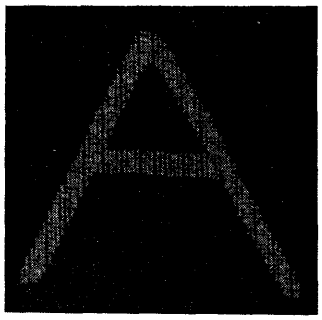

Fig. 8—A4, after averaging with threshold 5.

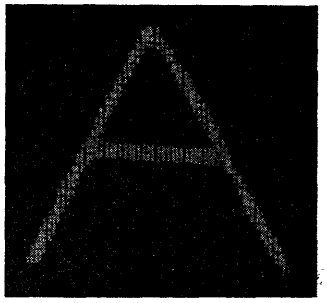

Fig. 9—A4, after averaging with threshold 10.

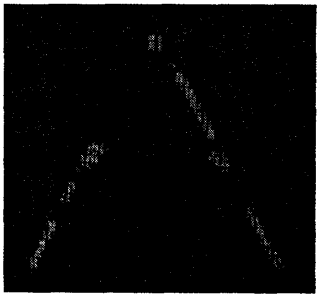

Fig. 10—A4, after averaging with threshold 13.

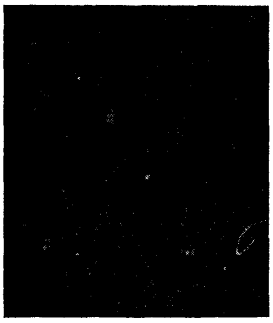

Fig. 11—A4, after averaging with threshold 15.

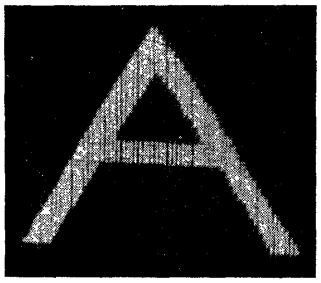

Fig. 12—A5, input image.

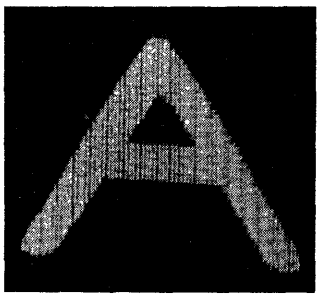

Fig. 13—A5, after averaging with threshold 5.

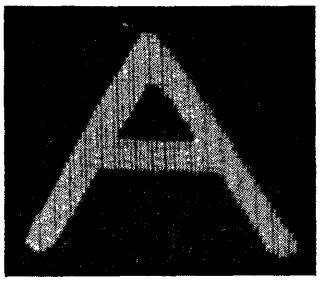

Fig. 14—A5, after averaging with threshold 10.

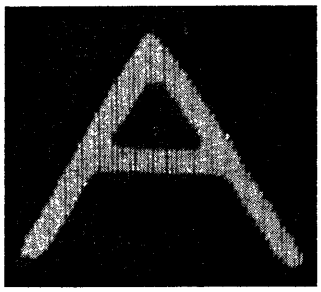

Fig. 15—A5, after averaging with threshold 15

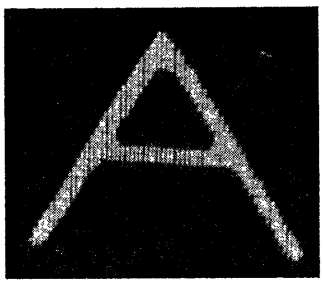

Fig. 16—A5, after averaging with threshold 20.

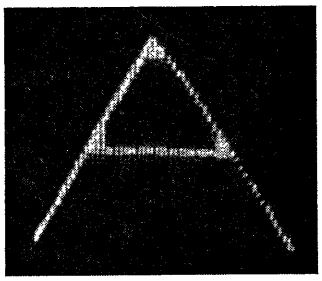

Fig. 17—A5, after averaging with threshold 25.

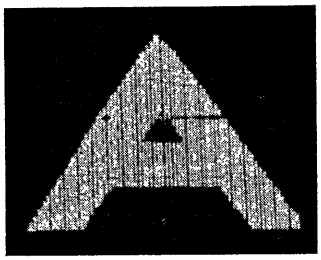

Fig. 18—A6, input image.

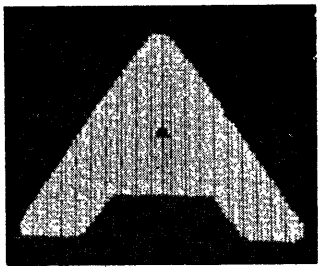

Fig. 19—A6, after averaging with threshold 3.

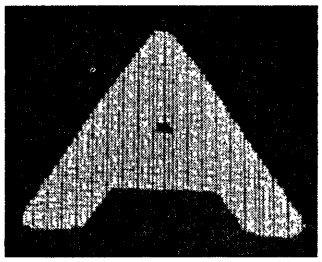

Fig. 20—A6, after averaging with threshold 5.

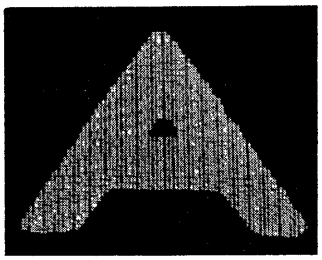

Fig. 21—A6, after averaging with threshold 10.

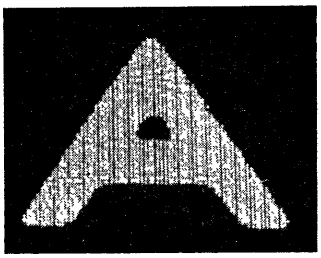

Fig. 22—A6, after averaging with threshold 15.

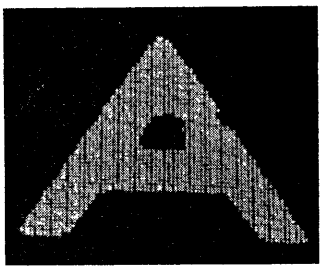

Fig. 23—A6, after averaging with threshold 20.

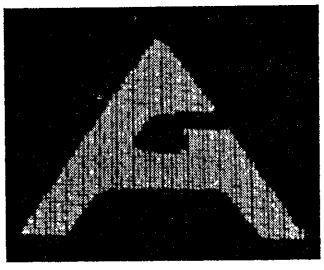

Fig. 24—A6, after averaging with threshold 22.

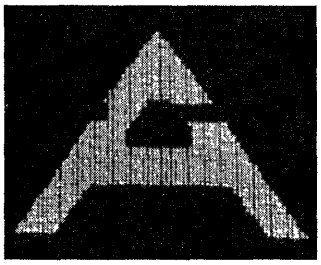

Fig. 25—A6, after averaging with threshold 25.

We see that for rough smoothing we like a low threshold, and for thinning we like a high threshold. Our notion of sequence of operations is now brought to mind. For if a letter such as the last one were first averaged with a low threshold the gaps would be filled. Then successive averaging with a higher threshold would thin the letter. The problems one encounters in this next phase are apparent. How often should these operations be repeated? What thresholds should be used?

You will have noticed that although the transformation is not sensitive to small changes in the threshold, large changes have a major effect. To attack the problem of setting the threshold, we have considered another averaging operation where the threshold for the small 5×5 window is determined by the degree of homogenity inside a larger window, say 15×15. In particular, for each cell we count the number of ones inside a 15×15 window surrounding it and use this number to set the threshold for the 5×5 window. This is like an automatic gain control. When the 15×15 window is very dense, we use a high threshold, and conversely. This operation has been tried and the results look favorable, but unfortunately we have no photographs yet.

An Edging Operation

The edging operation, quite unlike the averaging operation, preserves elements which are centers of asymmetry, that is, those which are located in the regions of discontinuity of the image. The operation sharpens differences. It picks out the edges of letters and for certain choices of thresholds locates the corners, junctions, and end points of letters. This operation is performed by observing the contents of all the elements in some $n\times n$ square surrounding an element which is one; zeros remain zero. In Fig. 26 we are considering element a_{ij}.

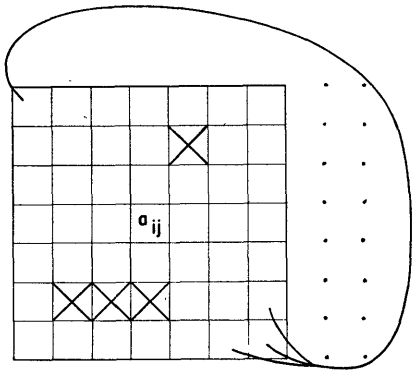

Fig. 26—The edging operation.

The edging operation is like a two-dimensional derivative, since we count changes about the center element. In particular, we scan the elements in the 7×7 window starting at $a_{i-3,j-3}$. When we reach a "one" we observe the three diagonally opposite elements in the same ring. If these are all zero we count one, otherwise we count zero. We consider the three diagonally opposite bins so that lines with small curvature give a zero count of differences. After the entire window has been scanned, with the exception of element a_{ij}, the total count of differences is compared with a threshold T. If the count is greater than the threshold, we enter a one in the element a_{ij}' of the new image. If the count is less than or equal to the threshold, we enter a "zero" in the element a_{ij}' of the new image. To fix the threshold, we first count the total number of ones in the window and take some proportion of this as the threshold. A further modification of this operation is the weighting of the count of differences depending on the ring, that is, we put weight W_1 on the elements in the first ring around the element a_{ij}, W_2 for the second ring, and so on. To identify this operation, we must specify the size of the window, the weighting, and the threshold.

We first tried an edging operation which used a 7×7 window with a weight of two on all differences which occur in the first and second ring, and a weight of one on all differences in the outer ring. The total number of ones in the 7×7 window, exclusive of the center element, will be denoted by N. The threshold T of the operation was taken to be $(j/8)\cdot N$ for $j=0, 1, 2, \cdots, 8$.

Results of the Edging Operation

In testing the edging operation we were interested in its ability to pick out singular points, that is, junction points, end points, corners in the case of letters, and in its ability to transform the input image into a new image where only the outline remains.

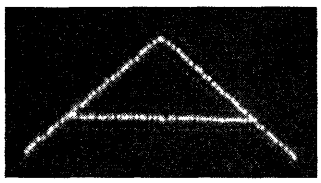

Fig. 27—Thin A.

For an input image, Fig. 27, edging with thresholds 2/8, 4/8, and 6/8 of the ring total isolates the singular points as shown in Figs. 28, 29, and 30. Edging with a threshold of 0/8 of the ring total is equivalent to removing only those points which are centers of symmetry. This effect is shown in Fig. 31 for the input image of Fig. 12. The outline of the letter is shown in Fig. 32, the result of edging with a threshold of 2/8 times the ring

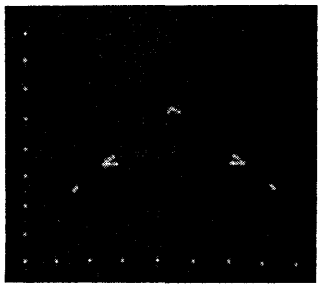

Fig. 28—Edging with threshold 2/8.

Fig. 29—Edging with threshold 4/8.

Fig. 30—Edging with threshold 6/8.

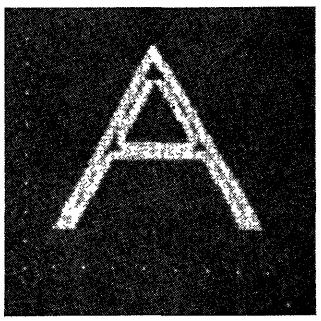

Fig. 31—Edging with threshold 0/8.

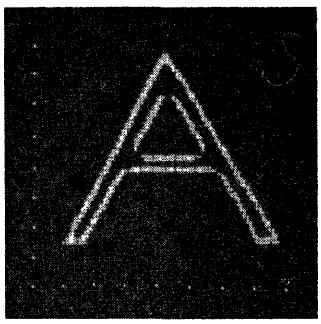

Fig. 32—Edging with threshold 2/8.

total. For a thick A (see Fig. 18) the results are shown in Figs. 33 through 35. It is significant that the irregularities are emphasized for low thresholds and suppressed for the high thresholds. Figs. 36 through 39 demonstrate the way corners are isolated in a square and the edging effect for a 0 is shown in Figs. 39 and 40.

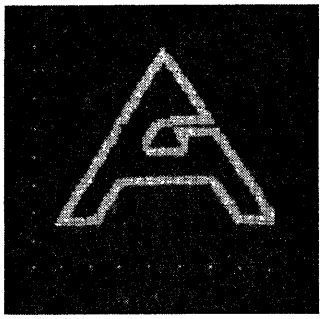

Fig. 33—Edging with threshold 0/8.

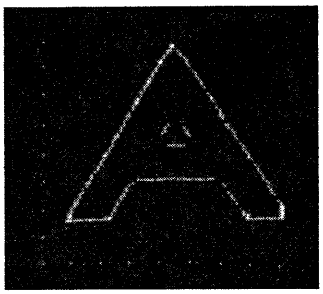

Fig. 34—Edging with threshold 4/8.

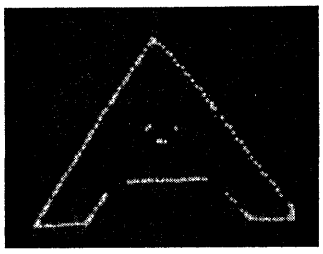

Fig. 35—Edging with threshold 6/8.

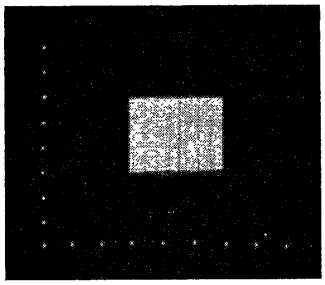

Fig. 36—Square.

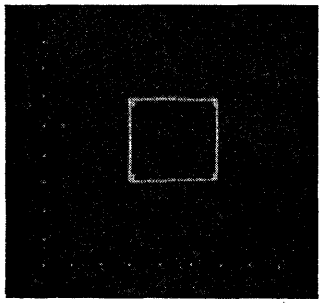

Fig. 37—Edging with threshold 4/8.

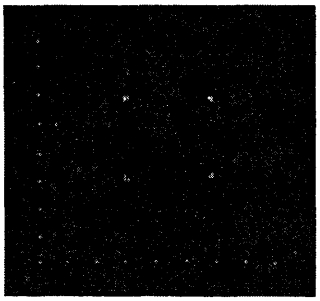

Fig. 38—Edging with threshold 8/8.

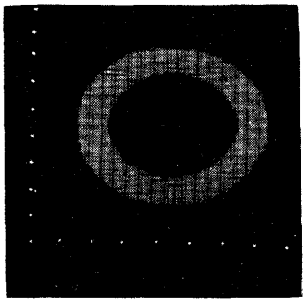

Fig. 39—Thick 0.

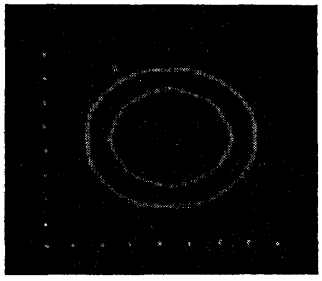

Fig. 40—Edging with threshold 2/8.

Other Operations

Before I say anything about new operations, a word about our reasons for these first two operations is in order. We tried to pick operations which were simple in structure and unrelated to particular patterns. In other words, we avoided special operations which might work very well for A's and O's, but not very well for other shapes. There is evidence in neurophysiology, moreover, that both averaging and edging of visual images are performed by the nervous system of many animals.

Many other operations have been proposed but I would like to discuss just one of these. Selfridge talked about identifying a letter by counting blobs. By blob we mean a region of ones with no separation of more than one cell. For example, there were five blobs in Fig. 29. A blob counter is then simply an operation which counts these blobs. We propose to do this by scanning the image until a one is reached. Then a rectangle with a perimeter two cells wide will be constructed and gradually enlarged until the entire perimeter is empty. Thus the blob is isolated and counted.

Certainly some operation to measure curvature will be needed, as well as some way to describe relative positions of blobs.

Although the first two operations are very simple, the coding of the program is quite complicated. This is due to a large extent to the necessity of using the individual bits of the storage registers as independent storage bins. We need a lot of shifting and cycling; the simple averaging operation takes about 300 instructions and requires about 20 seconds a letter. The two window averaging takes about 400 instructions and requires about 4 minutes. The edging requires 700 registers of instructions and takes 2 minutes.

Conclusion

In his paper, Selfridge has given our model for pattern recognition. I think we have shown that even with just two simple operations, a sequence can be found such that some A's can be reduced to five blobs. One of our next tasks will be to choose such a sequence, try it on many A's, and plot the distribution of the number of blobs. In this way we shall determine the useful sequences. This leads naturally to the next phase of the study, which is the study of the model for pattern recognition.

Acknowledgment

For their advice and encouragement, I wish to thank J. V. Harrington, I. S. Reed, O. G. Selfridge, H. Sherman, and G. E. Valley, who participated in the seminar which launched this study. It would have been impossible to obtain the experimental results without the excellent assistance of Miss B. Jensen and Mrs. M. Kannel in programming, coding, and operating the Memory Test Computer. I wish to acknowledge the excellent co-operation of those responsible for the operation of the Memory Test Computer.