On the Optimal Detection of Curves in Noisy Pictures

Ugo Montanari
Istituto di Elaborazione della Informazione, Pisa, Italy

A technique for recognizing systems of lines is presented. In this technique the heuristic of the problem is not embedded in the recognition algorithm but is expressed in a figure of merit. A multistage decision process is then able to recognize in the input picture the optimal system of lines according to the given figure of merit. Due to the global approach, greater flexibility and adequacy in the particular problem is achieved. The relation between the structure of the figure of merit and the complexity of the optimization process is then discussed. The method described is suitable for parallel processing because the operations relative to each state can be computed in parallel, and the number of stages is equal to the length N of the curves (or to $\log_2 (N)$ if the approximate method is used).

Key Words and Phrases: picture processing, picture recognition, picture description, curve detection, line detection, edge detection, optimal detection, heuristic methods, global recognition, parallel processing, dynamic programming, interaction graph, secondary optimization problem
CR Categories: 3.63, 3.66, 5.42

1. Introduction

The recognition of curves is often an important and critical stage in image processing. In some cases—e.g. in bubble chamber photograph analysis or in character recognition—the image is already line-like. In other applications it is reduced to a line-like image by means of derivative techniques (gradient, Laplacian). Then the problem arises of finding a system of curves in the picture which is meaningful to the next step in the processing. For a deep analysis and comparison of the known techniques for edge and curves detection see [1]. A particular field in which curve detection plays a pivotal role is, for instance, television bandwidth reduction through transmission of coded contours [2,3].

For most applications, we can distinguish two phases in the processing of line-like figures as follow.
(1) A set of procedures which operates on the input picture with the purpose of eliminating most of the local, uncorrelated noise. Characteristics of this stage are that operation takes place on a local basis and that a repertory of methods exists for filtering and enhancing which is practically problem-independent. These operations are usually simple because the number of points to be processed is often very high.
(2) A more sophisticated "ad hoc" recognition procedure. This phase is usually global, and context is used for solving ambiguous situations.

This scheme applies, for instance, to fingerprint analysis: it is too complicated to follow dermatoglyphics directly on the original figure, and thus a first phase of "information concentration" is required [4].

The presence of a "preprocessing" and a "processing" is not too satisfactory from an information point of view. In fact a filter, if optimal, must always be "matched" to the class of images to be found. On the other hand, a "tuned" procedure is often very expensive in programming time, because the heuristic of the problem must be "built in" in the program and usually cannot be given as a datum. Another disadvantage is inflexibility: minor changes in the processing technique or in the characteristics of the source can require reprogramming of large sections.

In this paper a recognition technique is presented which is based on the determination of the optimal system of curves in a picture with respect to a given figure of merit (FOM). Therefore the heuristic content of the problem is expressed specifying the properties which a curve must have and the relative weights of them. Constraint of various kinds (geometrical, topological) can also be embedded in the FOM.

The number of possible curves is usually very high (in the experimental results given in this paper it is approximately 10^{24}) and grows exponentially with the length of the curves. However, for a large class of FOM's, a multistage optimization procedure [5] can be carried on, so that the optimum can be found during practical computing times.

The idea of using dynamic programming in pattern recognition and description was presented in earlier papers by Kovalewsky [6]. He applied this method to the recognition of a line of typed characters and to the description of individual handprinted characters.

In what follows, particular relevance is given to the preprocessing problems, because sequential optimization methods are sufficiently fast (especially if approximate solution procedures are used) for handling large masses of data. On the other hand, "tuned" FOM's can be used which allow recognition under otherwise prohibitive conditions.

In the general case of detection of *systems* of lines, the serialization of the optimization problem is nontrivial. Therefore, the "secondary optimization problem" [7,8] is introduced. The structural relations between elements of the system of lines have a counterpart in the connections of an "interaction graph." Simple rules for finding an optimal solution are given for trees, series-parallel graphs, and other particular classes of graphs.

Some experimental results are finally presented: a generic curve with superimposed Gaussian noise is successfully recognized even if the mean square values of the signal and of the noise are the same. Also, when there is substantially less noise, no recognition of the line is possible by a visual inspection of the image.

2. Dynamic Programming

In this section we briefly review the concepts used in dealing with multistage optimization processes.

Let us state a rather general optimization problem: (a) find the maximum value M of the merit function $g = g(x_1, x_2, \cdots, x_N)$, $0 \leq x_i \leq n_i$, $i = 1, \cdots, N$; (b) find a value $(\bar{x}_1, \cdots, \bar{x}_N)$ for which the maximum is actually achieved.

If the variables $x_1, \ldots, x_N$ can assume only discrete values, and no particular regularity of the function (like unimodality) is known, the problem can be solved only by complete exhaustion. However, if the merit function g is a sum of terms, each of which depends on only a few variables, then a multistage optimization procedure applies. For instance, if

$$g(x_1, \cdots, x_N) = g_1(x_1, x_2) + g_2(x_2, x_3) + \cdots + g_{N-1}(x_{N-1}, x_N), \tag{1}$$

then the following recursion formula can be used [5]:

$$f_1(x_1) = 0,$$
$$f_{k+1}(x_{k+1}) = \max_{0 \leq x_k \leq n_k} (g_k(x_k, x_{k+1}) + f_k(x_k)),$$

where $k = 1, \cdots, N-1$ and $0 \leq x_{k+1} \leq n_{k+1}$. The intermediate values $f_{k+1}(x_{k+1})$, $0 \leq x_{k+1} \leq n_{k+1}$, and the values $m_{k+1}(x_{k+1})$ of x_k for which the maximum is achieved must be saved in a table with n_{k+1} entries. Thus, at the end of this phase, N tables have been stored. The formula

$$M = \max_{x_N} f_N(x_N)$$

solves part (a) of the original problem. Part (b) is solved by scanning the stored tables with the recursion formula

$$\bar{x}_N = m_N = \arg\max_{x_N} f_N(x_N),$$
$$\bar{x}_k = m_{k+1}(\bar{x}_{k+1}), \qquad k = N-1, \cdots, 1.$$

In Table I we show a simple numerical example. It is easy to see that in this case the number of operations required using dynamic programming is much less than it is for the exhaustion method. If for instance, $n_i = n$, $i = 1, \cdots, N$, then roughly n^2N additions and tests are required for the dynamic programming method and $n^N N$ additions and n^N tests for the brute-force method. This improvement is obtained at the expense of an amount of storage roughly equal to $n^2 + nN$.[1] Note that

Table I

$N = 3; \quad n_1 = 2; \quad n_2 = 3; \quad n_3 = 2$

$g(x_1, x_2, x_3) = g_1(x_1, x_2) + g_2(x_2, x_3)$

x_1/x_2	$g_1(x_1, x_2)$		
	0	1	2
0	7	3	8
1	4	5	4

x_2/x_3	$g_2(x_2, x_3)$	
	0	1
0	4	6
1	8	4
2	3	7

$f_1(x_1) = 0$

x_2	$f_2(x_2)$	$m_2(x_2)$
0	7	0
1	5	1
2	8	0

x_3/x_2	$f_2(x_2)$			$g_2(x_2, x_3)$			$f_2 + g_2$			$f_3(x_3)$	$m_3(x_3)$
	0	1	2	0	1	2	0	1	2		
0	7	5	8	4	8	3	11	13	11	13	1
1	7	5	8	6	4	7	13	9	15	15	2

$M = 15; \quad m_N = 1$

$\bar{x}_3 = m_N = 1$

$\bar{x}_2 = m_3(\bar{x}_3) = m_3(1) = 2$

$\bar{x}_1 = m_2(\bar{x}_2) = m_2(2) = 0$

[1] The n^2 part is used in interstage optimizations and thus must be high speed memory, while the nN part is used for storing functions m_i and therefore can be allocated in peripheral devices like disks or drums.

with this scheme constrained problems can also be treated, because the constraints can be formally inserted in the merit function. Let us consider for instance the following well-known problem:

maximize

$$\bar{g}(x_1, \cdots, x_N) = g_1(x_1) + g_2(x_2) + \cdots + g_N(x_N)$$

with the constraint

$$\sum_{i=1}^{N} x_i = c; \qquad x_i \geq 0, \quad i = 1, \cdots, N.$$

With the new set of variables,

$$y_i = \sum_{k=1}^{i} x_k, \qquad i = 1, \cdots, N,$$

the above problem becomes

maximize

$$\bar{g}(y_1, \cdots, y_N) = g_1(y_1) + g_2(y_2 - y_1) + \cdots + g_N(y_N - y_{N-1})$$

with the constraints

$$y_i - y_{i-1} \geq 0; \qquad y_i \leq c, \quad i = 1, \cdots, N, \quad y_0 = 0.$$

Introducing the "infinite step function"

$$s(x) = \text{a very great number} \quad \text{if } x < 0,$$
$$s(x) = 0 \quad \text{if } x \geq 0,$$

we have finally:

maximize

$$\bar{g}(y_1, \cdots, y_N) = \sum_{k=1}^{N} (g_k(y_k - y_{k-1}) - s(y_k - y_{k-1}) - s(c - y_k)) = \sum_{k=1}^{N} \bar{g}_k(y_k, y_{k-1}).$$

This problem is now of the form (1) and thus can be treated with the dynamic programming approach.

These concepts are extensible to merit functions more general than (1). In general, a multistage optimization method can be described as a procedure in which the optimization is carried out separately for each variable x_i. Of course, this optimization must be performed for all the values of the independent variables which "interact" with the variable x_i. An *interaction graph* [7,8] can help in explaining this concept. In this graph, vertices V_i correspond to variables x_i and two vertices V_i and V_j are connected with an (undirected) arc iff the variables x_i and x_j *interact*: namely, if there exists at least one term of the merit function which depends on both x_i and x_j. For example, if the merit function is of the form:

$$\begin{aligned} g(x_1, x_2, x_3, x_4, x_5) &= g_1(x_1, x_2) + g_2(x_2, x_3) + g_3(x_3, x_4) \\ &\quad + g_4(x_1, x_4, x_5) \end{aligned} \tag{2}$$

then the interaction graph of Figure 1(a) is obtained. Now we carry on optimization with respect to variable x_i. Let $x_{i_1}, \cdots, x_{i_s}$ be the variables whica interact with x_i. The merit function can clearly be written as

$$g(x_1, \cdots, x_N) = h_1(x_1, x_{i_1}, \cdots, x_{i_s}) + h_2$$

where h_2 does not depend on x_i. Thus h_1 can be substituted with

$$h_{1_{\text{opt}}}(x_{i1}, \cdots, x_{i_s}) = \max_{x_i} h_1(x_i, x_{i_1}, \cdots, x_{i_s}).$$

The *cost* of the elimination of the variable x_i (in computing time and storage) is evidently substantially dependent on the number of points m which constitute the domain of the function $h_{1_{\text{opt}}}$. If $n_1 = n_2 = \cdots = n_N = n$, then $m = n^s$. The exponent s is called the *dimension* of the stage x_i. After elimination of the variable x_i, an optimization problem of the same type as the previous one is obtained. It is easy to see that the interaction graph of the new problem is derived from the interaction graph of the old problem by erasing vertex V_i and connecting all pairs of vertices of the set $(V_{i_1}, \cdots, V_{i_s})$. For instance, if we eliminate x_2 from the merit function (2) we get

$$\begin{aligned} g_5(x_1, x_3) &= \max_{x_2} (g_1(x_1, x_2) + g_2(x_2, x_3)), \\ \bar{g}(x_1, x_3, x_4, x_5) &= g_5(x_1, x_3) + g_3(x_3, x_4) \\ &\quad + g_4(x_1, x_4, x_5). \end{aligned}$$

The graph of the new problem is shown in Figure 1(b).[2]

A multistage optimization process can be seen as a step-by-step elimination of all the variables. Any sequence of optimization steps corresponds to "parsing" the interaction graph according to some rewriting rules [9]. In a sense, the operations to be taken at every step represent the semantic meaning attached to the parsing.

In general, the *order* in which the variables are eliminated is very important in determining the amount of computing time and storage required: usually the dimension of the step with maximal dimension must be minimized. Thus a new optimization problem, which is called the *secondary optimization problem*, arises. Various methods exist for its solution [7,8]. However, for most of our applications, this problem can be optimally solved with simple rules. Some examples are given in Section 6.

Fig. 1. (a) An interaction graph. (b) The same graph after elimination of vertex V_2.

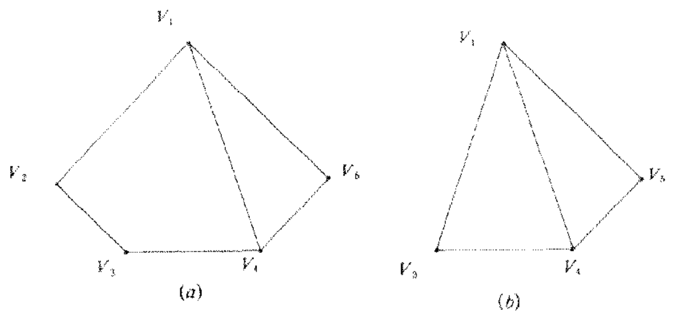


Fig. 2. The interaction graph of the FOM (4).

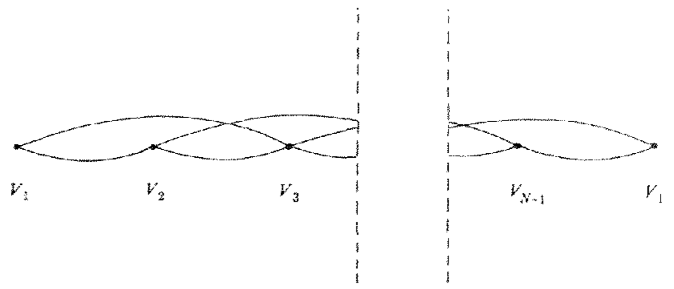


Fig. 3. Optimization process for finding a low-curvature curve of length 4. (a) Given image. (b) Interaction graph. (c), (d), (e) Interaction graph after elimination of the variables z_1, z_1 and z_2, z_1z_2 and z_3, respectively. (f) Optimal line.

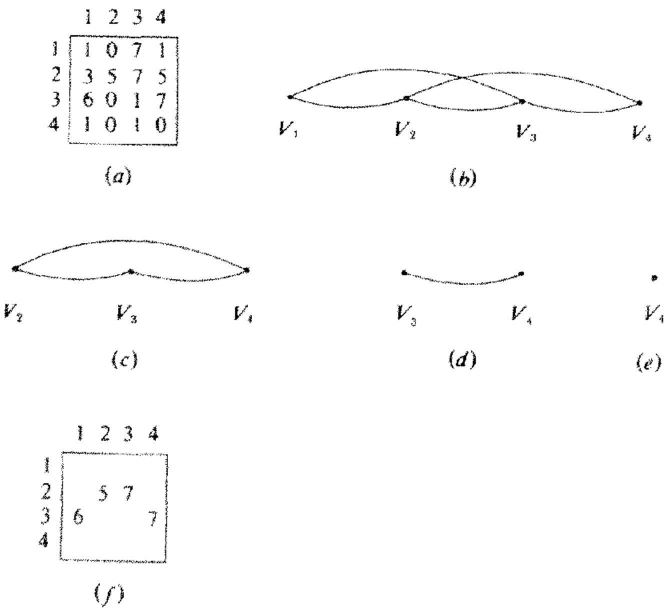


Table II

$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 1 & 0 & 7 & 1 \\ 2 & 3 & 5 & 7 & 5 \\ 3 & 6 & 0 & 1 & 7 \\ 4 & 1 & 0 & 1 & 0 \end{array}$$

$z_2 = 2,3$

$z_3 = 2,2$

z_1	(a)	(b)	(c)
2,4	5	0	5
1,4	1	1	0
3,4	7	1	6

(d)	(e)
6	3,4

Table III

$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 1 & 0 & 7 & 1 \\ 2 & 3 & 5 & 7 & 5 \\ 3 & 6 & 0 & 1 & 7 \\ 4 & 1 & 0 & 1 & 0 \end{array}$$

$z_3 = 2,2$

$z_4 = 3,1$

z_2	(p)	(a)	(b)	(c)
1,3	0	7	0	7
1,2	$-\infty$	0	1	$-\infty$
2,3	6	7	1	12

(d)	(e)
12	2,3

Table IV

$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 1 & 0 & 7 & 1 \\ 2 & 3 & 5 & 7 & 5 \\ 3 & 6 & 0 & 1 & 7 \\ 4 & 1 & 0 & 1 & 0 \end{array}$$

$z_4 = 3,1$

z_3	(p)	(a)	(c)
2,2	12	5	17
2,1	6	3	9
4,1	$-\infty$	1	$-\infty$
4,2	6	0	6
3,2	13	0	13

(d)	(e)
17	2,2

Table V

z_4	(p)	(a)	(c)
1,1	17	1	18
1,2	17	0	17
1,3	12	7	19
1,4	16	1	17
2,1	18	3	21
2,2	12	5	17
2,3	10	7	17
2,4	17	5	22
3,1	17	6	23
3,2	5	0	5
3,3	6	1	7
3,4	16	7	23
4,1	17	1	18
4,2	14	0	14
4,3	17	1	18
4,4	19	0	19

(d)	(e)
23	3,1

3. An Algorithm for the Optimal Detection of a Low-Curvature Curve

In this and in the next two sections we are concerned with the problem of extracting from a picture one line of fixed length which is optimal according to some given FOM. The underlying idea is that it is easier and more flexible to embed the heuristic of the problem in an FOM than in the recognition algorithm itself. We will first take a particular FOM and then generalize to a wider class.

The picture is entered in the form of a rectangular array $[a_{ij}]$, $i = 1, \cdots, r$, $j = 1, \cdots, c$. The value a_{ij} gives the gray level or optical density of the point of integer coordinates i and j. Two points (i, j) and (r, s) are *neighbors* if

$$\max(|i - r|, |j - s|) = 1. \qquad (3)$$

With this definition, any point which is not in the first or last row or column has exactly 8 neighbors. A *curve* is defined as any sequence of points $P_1, \cdots, P_N$ such that P_k and P_{k+1}, $k = 1, \cdots, N - 1$, are neighbors. Given any point of a curve, in order to determine the successive point, it is sufficient to give an octal number. If the correspondence is regular, this number gives the discrete *slope* of the curve; thus the difference (mod 8) between successive values of the slope gives the *curvature* of the curve. Now let us assume that our goal is to recognize a low-curvature black curve hidden in both additive and subtractive noise. A good figure of merit could be, for instance, the sum of gray levels along the curve, minus the sum of the curvatures at every point.[3] That is, if $z_i = (x_i, y_i)$ are the coordinate vectors of the points of the curve, the figure of merit is

$$g(z_1, \cdots, z_N) = \sum_{i=1}^{N} a(z_i) - q\sum_{i=2}^{N-1} (d(z_{i+1}, z_i) - d(z_i, z_{i-1}) \bmod 8) \quad (4)$$

[2] The reduction method we have shown for (1) can be obviously interpreted as the reduction of a simple chain-like interaction graph.
[3] In this example, the length N of the curve is given. See Section 6 for the case in which N is determined by the optimization process itself.

Fig. 4(a). An example of a low curvature curve—the gray intensity is 5.

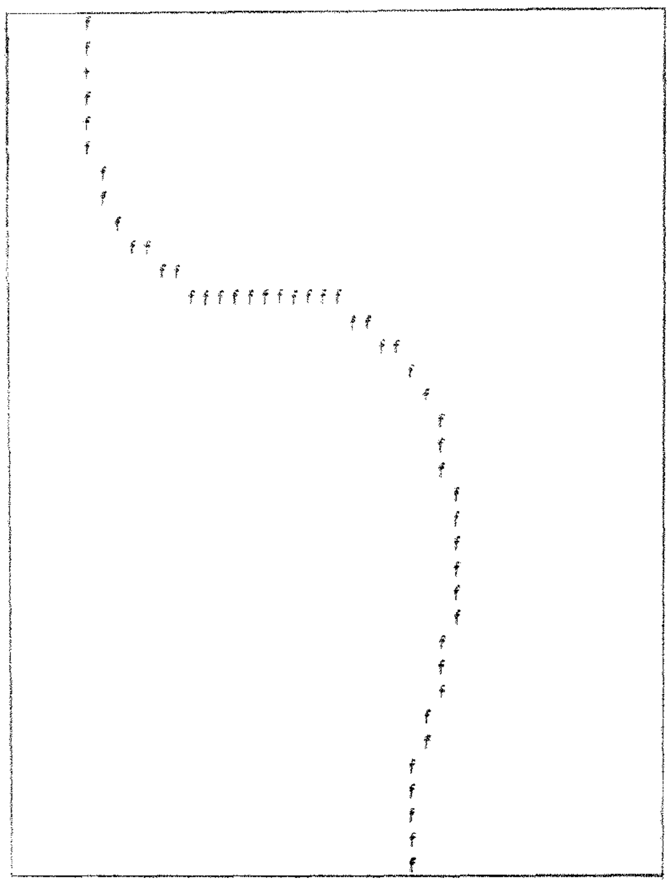
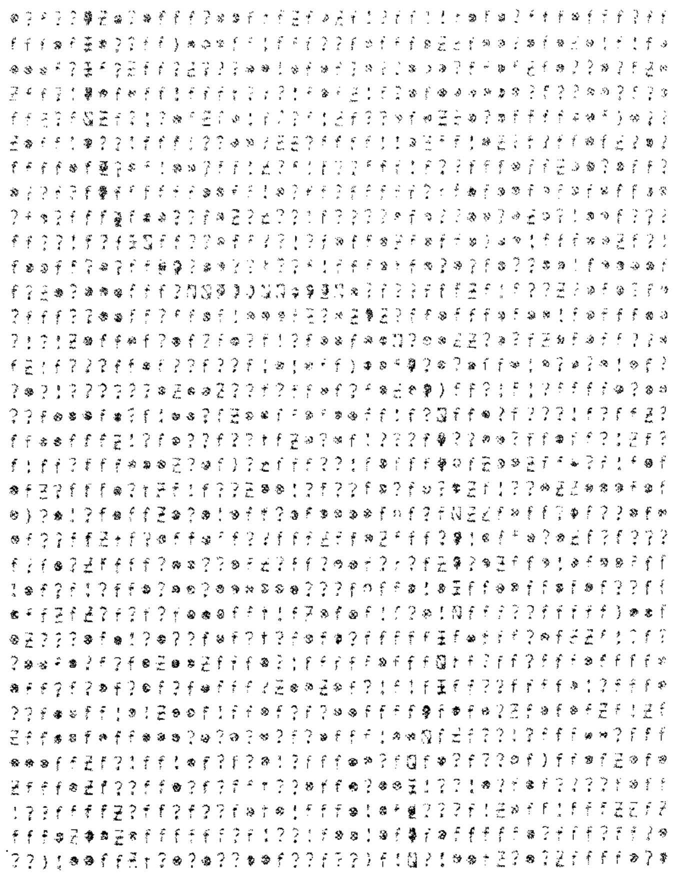


Fig. 4(b). The same curve of Figure 4(a) after addition of normal noise of mean value 5 and standard deviation 1.



with the constraints

$$\max\left(|x_{i+1} - x_i|, |y_{i+1} - y_i|\right) = 1,$$
$$(d(z_{i+1}, z_i) - d(z_i, z_{i-1}) \bmod 8) \leq 1.$$

In formula (4), $a(z_i)$ are gray levels and $d(z_{i+1}, z_i)$ is the slope of the curve between points P_i and P_{i+1}. As explained in Section 2, the constraints can be thought of as embedded in the FOM. The interaction graph for this problem is shown in Figure 2. Figure 3 (a) gives a simple picture of 4 rows and 4 columns which we will use as a running example. A curve with 4 points is sought, and thus Figure 3(b) gives the interaction graph. The coefficient q is assumed to be 1.

In the multistage optimization process the variables are eliminated starting from z_1 to z_N. Let us consider the first step, namely the elimination of vertex V_1. The variable z_1 interacts with variables z_2 and z_3. Thus the output of the first step will be a table which for every value of z_2 and z_3 gives (i) the optimal sum of all the terms of the merit function which contain the variable z_1, and (ii) a value of z_1 for which the optimum is achieved. Note that the number of entries of this table will be $8rc$ because only 8 values of z_3 for each value of z_2 satisfy the first constraint.[4] For each entry of the table, the values of z_1 to be considered in the optimization are only 3, for the second constraint of the problem. In Table II we can follow the computation of one entry of the table. For $z_2 = (2,3)$ and $z_3 = (2,2)$, we have:

(a) the value of the term $a(z_1)$ for all admissible z_1;
(b) the value of the term $q(d(f_3 - f_2) - d(f_2 - f_1) \bmod 8)$ for all admissible z_1;
(c) the difference between (a) and (b);
(d) the maximal value of (c) with respect to z_1;
(e) a value[5] of z_1 for which (d) is obtained.

Figure 3 (c) is the new interaction graph.

[4] Actually, they are less than 8 for border effects.
[5] In this and in the other stages the maximal value (d) must be unique, but it can be attained for many different values of z_1. This means that many curves have the same optimal cost.

Fig. 4(c). The same curve of Figure 4(a) after addition of normal noise of mean value 5 and standard deviation 2.

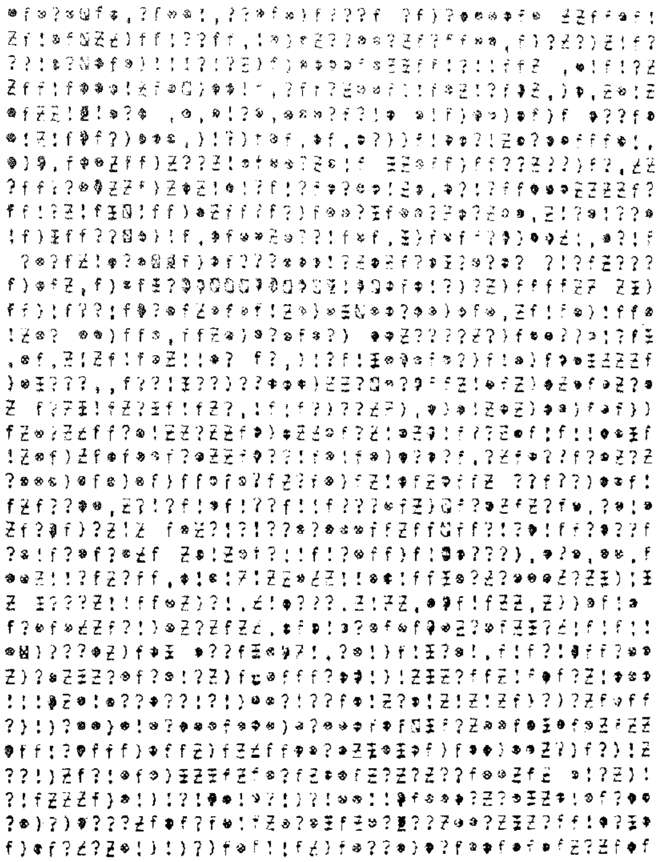
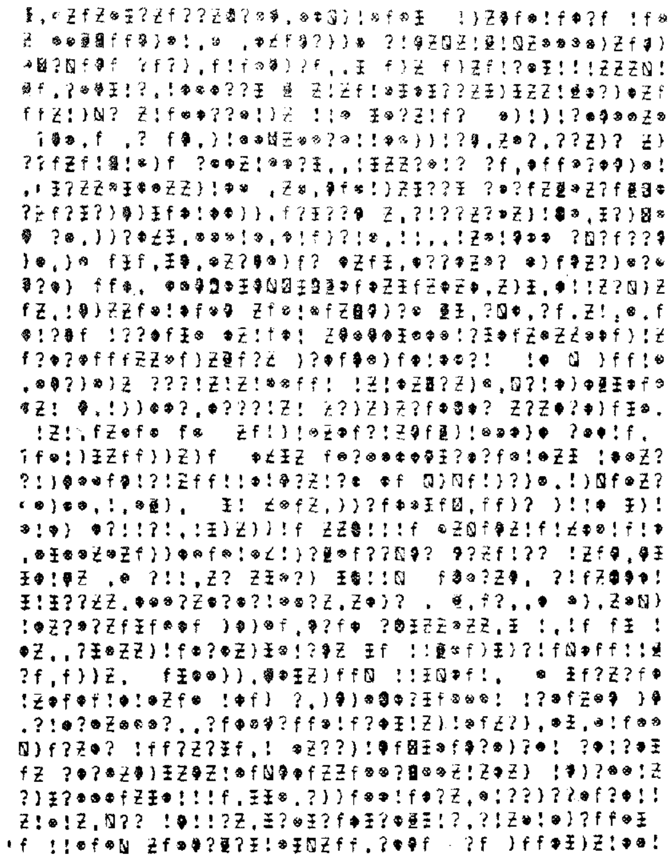


Fig. 4(d). The same curve of Figure 4(a) after addition of normal noise of mean value 5 and standard deviation 3.



Let us now consider the elimination of the kth variable, $k = 2, \cdots, N - 2$. A table is available which gives, for every value of z_k and z_{k+1}, the optimal value (with respect to $z_1, \cdots, z_{k-1}$) of the sum of the terms of the original FOM which contain at least one variable in the set $\{z_1, \cdots, z_{k-1}\}$. Thus, as we saw in Section 2, the output of the previous $k - 1$ stages appears as a term in the merit function of the present stage. A new table is now computed in the usual way which gives, for every allowed value of z_{k+1} and z_{k+2}, the output of the first k stages and the optimal value of z_k. Table III gives, for $z_3 = (2,2)$ and $z_4 = (3,1)$,

(p) the value of the output of the previous stages for all admissible z_2;
(a) the value of the term $a(z_2)$ for all admissible z_2;
(b) the value of the term $q(d(z_4, z_3) - d(z_3, z_2) \bmod 8)$ for all admissible z_2;
(c) the value (p) + (a) − (b);
(d) the maximal value of (c) with respect to z_2;
(e) a value of z_2 for which (d) is achieved.

Figure 3(d) shows the interaction graph after the two first stages.

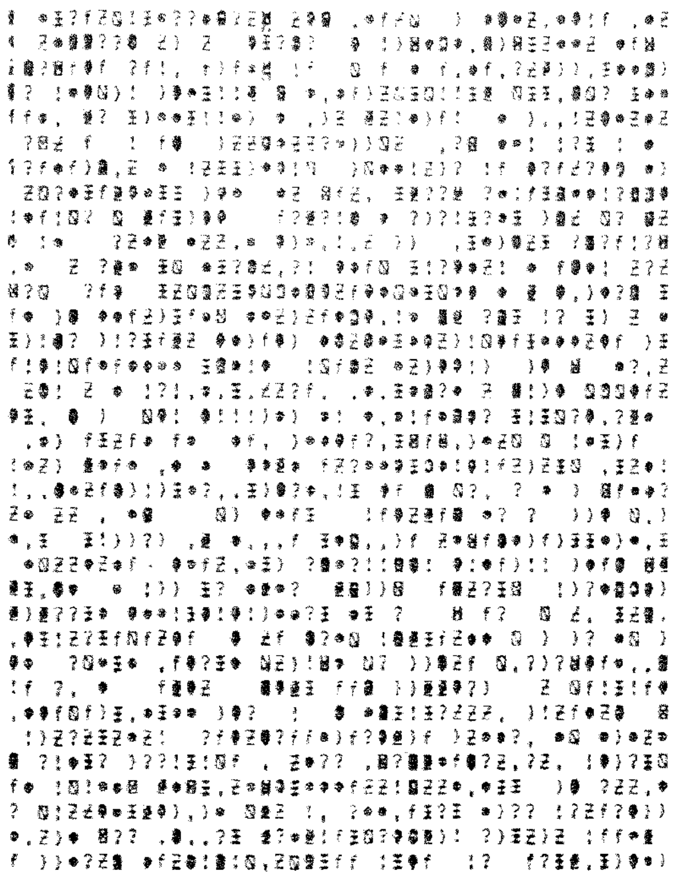
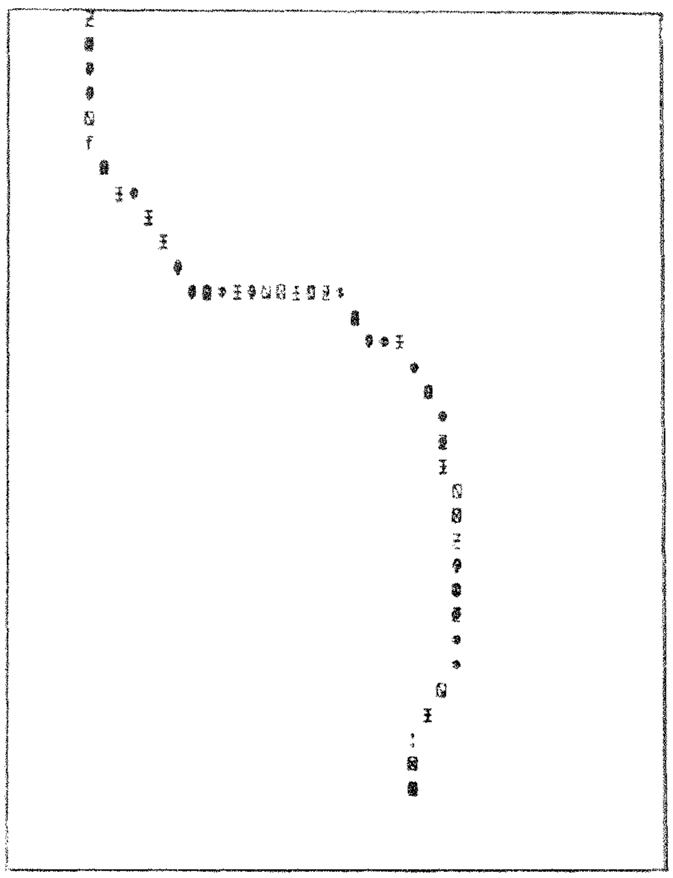
The last stages are in general slightly different because the dimension decreases toward zero. Table IV gives, for $z_4 = (3,1)$:

(p) the value of the output of the previous stages for all admissible z_3;
(a) the value of the term $a(z_3)$ for all admissible z_3;
(c) the sum (p) + (a);
(d) the maximal value of (c);
(e) a value of z_3 for which (d) is achieved.

Table V gives:

(p) the value of the output of the previous stages for all admissible z_4;
(a) the value of the term $a(z_4)$ for all z_4;
(c) the sum (p) + (a);
(d) the maximal value of (c);
(e) a value of z_4 for which (d) is achieved.

The table scanning is done as explained in Section 2; in our example, we know from Table V (e) that $z_4 = (3,1)$. Thus from Table IV(e) we have $z_3 = (2,2)$. Finally from Tables III(e) and II(e) we get $z_2 = (2,3)$ and $z_1 = (3,4)$. The optimal curve is shown in Figure 3(f).

4. Some Experimental Results

The algorithm explained in Section 3 was implemented in SAIL (an extension of ALGOL) on the PDP-10 of the Stanford Artificial Intelligence Project, and it is available from the author. For testing its performances, a noisy picture generator was programmed, using *Comm. ACM* Algorithms 266 [10] and 267 [11]. Pictures are represented with a 16-level gray scale, and are printed with superimposed characters for obtaining the impression of different gray levels through different percentages of blackened area.

Figure 4(a) is an example of a low-curvature line, whose points have a gray level of 5. This picture has 35 lines and 45 columns. In Figures 4(b), (c), (d), and (e) an independent amount of noise is added to every point. The statistical distribution of the noise at every point is normal, with mean value 5 and standard deviation $s = 1$, 2, 3, and 5, respectively. Note that the line is still clear in Figure 4(b), hardly recognizable in Figure 4(c) and quite unrecognizable in Figures 4(d) and 4(e). In Figure 5(a) we see the optimal line found in the image of Figure 4(d). The length is 45 and the weight factor q is equal to 5. Note that only small, local distortions have occurred. In Figures 5(b) and (c) we see the much worse case of Figure 4(e) for $N = 40$ and $q = 2$ and 5, respectively. Note that in this case the mean square value of the signal is locally equal to the mean square value of the noise. For bigger values of noise, the recognition becomes marginal and the found curve often turns on itself, repeating the most marked pieces.

The computing time for each curve was 10 minutes (5 minutes is sufficient if the trick of starting from both endpoints is used, see Section 5).

5. Some Classes of Figures of Merit for the Optimal Detection of One Curve

A simple extension of the FOM used in Section 3 is an FOM in which every variable interacts with k subsequent variables.[5] Such an FOM will be called of *type* k. For the algorithm of Section 3, we had $k = 2$. Note that $k \geq 1$ because at least the neighborhood con-

Fig. 5(b). Optimal curve detected in the noisy picture in Figure 4(e), length $N = 40$ curvature cost $q = 2$.

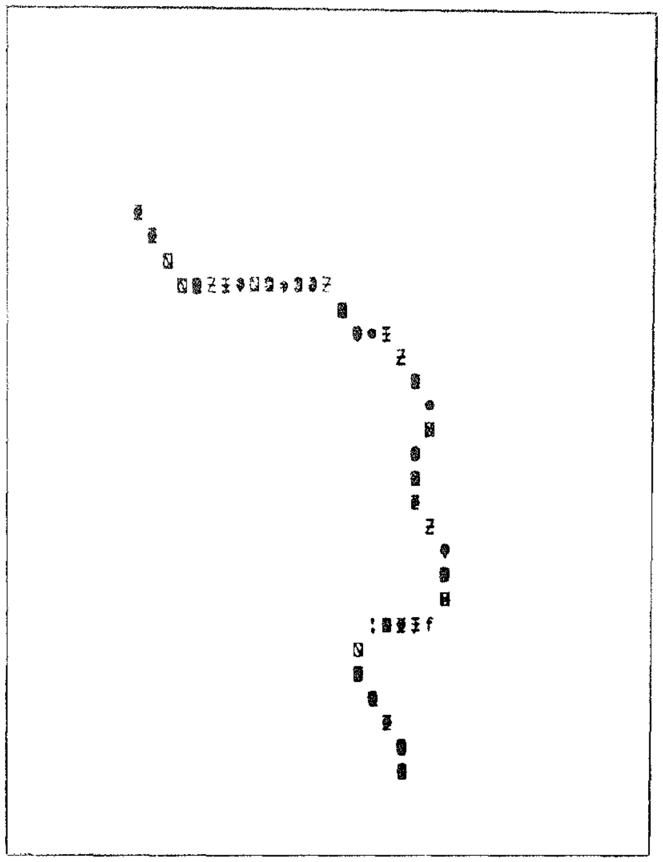
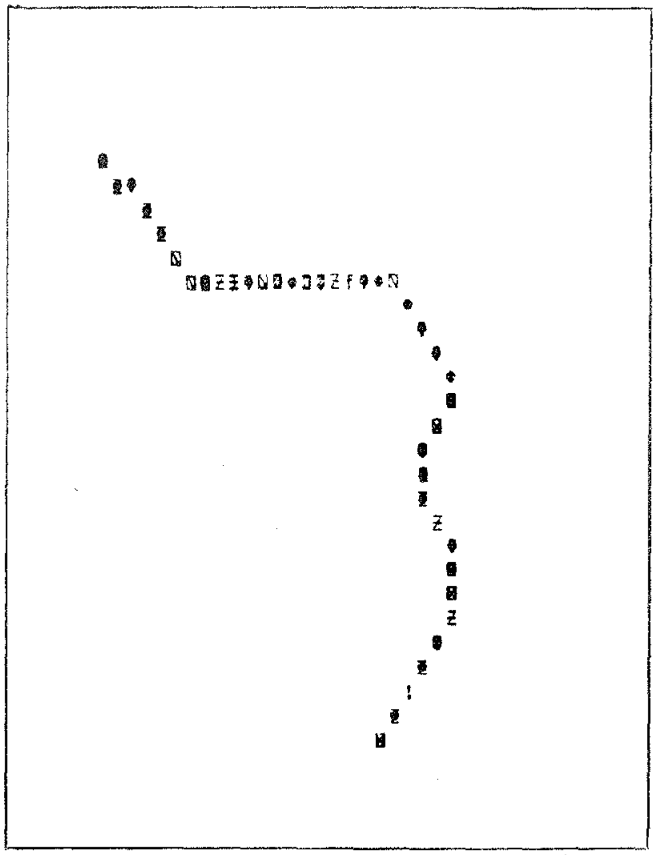


Fig. 5(c). Optimal curve detected in the noisy picture in Figure 4(e), length $N = 40$, curvature cost $q = 5$.



straint is always present. The form of an FOM of type k will be

$$g(z_1, \cdots, z_N) = \sum_{i=1}^{N} g_i(z_i, \cdots, z_{i+k-1}).$$

Note that the functions g_i can be different from stage to stage; this feature allows the user to construct an FOM which "looks for" curves of particular shape. For instance, if the character "5" must be found, a roughly horizontal straight segment should be sought first; then a sharp left turn, a straight vertical segment, a sharp left turn, and a smooth right turn should be found. These expectations, which correspond to a priori probabilities of a Bayesian decision process, can be built in the FOM giving high cost to unexpected features and vice versa. The type k of the figure of merit gives roughly the *length* of the features which can be sought. A proper weight can be given to the position in the picture and to the orientation of the features. Furthermore, no rigid specification of the features is required, because a different cost can be assigned to every possible deformation of them.

The storage and time requirements grow pretty fast with the type k. On the contrary, no substantial extra cost is paid for a complicated (and, it is hoped, more efficient) FOM of a given type. Roughly speaking, and taking into account the neighborhood constraint, the number of entries in the table of one intermediate stage is $F = rc8^{k-1}$ and the same is the number of partial optimizations required at every stage. Of course, F decreases if other constraints are present, e.g. on the curvature; this corresponds to limiting the variety of features of length k which we consider. F is also the amount of fast memory required by the entire process. The total number T of optimizations and the amount of slow storage required is instead $Nrc8^{k-1}$.[7] Note that this number grows linearly with N, while the number of curves among which the optimum is sought grows exponentially with N.

[6] Except the last k variables.

[7] Note that it is not necessary to use a computer word for every entry in the tables. In fact, the tables used in the final scanning contain the optimal variables only. Thus a few bits are usually sufficient for each entry.

This method is suitable for parallel computation; in fact, at every stage, the optimizations can be carried on at the same time for all the points of the image. Furthermore, only local storage is necessary, and for simple FOM's no complicated operations like multiplications or divisions are required.

Figures of merit which are more general than the class described above can be allowed without too big an increase in computational effort. For example, assume that an L-shaped curve is looked for, but that no exact location of the corner along the curve is known. The design of a suitable figure of merit can be sketched in the following way:
(1) the curve begins and ends with straight segments: thus in the first and last stages turns in both directions must be costly;
(2) a sharp left turn must become cheap in the central stages; however, one turn only must be present, so that if a turn has occurred in an early central stage, no turns must be present in successive central stages.

If the interaction graph of the above FOM is drawn, it is clear that every central vertex must be connected with all the precedent central vertices in order to know whether or not the turn has already occurred. As a result, the dimensionality of the central stages is very high, so that storage requirements are prohibitive. A quite different situation arises if the information about the turn is "added" to the decision variables of all the central stages; namely, the variables z_i of central stages will be $z_i = (x_i, y_i, b_i)$ where b_i is a Boolean variable which specifies whether or not the turn has already occurred. In this way the number of entries of the central tables will only be doubled, and no extra connections will be added to the interaction graph. This trick can easily be extended to the case in which the average direction or the average curvature of the entire segment of curve determined in the previous stages must be known.

In all the cases considered above every stage of the optimization process corresponds to the determination of one point of the best curve. However, in low-definition cases it can be convenient to determine at every stage an entire segment of the curve. If interaction with only the previous stage is allowed, this application seems to be close to Kovalewsky's method [6] for description of handprinted characters. The advantage of low-definition methods is essentially that bigger features can be considered without increase in dimensionality. In particular, elementary segments can have a bigger number of allowed values of the slope. On the contrary, no reduction in computing time can be expected because the reduced number of stages is balanced by the bigger number of decisions allowed at every stage.

A reduction in slow storage and computing time (as much as 50 percent) can be achieved if the FOM is symmetric with respect to the two extremes of the curve. In fact, in this case it is sufficient to find the best pair of optimal half-curves which smoothly concatenate. From the point of view of the interaction graph, it corresponds to start the elimination of vertices from both sides and to recognize that symmetrical stages will give exactly the same output.

If (a) the computing time or (b) the storage requirements become too heavy, approximate methods can be devised. In case (a), and if the curve is homogeneous, a suboptimal curve of length $N = pq$ can be computed as the optimal concatenation of p curves of length q. For instance, in the example of Section 3 a standard q-stage optimization process can determine, for each point of the picture and for each direction, the best curve of length q. Then a p-stage process can use as elementary segments the curves which are the output of the first process. In fact, inductively, if for each point and each direction we know the best chain of $(k - 1)$ segments, we can determine the best chain of k segments optimizing among the only three segments which can be smoothly added to the precedent chain. In conclusion, this approximate solution can be obtained in $p + q$ stages, while $N = pq$ stages are required for the exact solution. This trick can be iterated more than once; if at each stage we take as elementary segments the optimal curves of the previous stage, a reduction to $\log_2 N$ stages is achieved.

In case (b), the dimensionality of each stage and thus the storage can be reduced if an iterative method is used (see [5 pp. 78–87]). However, with iterative low-dimensional methods, only local maxima are found.

6. Recognition of Systems of Curves

In the previous sections we discussed the dynamic programming approach to the determination of one curve. In practice, it can happen that we want to recognize more constrained line-like figures, e.g. circle or an X-shaped or R-shaped pattern. The first step in our method is, as usual, the determination of a suitable figure of merit. However, more complicated constraints are now imposed by the structure of the image, so that the order of elimination of variables is not trivial. In the general case of a system of curves the interaction graph becomes a synthetic way of *describing* the picture. In a sense, the interaction graph carries information about the *topology* of the system of curves, while the rest of the FOM tells us the *geometrical* characteristics of the elementary curves. Note that "invisible links" can belong to the structure: for instance, two parallel straight lines can be well described by means of two curves with high cost on the local and average curvature, and with two perpendicular, "invisible" links of the same length at the extrema.

If the secondary optimization problem is very complicated, it can be solved [7] using a dynamic programming technique in the lattice of the 2^N subsets of the variables. This method is based on a theorem which states that after elimination of a set of variables, the remaining problem (i.e. its interaction graph) is not affected by the order in which the variables were eliminated. However, before we use this algorithm, the following rules (applied in order and as many times as possible) are assured to reduce the number of variables in an optimal way [8]. Let H_v be the subgraph of the interaction graph G whose vertices are the vertices connected to the vertex V in G; then
(a) if H_v is complete, erase V;
(b) if the vertex set of H_v has cardinality 2, erase V.

Note that if H_v has cardinality one, rule (a) always applies; thus it allows the solution of all the interaction graphs which are trees. On the other hand, rule (b) allows simplification of the chains of serially connected arcs and vertices. Therefore it solves series-parallel graphs. The combined use of rules (a) and (b) allows us to solve most practical cases.

In this paper we have always considered the interaction graph as derived from the FOM and thus as a datum. One could ask if the structure of the FOM (namely, the interaction graph itself) also can become the object of an optimization process. The simplest case happens when we are looking for a curve but do not know its length. This situation also can be reduced to the scheme we have followed. In fact, it is possible to introduce a new variable y whose value is the number of stages of the process. Of course, it is also wise to normalize the FOM with respect to the length of the curve. In the interaction graph, the new vertex is connected with all the other vertices. However, in this particular case little complexity is added if y is the last variable to be eliminated. In fact, if we consider the table corresponding to the stage k, a new dimension corresponding to the variable y is added. However, for $y > k$ the table does not depend on y and is equal to the table we had in the case of the curve of fixed length, and for $y \leq k$ the table depends on y only because variables x_i, $i > k$, do not contribute to the FOM. In the general case, however, the addition of control variables can increase the dimensionality of the problem.

7. Conclusion

We have presented a global approach to the problem of finding systems of curves in noisy pictures. This method allows us to give a definition, by means of an FOM, of what we mean by a "good curve"; then the best curve is found. The globality of the method, a peculiar feature, also makes it flexible and good in otherwise unmanageable situations. For instance, large gaps can only be bridged by using global context. This approach has some drawbacks. First, it uses a computing time sensibly greater than do simple local preprocessing techniques. However, approximate implementations of the optimization process and less definition in the picture could allow competitive times still preserving globality at some extent. Second, storage requirements are high. Finally, many global specifications are very expensive for the optimization algorithm; for instance, the constraint that the found curve is simple (non-self-intersecting) is almost impossible to embed in an FOM. Many trials (perhaps interactively performed) are then sometimes required for finding a suitable FOM for a given class.

Received July 1970; revised December 1970

References

1. Rosenfeld, A., et al. Edge and curve enhancement in digital pictures. Tech. Rep. 69-93, U. of Maryland, Computer Science Ctr., College Park, Md., May 1969.
2. Graham, D. N. Image transmission by two-dimensional contour coding. *Proc. IEEE. 55*, 3 (Mar. 1967), 336–346.
3. Moharir, P. S., and Prasada, B. Two-dimensional picture, processing. In *Picture Bandwidth Compression*, T. S. Huang and O. J. Tretiak, (Eds.), Gordon Breach, New York, 1969.
4. Grasselli, A. On the automatic classification of fingerprints. In *International Conference on Methodologies of Pattern Recognition*, S. Watanabe (Ed.), Academic Press, New York, 1969.
5. Bellman, R. and Dreyfus, S. *Applied Dynamic Programming*. Princeton U. Press, Princeton, N.J., 1962.
6. Kovalewsky, V. A. Sequential optimization in pattern recognition and pattern description. Proc. IFIP Cong., 1968, North-Holland Pub. Co., Amsterdam, pp. 146–151.
7. Brioschi, F., and Even, S. Minimizing the number of operations in certain discrete variable optimization problems. *Operations Res. 18*, 1 (Jan.–Feb. 1970), 66–81.
8. Bertelē, U., and Brioschi, F. A new algorithm for the solution of the secondary optimization problem in nonserial dynamic programming. *J. Math. Analysis and Applications 27*, 3 (1969), 565–574.
9. Montanari, U. Separable graphs, planar graphs and Web grammars. *Information and Control 16*, 3, (May 1970), 243–267.
10. Pike, M. C., and Hill, I. D. Algorithm 266. Pseudo-random numbers [G5]. *Comm. ACM 8*, 10 (Oct. 1965), 605–606.
11. Pike, M. C. Algorithm 267. Random Normal Deviate [G5]. *Comm. ACM 8*, 10 (Oct. 1965), 606.