[4] K. D. Senne and R. S. Bucy, "Digital realization of optimal discrete-time nonlinear estimators," in *Proc. 4th Annu. Princeton Conf. Syst. Sci.*, 1970.

[5] R. S. Bucy and K. D. Senne, "Digital synthesis of non-linear filters," *Automatica*, vol. 7, pp. 287–298, 1971.

[6] E. C. Tacker and T. D. Linton, "Digital and hybrid simulation of a discrete-time optimal nonlinear filter," in *Proc. 4th Hawaii Int. Conf. Syst. Sci.*, 1971, pp. 465–467.

[7] ——, "Digital and hybrid simulation of a Bayes-optimal nonlinear filter," Studies in Digital Automata, Louisiana State Univ., Baton Rouge, Air Force Office of Scientific Research Contract F-44620-68-C-0021, Tech. Rep. LSU-T-TR-40, Sept. 1970.

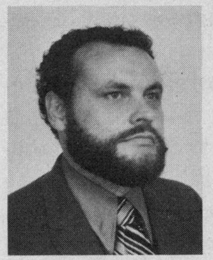
[8] R. E. Mortensen, "Mathematical problems of modeling stochastic nonlinear dynamic systems," *J. Statist. Phys.*, vol. 1, no. 2, 1969.

[9] C. W. Helstrom, "Markov processes and applications," in *Communication Theory*, A. V. Balakrishnan, Ed. New York: McGraw-Hill, 1968.

[10] R. S. Bucy, M. J. Merritt, and D. S. Miller, "Hybrid computer synthesis of optimal discrete nonlinear dynamic systems," in *Proc. 2nd Symp. Nonlinear Estimation Theory and Its Applications* (San Diego, Calif.), 1971.

[11] ——, "Hybrid synthesis of the optimal discrete nonlinear filter," *Stochastics*, vol. 1, Jan. 1973.

[12] D. S. Miller, Ph.D. dissertation, Dep. Elec. Eng., Univ. Southern California, Los Angeles, 1972.

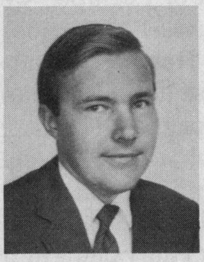


Edgar C. Tacker (S'59–M'64) was born in Savannah, Tenn., on September 26, 1935. He received the B.S. degree (with distinction) in electrical engineering from the University of Oklahoma, Norman, in 1960, the M.S. degree in electrical engineering from New York University, New York, N. Y., in 1962, and the Ph.D. degree from the University of Florida, Gainesville, in 1964.

His industrial experience includes two years as a Systems Engineer with Bell Telephone Laboratories, Inc. He is presently an Associate Professor in the Departments of Electrical and Chemical Engineering, Louisiana State University, Baton Rouge. He has developed graduate courses in applied functional analysis, systems science, and digital and hybrid computation at LSU and Michigan State University, East Lansing. His current research interests are in the areas of stochastic control theory and multilevel system theory, with emphasis on computational aspects. He has applied these theories to problems involving the control of chemical processes and interconnected electrical energy systems.

Dr. Tacker is a member of Tau Beta Pi, Pi Mu Epsilon, and Eta Kappa Nu.

❖



Thomas D. Linton was born in Frost, Ohio, on November 9, 1944. He received the B.S. degree in electrical engineering from the University of Arkansas, Fayetteville, in 1967, and the M.S. degree in systems science from Michigan State University, East Lansing, in 1969.

He is presently working toward the Ph.D. degree in the Department of Electrical Engineering, Louisiana State University, Baton Rouge.

Mr. Linton is a member of Eta Kappa Nu and Phi Kappa Phi.

The Representation and Matching of Pictorial Structures

MARTIN A. FISCHLER AND ROBERT A. ELSCHLAGER

Abstract—**The primary problem dealt with in this paper is the following. Given some description of a visual object, find that object in an actual photograph. Part of the solution to this problem is the specification of a descriptive scheme, and a metric on which to base the decision of "goodness" of matching or detection.**

We offer a combined descriptive scheme and decision metric which is general, intuitively satisfying, and which has led to promising experimental results. We also present an algorithm which takes the above descriptions, together with a matrix representing the intensities of the actual photograph, and then finds the described object in the matrix. The algorithm uses a procedure similar to dynamic programming in order to cut down on the vast amount of computation otherwise necessary.

One desirable feature of the approach is its generality. A new programming system does not need to be written for every new description; instead, one just specifies descriptions in terms of a certain set of primitives and parameters.

There are many areas of application: scene analysis and description, map matching for navigation and guidance, optical tracking, stereo compilation, and image change detection. In fact, the ability to describe, match, and register scenes is basic for almost any image processing task.

Index Terms—**Dynamic programming, heuristic optimization, picture description, picture matching, picture processing, representation.**

Introduction

THE PRIMARY PROBLEM dealt with in this paper is the following. Given some description of a visual object, find that object in an actual photograph. The object might be simple, such as a line, or complicated, such as an ocean wave, and the description can be linguistic, pictorial, procedural, etc. The actual photograph will be called the "sensed scene," a two-dimensional array of gray-level values, while the object being sought is called the "reference."

This ability to find a reference in a sensed scene, or, equivalently, to match or register the images of two scenes, is basic for almost any image processing task. Application to such areas as scene analysis and description, map matching for navigation and guidance, optical

Manuscript received November 30, 1971; revised May 22, 1972, and August 21, 1972.

The authors are with the Lockheed Palo Alto Research Laboratory, Lockheed Missiles & Space Company, Inc., Palo Alto, Calif. 94304.

tracking, stereo compilation, and image change detection is direct and obvious.

There are two basic approaches to solving the image-matching problem as defined above.

If we possess a precise description of the noise and distortion process which defines the mapping between the reference and its image in the sensed scene, we can employ statistical decision theory techniques to derive an image-matching procedure which optimizes some objective criterion (e.g., minimum error, or minimum risk, in determining the best embedding of the reference in the sensed scene). A typical outcome of such an analysis is the use of a correlation-like matching procedure. However, in most practical problem situations, the required noise and distortion model is not available, nor is it feasible to attempt to construct one. For example, we might consider all human faces to be perturbed versions of some single ideal or reference face. However, to attempt to define completely a valid noise and distortion model for this situation would be a hopeless task.

A second and more general approach to the image-matching problem bypasses the need for a noise and distortion model by accepting an embedding metric without requiring its justification as being equivalent to a minimum error (or minimum risk) procedure. In fact, without a noise and distortion model, there is no theoretically valid way to derive or predict the error performance of a selected procedure prior to its actual application. Our primary concern in this paper is with this latter case.

We offer the following two sets of criteria for an embedding metric not theoretically derived on the basis of error performance. First, it must be successful in application (i.e., its observed error performance must be acceptable), intuitively satisfying (so we can have some confidence in its ability to deal with as yet untried applications), and general enough so that it can be employed over a wide range of problems without significant modification. Second, it must be possible to specify a computationally feasible decision algorithm which selects a suitable embedding based on the given embedding metric. (The combination of embedding metric and corresponding decision algorithm will be called an embedding model.) In the remainder of this paper, we will present an embedding metric and corresponding decision algorithm, present examples of the application of this embedding model to a number of distinct problem areas, and show how this embedding model is relevant to the problem of scene representation as well as to image matching.

An Embedding Metric

To introduce the generic form of our embedding metric, and establish its intuitive validity, let us first consider the following process. Assume that the reference is an image on a transparent rubber sheet. We move this sheet over the sensed image and, at each possible placement, we pull or push on the rubber sheet to get the best possible alignment between the reference image on the sheet and the underlying sensed image. We evaluate each such embedding both by how good a correspondence we were able to obtain and by how much pushing and pulling we had to exert to obtain it.

Let us now consider a discrete version of the above process which is both more precise and more reasonable from an implementation standpoint. In a specific application, we might have some information on the range of permissible distortions that can occur between the reference and sensed images. For instance, some subset of the items appearing in the sensed image might always retain their internal shape even though their relative positions might be subject to change with respect to their locations in the reference scene. Further, where change of relative position is possible, we might be able to bound the extent of such change; and, finally, we might like to assign variable "costs" to the different types of change of relative position or relative change in some nongeometric attribute.

To achieve these capabilities, we replace the rubber sheet by a reference image which is composed of a number of rigid pieces (components) held together by "springs." A rigid piece of the reference image can be as small as a single resolution cell, or as large as the entire reference image, and corresponds to a single coherent entity in the reference image. The springs joining the rigid pieces serve both to constrain relative movement and to measure the "cost" of the movement by how much they are "stretched." (Typically, the springs will be highly nonlinear in their behavior.) In determining the cost of an embedding, we measure the "tension" on each spring (the tension can be a function of direction as well as stretch or even a relative change in some locally defined attribute), and also make a local evaluation of how well each coherent piece is embedded as an independent entity.

The above model permits two interesting dichotomies. The first dichotomy is the separation of "syntactic" and "semantic" information. The semantic information, which is application dependent, is embodied in the specific partitioning of the reference into coherent pieces, the placement and cost functions assigned to the springs, and the cost functions associated with the independent embedding of the coherent pieces. The syntactic information, which is relatively independent of the particular application, defines the class of descriptions which the algorithm can process. These data are embodied in the limits set on reference decomposition (e.g., number and maximum size of pieces, etc.); in the formats which must be employed to specify the global constraints and costs; and in the form of the embedding metric which evaluates "global" fit. The separation of semantic and syntactic information is essential to permit application of the model to a broad range of problem areas without the necessity of making significant changes in the implementation.

The second dichotomy is the separation between the local and global evaluation functions. The global evaluation function, associated with the relative positioning of the coherent pieces as described previously, has strong syntactic controls on its form to permit its integration directly into the decision algorithm. This is important because the global evaluation produces the most severe combinatorial problems. A local evaluation function, associated with how well a given coherent piece is independently embedded, is easily changed from problem to problem (based on problem-dependent considerations) without requiring any change in the core algorithms. Thus, the form of a local evaluation function can be a (conventional) correlation function together with a pictorial reference component, or a procedure based on linguistic concepts together with a formal description of a reference component,[1] or even a series of guesses inserted interactively by a human evaluator. The decoupling of the local evaluation functions from the core algorithms provides a great deal of flexibility in making changes or improvements in the evaluation functions for a given problem, as well as when switching from problem to problem. Further, because of the above separation, the performance of the algorithms (both local and global) can be independently evaluated in a direct and intuitively obvious manner. Such an evaluation then permits iterative improvement in performance by selective alteration in the problem-dependent options.

We are now in a position to present formally the proposed embedding metric. Let the reference be composed of p components (i.e., p coherent, or primitive, pieces). For $1 \leq i \leq p$, let x_i be a variable ranging over the set of all locations of the sensed scene. x_i is defined to be the postion of the ith component. Suppose there is a mechanism, either a computer program, or possibly a person, or some mechanical device, which, for location x_i of the ith component, outputs a numerical value $l_i(x_i)$ that indicates how strongly the ith component fits at location x_i of the sensed scene. The smaller $l_i(x_i)$, the better the fit.

While not formally required, the intent is that $l_i(x_i)$ measure the presence of the ith component at a location in the sensed scene independent of any knowledge of the locations of the other components. That is, $l_i(x_i)$ is a purely local and possibly imprecise measure of the presence of the ith component at location x_i.

In addition to the purely local measure l_i, $1 \leq i \leq p$, there are the following considerations: 1) how well the different components are situated in the required spatial relations to each other; and 2) how *relative* values of attributes of the components compare with the corresponding measured values in the sensed image (e.g., we might want to specify that the ith component be thicker and more greenish than the jth component). The extent to which the above specifications are not satisfied is reflected in the "stretching" of the springs between the corresponding components.

Each location in the sensed image can be associated with a two-dimensional vector (e.g., the components of the vector can be the row and column number of the location in the sensed scene). In that case, $x_i - x_j$ (usual vector subtraction) is a vector pointing from x_j to x_i. We can now let $g_{ij}(x_i, x_j) = g_{ij}(x_i - x_j)$ be the cost associated with the spring joining the ith and jth components. If there is no spring between these components, then g_{ij} is identically zero.

If we set $g_{ij}(x_i, x_j) = l_i(x_i)$ when $i = j$; and let $X_i = \{x_i, x_2, \cdots, x_i\}$, then the total cost of embedding p components at locations X_p is $G(X_p)$.

$$G(X_p) = \sum_{i=1}^{p} \sum_{j=1}^{i} g_{ij}(x_i, x_j). \tag{1}$$

Expression (1) can also be written as

$$G(X_p) = \sum_{i=1}^{p} h_i(X_i) \tag{2}$$

where

$$h_i(X_i) = \sum_{j=1}^{i} g_{ij}(x_i, x_j).$$

$h_i(X_i)$ can be thought of as the cost of embedding the ith component at location x_i, given that the previous $i-1$ components are at the locations specified by X_i.

Computational Procedures

In this section of the paper, we will present computational procedures for locating a suitable embedding of one image in another, based on the embedding metric just presented. A discussion of dynamic programing (DP) is included to place our proposed algorithm [the "linear embedding algorithm" (LEA)] in proper perspective. In particular, a generic (but computationally impractical) approach to solving the embedding problem is some form of DP. The specific form of our embedding metric permits a simplification of the general DP formulation, and the LEA is offered as a computationally feasible approximation to this restricted DP formulation. A graph theoretic interpretation is included to provide a better intuitive appreciation of the LEA in relation to DP.

Let us assume that the sensed image, designated by the abbreviation SM, is composed of M resolution elements; while the reference, designated by the abbreviation RM, is composed of P pictorially defined components (coherent pieces) with a total of $N = \sum_{i=1}^{P} n_i$ resolution elements, n_i being the number of resolution elements in the ith component.

The most direct procedure for locating a best embedding is to select combinationally N resolution elements at a time from the SM, determine if each such selection satisfies the coherent (intracomponent) and

[1] Note that we are now further generalizing the concept of "component." It no longer has to be a rigid entity defined pictorially, but rather may be any information structure or decision procedure which can be used to define a real-valued function whose domain of definition is the set of all locations in the sensed image.

global (intercomponent) constraints, and, if acceptable, then evaluate the embedding metric for the given selection. Obviously, such an approach is completely impractical even for small pictures. For example, a 50×30 SM ($M=1500$) and a 5×5 RM ($N=25$) would require more than 10^{54} selections and evaluations, a hopelessly large number. If we assume that the coherent and relational constraints provide us with $P=6$ nonoverlapping components, each component sequentially constrained to stay within $w=10$ locations referenced to the location of the previously placed component, then we would still be required to perform on the order of $1500 \times 10^5 = 1.5 \times 10^8$ evaluations. Assuming 10^{-3} s per evaluation, we would require 1.5×10^5 s or approximately two days of computation time. It is thus obvious that a more effective technique is required.

Dynamic Programming

Expressed in formal terms, the evaluation of the embedding metric for a typical picture results in a nonlinear, integer programming problem with local optima different from the global optimum (i.e., no particular regularity, such as unimodality). The only available class of computational procedures for finding the global optimum under the above conditions (other than the exhaustive search techniques discussed earlier) is usually designated by the generic name dynamic programming.

DP is a multistage or iterative optimization procedure which can be described in general terms as follows (see [6] and [7]). We wish to find

$$\min_X G(X) = \sum_{i \in I} h_i(X^i) \qquad (3)$$

where $X = \{x_1, x_2, \cdots, x_p\}$, each x_i, $1 \leq i \leq p$, ranges over a set of vectors with discrete components, $I = \{1, 2, \cdots, p\}$, and X^i is the set of those variables (among X) upon which h_i depends. $\{h_i\}$, $1 \leq i \leq p$, are a given set of real-valued functions.[2]

Let $|X^i|$ be the number of variables in X^i, and let each x_i, $1 \leq i \leq p$, range over M values. Then each component $h_i(X^i)$ of the cost function is specified by means of a table with $|X^i|+1$ columns and $M^{|X^i|}$ rows.

The solution proceeds as follows. We select a variable $y_1 \in X$ and compute the following expressions (this gives us the minimization of G with respect to y_1):

$$f_1(\Gamma(y_1)) = \min_{y_1} \sum_{i \in I_1} h_i(X^i) \qquad (4)$$

$$y_1^*(\Gamma(y_1)) = \text{the value of } y_1 \text{ which minimizes expression (4)} \qquad (5)$$

where $\Gamma(y_1)$ is the set of all those variables (except y_1) which occur in any one of those X^i which contains y_1. In other words, $\Gamma(y_1)$ is the set of variables which interacts with y_1. I_1 is the set of those i such that X^i contains y_1. y_1^* is the optimizing assignment for y_1 as a function of the variables of $\Gamma(y_1)$.

The operation described by (4) is called the elimination of the variable y_1 and results in the following transformation of (3):

$$\min_X G(X) = \min_{X-\{y_1\}} \left[f_1(\Gamma(y_1)) + \sum_{i \in [I-I_1]} h_i(X^i) \right]. \qquad (6)$$

Expression (6) has the same form as (3), but does not contain y_1. Thus, we can find an optimal assignment for X be sequentially "eliminating" all of the variables, and then tracing back through the stored tables of $y_i^*(\Gamma(y_i))$, where $\Gamma(y_i)$ can only contain y_j such that $j > i$. That is, we must eventually reach a point where, for some s,

$$\bigcup_{j=1}^{s} I_j = I$$

and expression (6) has the form

$$\min_X G(X) = \min_{\{y_{s+1}, \cdots, y_p\}} [f_s(\overbrace{y_{s+1}, \cdots, y_p}^{\Gamma(y_s)})]. \qquad (7)$$

From (7) we can directly determine the global minimum cost for G and also the optimizing values for $Y = (y_s, y_{s+1}, \cdots, y_p)$. Given the value for Y, we can determine the value of y_{s-1}^* from the stored table for $y_{s-1}^*(\Gamma(y_{s-1}))$, as indicated in expression (5). This "backward" recursive process is continued to provide us with the complete optimizing assignment for X.

The computational feasibility of the DP approach depends on storage and computing time requirements. For a given objective function (in our case, the embedding metric), storage and computing time requirements are a function of the order in which the variables are eliminated [3], [7] and the dimensionality of each of the eliminated variables. We will say that variable y_i has dimensionality $|\Gamma(y_i)|$, where $|\Gamma(y_i)|$ is the number of variables in the set $\Gamma(y_i)$ as defined following (4) and (5). For many of the problems we shall be concerned with, the dimensionality will be essentially constant over all variables (i.e., a constant number of springs attached to each component and a symmetric interconnection topology) and relatively independent of order of elimination. Let us associate the dimensionality of the variables with the embedding function itself employing the designation $D(G)$ to denote the maximum dimensionality of any of the variables to be eliminated for a given order of elimination.

Where the dimensionality of all variables is constant for a given order of elimination, the complete embedding procedure will involve the iterative application of (4) and (5) $[p-D(G)]$ times, to evaluate the p arguments of G corresponding to the embedding of the p components. In the kth iteration,[3] we compute and store a

[2] The $\{h_i\}$ defined in (2) are independent of all x_j for $j > i$; the $\{h_i\}$ defined in the general DP formulation can be a function of any x_j.

[3] The kth iteration evaluates the "cost" of embedding the kth component at y_k^*, given some specific embedding of the components associated with the variables in $\Gamma(y_k)$. This evaluation corresponds to the elimination of y_k.

table for f_k and y_k^*. The number of lookups required for the construction of this kth table will be proportional to the number[4] of its rows $[M^{D(G)}]$ and (for each row) the constrained number of feasible locations of the variable being eliminated (denoted w_k, where $w_k \leq M$). Thus we have a storage requirement of $2 \cdot M^{D(G)}$ elements per table (the two entries stored in each row of the table are values of f_k and y_k^*), and a computation requirement of up to $M^{D(G)+1}$ lookups (with one or more additions and comparisons per lookup).

If $w_k = w$ for all k, and all variables have dimensionality $D(G)$, the complete embedding procedure then has a computation requirement proportional to

$$[p - D(G)](w)[M^{D(G)}] \quad \text{lookups} \tag{8}$$

transient (fast store) storage requirement proportional to

$$2 \cdot [M^{D(G)}] \quad \text{entries} \tag{9}$$

and a static (secondary store) storage requirement (for the "backward" selection of the y_i after obtaining the minimum global cost) proportional to

$$[p - D(G)][M^{D(G)}] \quad \text{entries.} \tag{10}$$

For the earlier example where $M = 50 \times 30 = 1500$ locations, (for $1 < k \leq P$)$w_k = w = 10$ locations, and $p = 6$ components, if these components are connected in a linear sequence where each interior component has only one spring attached to each of its two immediate neighbors and the end components only one spring each, the number of computations[5] would involve 75×10^3 lookups and evaluations, or 75 s of computing time at 10^{-3} s per computation. This is certainly a much more reasonable requirement than the two days of computing time for a direct evaluation. We pay for this speedup[6] by having a fast storage requirement of 3×10^3 entries (or words), and backup storage requirement of 7.5×10^3 entries, versus a storage requirement of only a few entries for the direct evaluation.

Now, however, note that, if we permit just one additional spring per component (or even a single spring linking the first and last components), $D(G) = 2$ and the number of computations increase by a factor of almost $M = 1500$ to 9×10^7 lookups and evaluations, or 9×10^4 s $= 25$ h. The fast storage requirement increases by a factor of $M = 1500$ to 4.5×10^6 entries, and the backup storage requirement increases to 9×10^6 entries.

This exercise demonstrates that, for even a small increase over unit dimensionality, the utility of dynamic programming as a computational technique is questionable for the embedding task. The next subsection introduces a heuristic modification to the DP type of sequential optimization which eliminates the growth of dimensionality as more global constraints (springs) are permitted.

Linear Embedding

The sequential embedding technique which we present in this section will be called the linear embedding algorithm (LEA). The essential property of this algorithm is its ability to locate a suitable embedding with a linear, rather than exponential, growth of storage and computing time requirements as a function of the number of components in the reference. The algorithm is formally described as follows.

Given a reference with p components, for $1 \leq i \leq p$, $l_i(x_i)$ is the externally supplied local evaluation array for the ith component as a function of x_i, its embedded location. Given that the ith component is embedded in location x_i, $w_i(x_i)$ is the constrained set of feasible locations for x_{i-1}.

If y is the sequence (1, 3, 2, 5), then $y * 8$ will be another way of writing (1, 3, 2, 5, 8). $S_i(x_1, \cdots, x_i) = \sum_{j=1}^{i-1} g_{ij}(x_i, x_j)$, where each g_{ij} is an externally supplied spring array, specified as a function of the relative embeddings of the ith and jth components. This decomposition of S_i into a sum of two-place functions (springs) is not required in the following LEA. Thus, if desired, we could extend the scope of the embedding metric to include more complex relational forms without increasing the computational complexity of the embedding algorithm (LEA).

The LEA is the computation, in order, of the following sequence of $2p+2$ equations. (Note that $h_i = s_i + l_i$.)

$$g_1(x_1) = l_1(x_1) \tag{11}$$

$$y_1(x_1) = x_1 \tag{12}$$

$$g_2(x_2) = \min_{x_1 \in w_2(x_2)} [S_2(x_1, x_2) + l_2(x_2) + g_1(x_1)] \tag{13}$$

$$y_2(x_2) = y_1(x_1) * x_2, \qquad \text{where } x_1 \text{ is that value which minimizes the previous equation.} \tag{14}$$

$$\vdots$$

$$g_i(x_i) = \min_{x_{i-1} \in w_i(x_i)} [S_i(y_{i-1}(x_{i-1}) + x_i) * l_i(x_i) + g_{i-1}(x_{i-1})] \tag{15}$$

$$y_i(x_i) = y_{i-1}(x_{i-1}) * x_i, \text{ where } x_{i-1} \text{ is that value which minimizes the previous equation.} \tag{16}$$

$$\vdots$$

$$g_p(x_p) = \cdots \tag{17}$$

$$y_p(x_p) = \cdots \tag{18}$$

$$G = \min_{x_p} g_p(x_p). \tag{19}$$

$$Y = y_p(x_p), \qquad \text{where } x_p \text{ is that value which minimizes the previous equation.} \tag{20}$$

As in the subsection on dynamic programming, G [see (19)] is the total embedding cost, and the cor-

[4] Relational constraints can reduce the number of feasible rows to a value considerably below the unconstrained case. However, the computational problems, in attempting to take "advantage" of this reduction, may be prohibitive.

[5] Assuming the variables are eliminated in the order in which they appear in the linear sequence, then $D(G) = 1$.

[6] Dynamic programming can be considered to be a way of trading storage for computation time.

responding locations of the embedded components are determined from Y [see (20)].[7]

Given the restriction that the h_i in (3) are independent of all x_j for $j>i$ [this is consistent with the embedding metric as presented in (2)], then dynamic programming can be viewed as a procedure for finding the shortest path through a graph. We define this graph in the following way. The nodes of the graph are arranged in p columns, where each node in the ith column is labeled by the values of the variables corresponding to a unique set of embeddings for the first i components; there are as many nodes in the ith column as there are unique embeddings of the first i components. The length of a branch between a node in column $i-1$ (with label X_{i-1}) and a node in column i (with label X_i) is $h_i(X_i)$. We delete branches with infinite length.

To determine the shortest path from the root node (a single node placed in column zero) to some node in column p, we can proceed as follows. In the ith column, for each node, sum all the branch lengths corresponding to the label of the node. We now determine that set of variables (x_i) which do not appear in any h_j for $j>i$, and call this set of variables Z_i (the variables in Z_i correspond to components in columns $j\leq i$ which do not have spring connections to components in columns $j>i$). We now place the nodes in the ith column into sets, such that, for each set, the nodes are identical in their labels except for the variables in Z_i. For each such set, we retain the node with the shortest path length from the root node, and delete all the other nodes in the set as well as those nodes in columns $j>i$ which branch from deleted nodes. It is this pruning process which gives DP its computational advantage over complete enumeration. After the above set of operations is carried out through the Pth column, we select the node defining the shortest path through the tree, and thus the lowest cost embedding for the P components. The LEA differs from DP in that in the sequential determination of the shortest path, a maximum of only m_i nodes will be retained in the ith column (m_i is the number of permissible locations for embedding the ith component). In processing the ith column, the nodes are grouped into sets such that, for each set, the nodes are identical in their labels for x_i. For each such set we then proceed as in the DP case. Thus, at the ith iteration we save only the m_i "best" current embeddings, such that every possible positioning of the ith component occurs in one of the embeddings. This approximation technique may fail to find the best embedding (shortest path) if the components with low indicies (small i) incur a high embedding cost when placed in their optimal locations.

If the components of the reference are linked by a single chain of springs (which are then called primary springs), DP and the LEA provide identical solutions. When additional springs are present, the LEA no longer assures the optimal embedding, and these additional springs are called hueristic springs (hueristic in the sense that while these additional springs provide more information and thus give an intuitively better match than in the single chain case, the best possible use of this additional information is not always assured). The operation of the LEA is illustrated by the example given in Fig. 1.

Additional Computation Speedup and Storage Reduction Techniques

By slowing the growth of the computation and storage requirements to a linear function of the size (M) of the pictures, the LEA establishes itself as a feasible embedding procedure. However, because of the large proportionality constants, the practicality of employing the LEA will probably depend on the efficiency of the associated computer programing, and on the employment of additional (second-order) speedup and storage reduction techniques. Two of the more important speedup techniques, applicable to both DP and the LEA, are discussed below.

The dynamic programming and LEA formalisms presented earlier do not explicitly consider constraints on the variables (x_i). The simplest way of treating such constraints is to introduce into the objective function (embedding metric) cost terms which become infinite if the relational constraints are violated. This approach handles the problem by increasing the computation time needed to evaluate the additional cost terms. A much more desirable technique is to employ the explicit relational constraints (the w_i) to limit both the table size and search requirements by ignoring or eliminating infeasible variable combinations. This technique is illustrated in [1]–[3], and was considered in deriving expression (8). When the relational constraints are implicit (i.e., must be computed from the given data), it is not clear whether any advantage can be gained from first converting them to explicit form, and then applying the above technique.

Dynamic programming and the linear embedding algorithm previously described can both be speeded up by the following (type of branch-and-bound) technique. We examine a number of complete embeddings, and select the lowest cost corresponding to any one of these trial embeddings. (The embeddings themselves could have been obtained by random placements of the com-

[7] The y_i defined in (16) clarify the presentation of the LEA. However, in a computer implementation one need not calculate these y_i, which are arrays, each element of which is a sequence of locations. Rather, it is only necessary to compute a more restricted function z_i defined by

$$z_i(x_i) = x_{i-1}$$

where x_{i-1} is that value which minimizes the previous equation

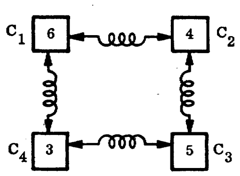
These z_i are arrays, each element of which is a single location rather than a sequence of locations. Then in the computations involving the S_i, if S_i depends on more than the two rightmost locations of $y_{i-1}(x_{i-1})*x_i$, the additional locations may be retrieved from the z_k, $1\leq k\leq i-1$.

SM

y				
4	5	2	8	8
3	7	5	1	3
2	8	1	5	7
1	4	3	2	4
	1	2	3	4 → z

x=(z,y)	SM (z,y)
1,4	5
2,4	2
3,4	8
4,4	8
1,3	7
2,3	5
3,3	1
4,3	3
1,2	8
2,2	1
3,2	5
4,2	7
1,1	4
2,1	3
3,1	2
4,1	4

(a)



$C_1(1,1) = C_1 = 6$

$C_2(1,1) = C_2 = 4$

$C_3(1,1) = C_3 = 5$

$C_4(1,1) = C_4 = 3$

$\ell_i(z,y) = |SM(z,y) - C_i|$
for $1 \le i \le 4$

Spring definition when (i,j) = (2,1) or (i,j) = (4,3)

$x_i - x_j = (z_i - z_j, y_i - y_j)$	$g_{ij}(x_i - x_j)$
1,0	0
2,0	1
otherwise	∞

Spring definition when (i,j) = (4,1) or (i,j) = (3,2)

$x_i - x_j = (z_i - z_j, y_i - y_j)$	$g_{ij}(x_i - x_j)$
0,1	0
0,2	1
otherwise	∞

(b)

Evaluation of g_2

x_2	x_1	g_1		S_2	
$z_2\ y_2$	$z_1\ y_1$	ℓ_1	ℓ_2	g_{21}	g_2
2 4	1 4	1	2	0	3
3 4	1 4	1	4	1	6
	2 4	4	4	0	
4 4	2 4	4	4	1	
	3 4	2	4	0	6
2 3	1 3	1	1	0	2
3 3	1 3	1	3	1	
	2 3	1	3	0	4
4 3	2 3	1	1	1	3
	3 3	5	1	0	
2 2	1 2	2	3	0	5
3 2	1 2	2	1	1	4
	2 2	5	1	0	
4 2	2 2	5	3	1	
	3 2	1	3	0	4

Evaluation of g_3

x_3	x_2	x_1		S_3		
$z_3\ y_3$	$z_2\ y_2$	$z_1 y_1$	ℓ_3	g_{32}	g_2	g_3
2 3	2 4	1 4	0	0	3	3
3 3	3 4	1 4	4	0	6	10
4 3	4 4	3 4	2	0	6	8
2 2	2 3	1 3	4	0	2	6
	2 4		4	1	3	
3 2	3 3	2 3	0	0	4	4
	3 4		0	1	6	
4 2	4 3	2 3	2	0	3	5
	4 4		2	1	6	
2 1	2 2		2	0	5	
	2 3	1 3	2	1	2	5
3 1	3 2	1 2	3	0	4	7
	3 3		3	1	4	
4 1	4 2	3 2	1	0	4	5
	4 3		1	1	3	

Evaluation of g_4 = G

x_4	x_3	x1						g_4	
$z_4\ y_4$	$z_3\ y_3$	$z_1\ y_1$	g_{43}	g_{41}	S_4	ℓ_4	g_3	G	
1 3	2 3	1 4	0	0	0	4	3	7	
	3 3	1 4	1	0	1	4	10	15	
2 3	3 3	1 4	0			2			
	4 3	3 4	1			2			
3 3	4 3	3 4	0	0	0	2	8	10	
1 2	2 2	1 3	0	0	0	5	6	11	
	3 2	2 3	1			5			
2 2	3 2	2 3	0	0	0	2	4	6	← [(2,3),(3,3),(3,2),(2,2)]
	4 2	2 3	1	0	1	2	5	8	
3 2	4 2	2 3	0			2			
1 1	2 1	1 3	0	1	1	1	5	7	
	3 1	1 2	1	0	1	1	7	9	
2 1	3 1	1 2	0			0			
	4 1	3 2	1			0			
3 1	4 1	3 2	0	0	0	1	5	6	← [(3,2),(4,2),(4,1)(3,1)]

(c)

Fig. 1. An example illustrating the operation of the linear embedding algorithm. The definitions of x, g_{ij}, l_i are given on pages 69 and 71. z and y are the components of x; that is, $x = (z, y)$. (a) The sensed image. (b) The reference description. (c) Linear embedding algorithm.

ponents, or perhaps by someone guessing at what a "good" embedding might actually look like.) Now, employing the LEA (or DP) to place and evaluate the cumulative cost of placement of the components sequentially, we can eliminate from further consideration any of those placements whose costs exceed the bound established by our best trial embedding. It should be noted that this technique is valid only if the cost associated with the embedding of each component is nonnegative. However, this can almost always be the case for the class of problems we are discussing in this paper.

A heuristic embellishment of the above branch-and-bound technique would be to use some fraction (say $[k/N]^{\frac{1}{2}}$) of the bound at the kth stage of an N-stage process as the threshold for eliminating a possible sequence of embeddings.

Scale and Rotation (S&R) Considerations

In attempting to match or register two images, we frequently are faced with the problem of unknown relative scale and orientation. While such variations are conceptually indistinct from any of a host of unwanted variations between the reference and the image, they (S&R) can serve as a vehicle for clarifying some important issues pertaining to the way the LEA is employed.

As noted in earlier sections, the embedding process is carried out at two levels. First, the components of the reference are searched for as independent entities. The particular processes by which these searches are executed are not a direct issue of concern here; the important point is that, regardless of the search mechanism, the outcome of the search for any individual component is presented to the LEA by a tabulation called the local evaluation array [L(EV)A]. Each entry in the L(EV)A corresponds to a possible embedding in the image of the associated reference component, indexed by the variables used to define the embedding. The entry consists of a number related to the probability that the component is actually present at the "location" specified by the indexing variables, and each entry can also contain the values of attributes of the component as measured at the indexed location. The LEA has no knowledge of the component beyond what is presented to it in the L(EV)A for that component. The purpose of the LEA is to integrate global or structural knowledge with the information provided in the L(EV)A's to find the best overall embedding (or embeddings) of the reference in the image. The acceptability of the final embedding selected by the LEA will thus be dependent on the quality of the information presented in the L(EV)A's, where the extent of this dependence is related to the relative importance of local (component definition) versus global (intercomponent) information for the particular problem. Thus, it is the responsibility of the local evaluation function, in attempting to gather evidence about the presence of some given reference component, to be able to deal with the various noise and distortion processes (such as S&R) which might be encountered. To the extent that these same noise and distortion processes affect the global or structural relationships between the reference components, the LEA provides the machinery necessary to deal with the resulting problems.

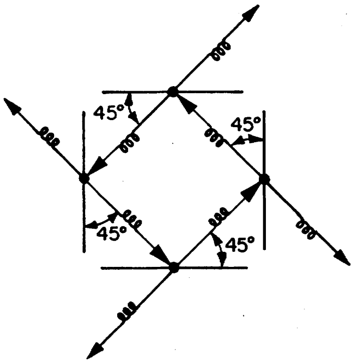


Fig. 2. Reference description of a square.

Ability to deal with variations at the global level is accomplished by defining "attributes" which measure (or estimate) these variations, and then making the "spring" parameters functions of these attributes.

Thus, in the case of S&R, if a component P_1 has scale and rotation attributes S_1 and θ_1, the springs (vectors) attached to P_1 would be (conceptually) scaled as a function of S_1, and rotated as a function of θ_1. The following example illustrates some of the above comments.

Problem

Given a two-dimensional region in which there are k randomly oriented and positioned line segments, find the four-line segments which best approximate a square. Each line is specified by a four-tuple of the type (x, y, θ, l) where the x, y coordinates locate the center of the segment, θ specifies the orientation, and l the length of the segment. To simplify this example, we will ignore the detection problem and assume that the given values for each segment are known with probability one. We assign a cost C_1 for each unit of positional disparity between the sides of a candidate square, and a cost C_2 for each degree of rotational disparity between the sides (i.e., sides should meet at right angles).

Solution

Consider a single "local evaluation array" consisting of the given list of four-tuples (x, y, θ, l). We can consider x, y to be the "location" indexing variables, and θ, l to be attributes. All entries have unit probability, entries with zero probability are deleted. The "description" of a square is shown schematically in Fig. 2. Each (spring) vector is rigidly attached to its line segment at a fixed angle of 45°. Costs (C_1) associated with (x, y) disparity between the end of a vector and the center of the next line segment are circularly symmetric (i.e., the "cost" function increases with distance from the tip of the vector) about the tip of the vector. We also assess a cost (C_2) proportional to the difference in measured (attribute value) orientation of more or less than 90° for sequential line segments.

The general form of the LEA, with orientation adjusted springs, and spring costs augmented by attribute (S&R) differences, is adequate to deal with the problem as posed. A minor difficulty arises from the fact that the orientation of a line segment is actually two valued (i.e., θ and $\theta+180°$). If we list each line segment twice in the local evaluation array, once for each of its two orientations, then the LEA can be applied without modification.

We can handle squares of differing size by using the line-segment-length attribute in the same manner as the orientation attribute (except that double entries are not required here). Spring stretching is augmented by a cost proportional to the difference in length attribute for sequential line segments. That is, when the LEA examines a new line segment as a possible additional side for a square already partially formed, it compares the length of the new line segment with the length of the line segment in the partial square to which the new segment will be attached. The spring between the new and the old line segments then is stretched (over and above any stretching due to angular and positional disparity) by an amount proportional to the difference in these lengths.

In the above example note that, because each entry in the local evaluation array had either zero (∞ cost) or unit (0 cost) probability associated with its occurrence, the size of this array could be reduced to listing only those few coordinate combinations associated with the feasible (nonzero probability) occurrence of a component (line segment). This procedure can be used in other situations where it is reasonable to reduce all low probability entries in the local evaluation array to zero.

Pictorial Representation

A central problem in much of the work concerned with the computer processing of pictorial data is that of representation. Since we cannot manipulate the real world object (itself) within the computer, we attempt to construct a representation (or model) which can be used in place of the actual object and which has the following (somewhat overlapping) properties.

Complete: Any question of interest which could be resolved by reference to the actual object should also be capable of being resolved by reference to the representation.

Compact: The representation should be free of information redundant to the purposes for which it will be used. This is necessary to minimize computer storage requirements.

Transformable: Much of the information contained in a representation will be implicit rather than explicit in form. The ability to manipulate easily the representation to extract required information is essential. For example, if we represent a picture by an intensity matrix or raster, then a count of the number of isolated objects appearing in the picture would be implicit information which could be extracted from the representation after considerable processing. However, if the representation consisted of the contours of the object appearing in the picture, then the required count could be obtained rather simply.

Incrementally Changeable: If we observe a slight change in the real world object, it should be a relatively simple and straightforward task to alter the representation. Further, from the standpoint of image matching, a small change in the real world should require only a small change in the representation.

Accuracy and Simplicity of Translation: Given a real world object, it should be relatively simple to derive an accurate representation of the object.

Over the past ten years or so, much of the work concerned with pictorial representation has been restricted to the domain of line type drawings, and the use of formal linguistic methods (see [8]–[10]). Very little success has been achieved in attempts to extend this work to scenes of terrains, cloud covers, human faces, etc., which can only be described meaningfully in terms of picture components which are not line elements, but which are regions with colors, textures, shadings, etc.[8]

Perhaps the most serious failing of the linguistic (and similar) techniques occurs with respect to the "translation" property. These techniques build a representation by constructing a hierarchy based on picture primitives, assembled into linear expressions employing specified relational forms, and satisfying a set of syntax rules. The problem arises from the fact that (usually) the only direct correspondence between the actual object and its representation occurs at the level of the primitive elements (typically points, intensities, and lines), while in practice there are pieces of a picture that are too involved or complicated to describe in terms of these primitives. Theoretically, the description is possible since the matrix representing the picture is finite. However, such a description would be so complicated that, aside from the difficulty of composing it, there would be a considerable likelihood of error and inaccurate representation.

In the previous portions of this paper we have presented a representational scheme for pictures, and were primarily concerned with its application to image matching. We will now show that the representational scheme has wide general applicability, and avoids many of the problems of the linguistic approaches. First we note that it is a hybrid type of representation in that it invokes symbolic (numerical) elements as well as allowing actual picture segments to be part of the representation. The ability to intermix picture segments and symbolic data in the same representation greatly simplifies

[8] Specialized systems have been developed, highly tailored to specific problems, which are exceptions to this assertion; e.g., see Kelly [11]. A number of papers, including Bledsoe [15], [16] and Goldstein and Harmon [17], effectively consider the problem of face identification based on feature measurements obtained manually.

the translation problem. Where a pictorial concept is difficult to describe symbolically, we can use an actual piece of the picture as part of the description. A second aspect of our representation that simplifies the translation problem is the fact that the components (picture pieces, local evaluation arrays, etc.) and the relational forms (springs) are two-dimensional rather than one-dimensional entities. We thus avoid the problem of having to construct a one-dimensional model for a two-dimensional structure.

Let us now examine some of the other representational attributes. Incremental changeability follows directly from ease of translation (although the inverse relationship would not necessarily hold). Transformability with respect to image matching has certainly been established; the fact that the representation is already in two-dimensional form, with metric, geometrical, and topological relationships explicitly and quantitatively expressed, implies that transformability for many other applications is more than adequate.

In many respects, compactness and transformability are antithetical since data in explicit form are usually more extensive than the equivalent implicit information. This is the case with our representation. It requires considerably more storage than might be required for a linguistic representation.

The one area where the linguistic approach has an obvious advantage is with respect to completeness. A linguistic representation can treat nonpictorial information (e.g., relations between items in a picture and other items not visible, but perhaps implied or normally associated with the pictured items) in a way that would be extremely difficult in our representation. This additional capability could, of course, be achieved by appending the necessary linguistic machinery to our current scheme, although the final result might well be a mixture of two representations rather than one integrated representation.

Experimental Results

In order to evaluate the practical implications of the techniques presented in this paper, we have initiated a program involving experiments on a variety of line type drawings and gray-level imagery. Over 400 experiments have already been performed with the following general results.

1) On well-defined imagery (i.e., relatively noise-free and unblurred pictures), the embeddings produced by the LEA were almost always in agreement with the best embedding as predetermined by human evaluators. Where the few deviations did occur, they were reasonable, and usually related to the crude component descriptions employed.

2) On noisy imagery (course resolution, additive random and coherent noise), the fall in performance paralleled the difficulty human evaluators had in locating suitable embeddings. Where the components were discernible in the image, the embeddings were usually correct; those components which were significantly altered by the noise were sometimes missed, but the substitution error was usually in close proximity to the correct embedding location, and, even in error, the correct embedding almost always had a score close to the best score.

Since the programmed version of the algorithm is still evolving, and some of the discussed features have not yet been implemented (e.g., the "attribute" feature is not operational as yet), most of the experiments were informal in nature. However, for the purposes of this paper, two sets of controlled experiments were run and are described below.

Image-Matching Experiments Using Faces

The majority of the experiments we have run to date had human faces as their subject. Reasons for this selection include the following.

1) The availability of a set of digitized gray-scale pictures containing faces.

2) A single reference (face) could be tested on all the faces in the data set. In the case of, say, terrain pictures, a separate reference (or, at best, a unique composition of standard reference components) is necessary for each picture.

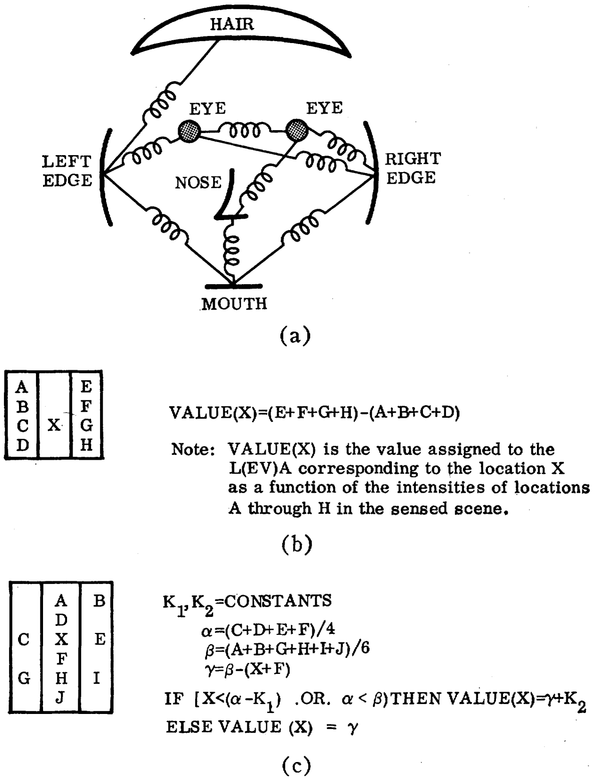
3) Our familiarity with faces and their components (eyes, nose, mouth, etc.) facilitates evaluation of performance as noise and distortion are introduced.

The data set used in the face experiments consisted of 15 human faces,[9] both men (some with beards) and women, digitized to approximately 16 true gray levels, and each face typically was contained in a picture field of from 2000 to 3000 resolution elements. Using a reference as shown schematically in Fig. 3(a), with components as described in Fig. 3(b) and (c), almost 300 formal and informal experiments were performed. In each experiment, additive (truncated) Gaussian random noise with zero-mean and standard deviation of either 0, 10, or 15 units was added to each resolution element (relative to a pseudogray scale of 64 units for the noise-free pictures). In some of the informal experiments, coherent noise consisting of randomly placed lines was also inserted [see Fig. 4(a)].

With no more than two or three exceptions, when the reference was restricted to hair, eyes, and sides of face, correct embedding was achieved. These results are gratifying in view of the simple component descriptions employed, and the equivocation displayed by the resulting L(EV)A's. (See examples shown in Fig. 4.)

Two series of formal experiments were run on the

[9] These data were obtained from the Stanford Artificial Intelligence Laboratory and are a subset of the data employed in the experiments described by Kelly [11].



VALUE(X)=(E+F+G+H)-(A+B+C+D)

Note: VALUE(X) is the value assigned to the L(EV)A corresponding to the location X as a function of the intensities of locations A through H in the sensed scene.

K_1, K_2=CONSTANTS

$\alpha=(C+D+E+F)/4$

$\beta=(A+B+G+H+I+J)/6$

$\gamma=\beta-(X+F)$

IF $[X<(\alpha-K_1)$ .OR. $\alpha<\beta)$THEN VALUE(X)=$\gamma+K_2$

ELSE VALUE (X) = γ

Fig. 3. Reference description of a face. (a) Schematic representation of face reference, indicating components and their linkages. (b) Reference description for left edge of face. (c) Reference description for eye.

(noisy) face pictures using two references which included, but differed in, the nose/mouth definitions. In the first series, consisting of 90 experiments, there were 83 completely correct embeddings, and 7 partially incorrect embeddings. The errors involved six experiments in which the nose/mouth complex was offset by three to four resolution cells from its ideal location, and one experiment in which both the eyes and the nose/mouth complex were improperly placed. In the second series, consisting of 45 experiments, the placement of the nose/mouth complex was judged incorrect in 3 experiments, while all the other components were always correctly embedded.

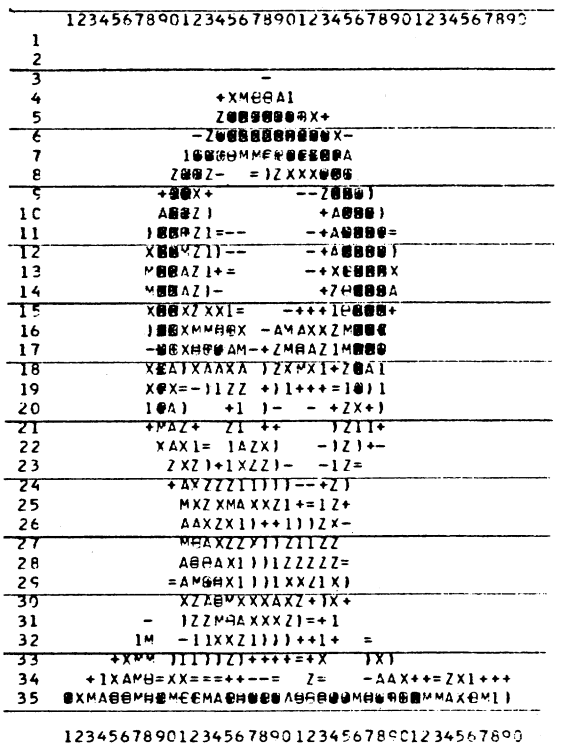
Analysis of the face experiments led to the following conclusions. In spite of almost perfect performance in embedding the hair, eyes, and sides of the face, precise placement of the nose/mouth complex based on strictly local evaluation was almost impossible in some of the noisy pictures due to loss of detail [e.g., see Fig. 4(b)]. With the attribute feature of the LEA not yet operational, and with the arbitrary decision to use binary (rather than multivalued) weights in the spring arrays for these experiments, the LEA restricted the feasible region over which an optimum value could be selected for embedding the nose/mouth complex, but did not bias the selection as would generally be the case. In the presence of heavy noise, the simple nose/mouth descriptions used in these experiments were not always adequate to produce a local optimum in the L(EV)A at or near the ideal embedding location. (A three-resolution cell deviation was considered an error.)

Image-Matching Experiments Using Terrain Scenes

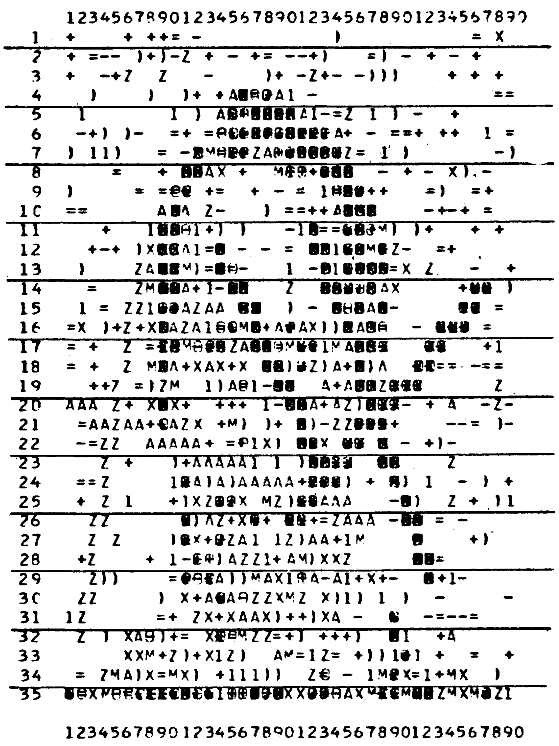
Approximately 40 experiments have been performed using terrain scenes (including both aerial and ground scenes). The object in each case was to create a relatively simple description of some portion of the scene and then attempt to find the proper embedding of the description in the image (or some distorted or alternate view of the image).

The descriptions employed two basic types of components: 1) *texture components*, in which the "texture value" of a point was defined as a crude statistical function of the intensity values and gradients in some local region surrounding the point; and 2) *shape components*, which were defined by collections of "edge" points having specified gradients.

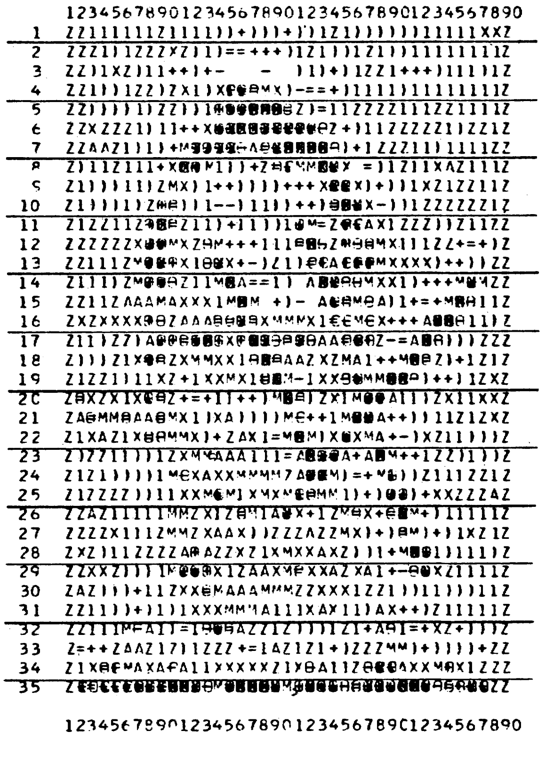
Fig. 5(a) shows an example of a terrain (reference) description. Fig. 5(b) shows its successful embedding relative to the computer-stored version of the photograph of the actual terrain segment as shown in Fig. 5(c). Each coherent piece in reference 5(a) is represented by several points enclosed by a dotted line. In this example, the points of each enclosure of the reference com-



Original picture.



Noisy picture (sensed scene) as used in experiment.

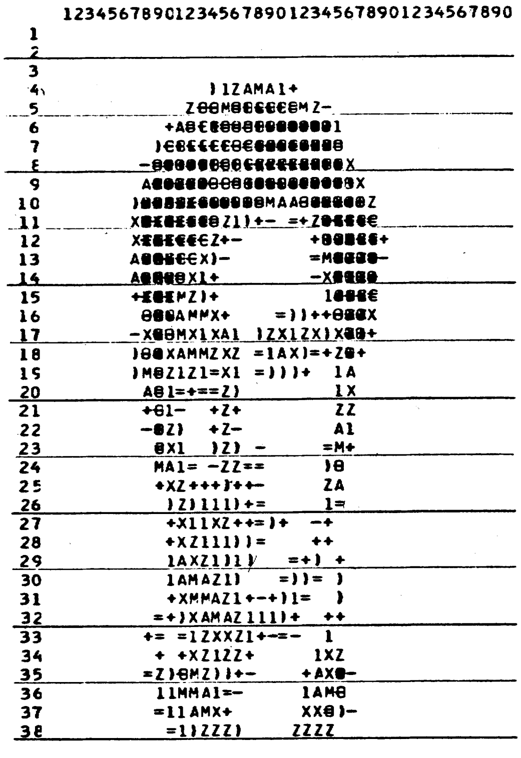


L(EV)A for eye. (Density at a point is proportional to probability that an eye is present at that location.)

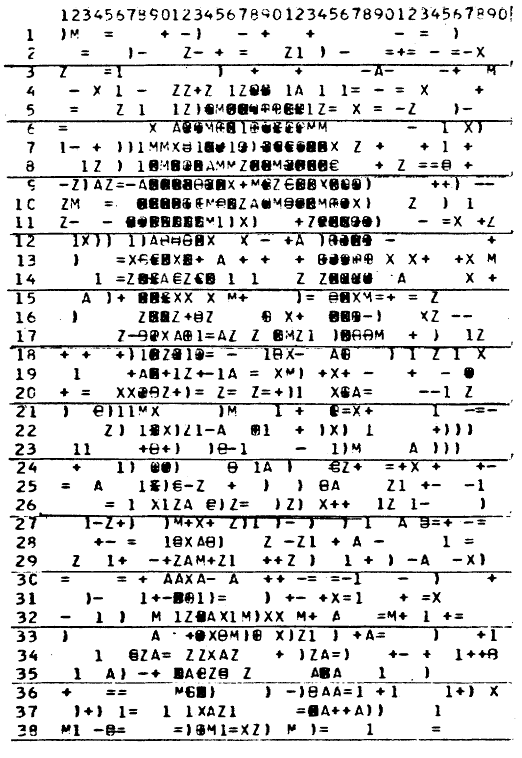
```
HAIR WAS LOCATED AT (6, 18)
L/EDGE WAS LOCATED AT (18, 10)
R/EDGE WAS LOCATED AT (18, 25)
L/EYE WAS LOCATED AT (17, 13)
R/EYE WAS LOCATED AT (17, 21)
NOSE WAS LOCATED AT (22, 18)
MOUTH WAS LOCATED AT (24, 17)
```

(a)

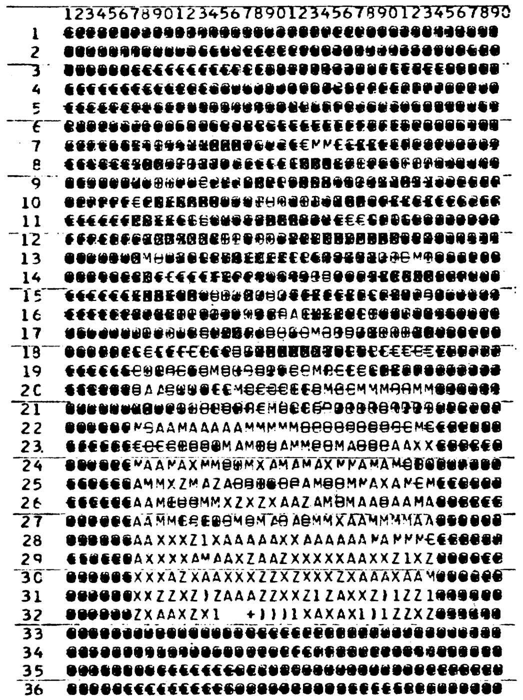
Fig. 4. Examples of image-matching experiments using faces. (a) Successful embedding under coherent noise.



Original picture.



Noisy picture (sensed scene) as used in experiment.

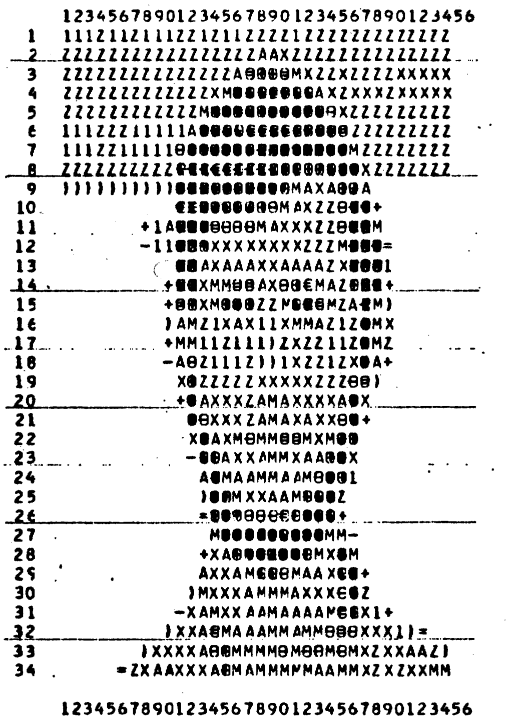


L(EV)A for nose. (Density at a point is proportional to probability that nose is present at that location.)

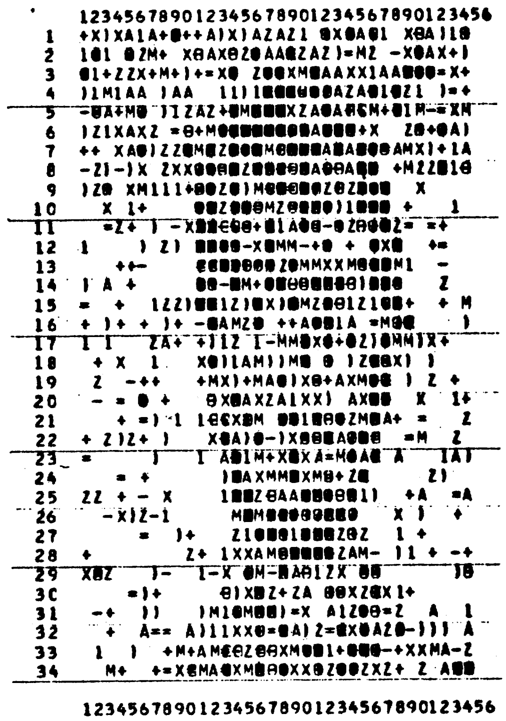
```
HAIR WAS LOCATED AT (8,21)
L/EDGE WAS LOCATED AT (17,11)
R/EDGE WAS LOCATED AT (17,25)
L/EYE WAS LOCATED AT (17,14)
R/EYE WAS LOCATED AT (17,20)
NOSE WAS LOCATED AT (21,16)
MOUTH WAS LOCATED AT (23,16)
```

(b)

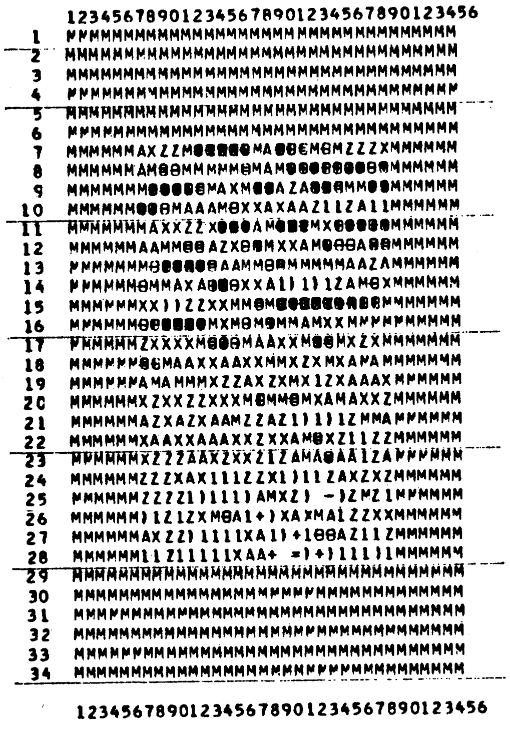
Fig. 4 (continued). (b) Incorrect embedding of nose under random noise.



Original picture.



Noisy picture (sensed scene) as used in experiment.

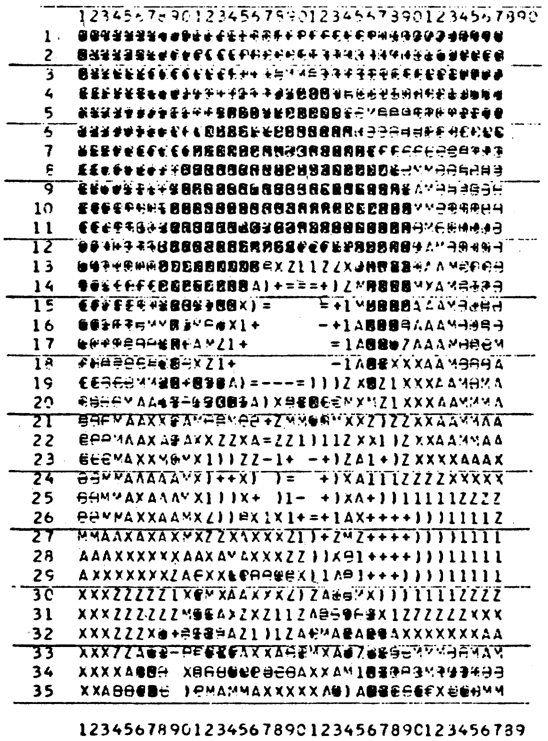


L(EV)A for nose. (Density at a point is proportional to probability that nose is present at that location.)

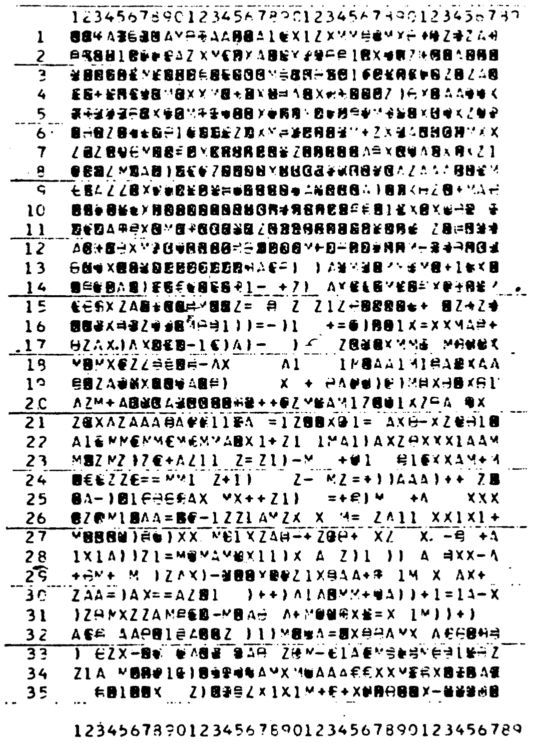
```
HAIR WAS LOCATED AT (7, 23)
L/EDGE WAS LOCATED AT (17, 13)
R/EDGE WAS LOCATED AT (17, 26)
L/EYE WAS LOCATED AT (14, 17)
R/EYE WAS LOCATED AT (14, 23)
NOSE WAS LOCATED AT (20, 20)
MOUTH WAS LOCATED AT (22, 20)
```

(c)

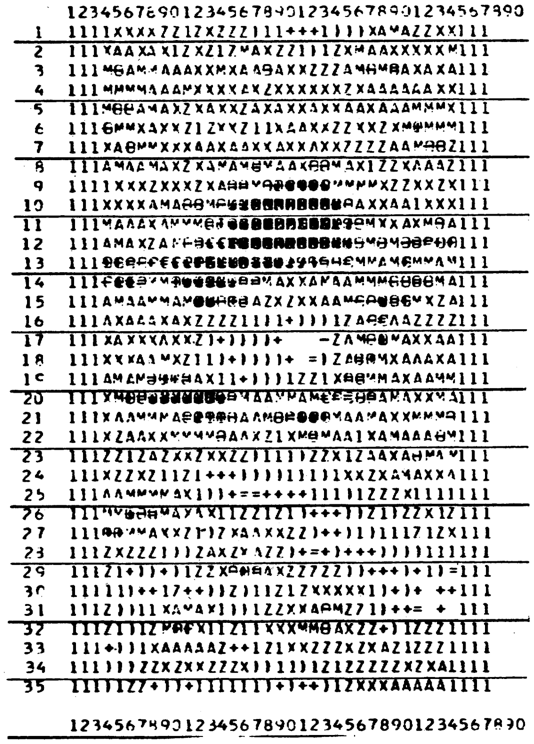
Fig. 4 (continued). (c) Successful embedding under random noise.



Original picture.



Noisy picture (sensed scene) as used in experiment.

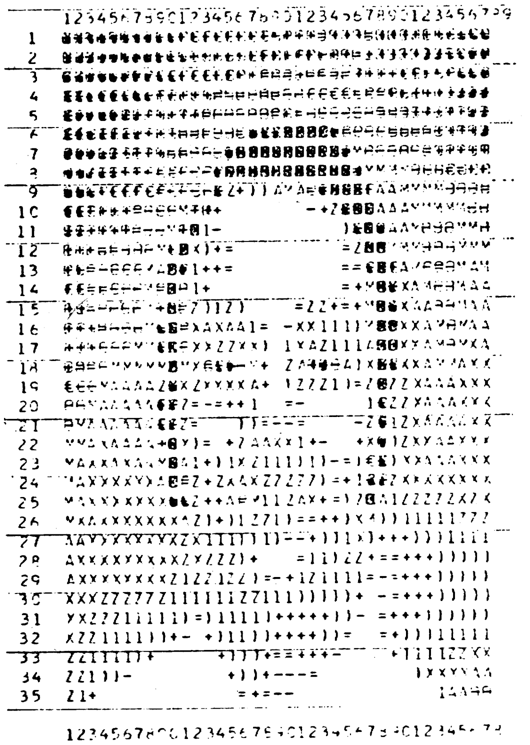


L(EV)A for hair. (Density at a point is proportional to probability that hair is present at that location.)

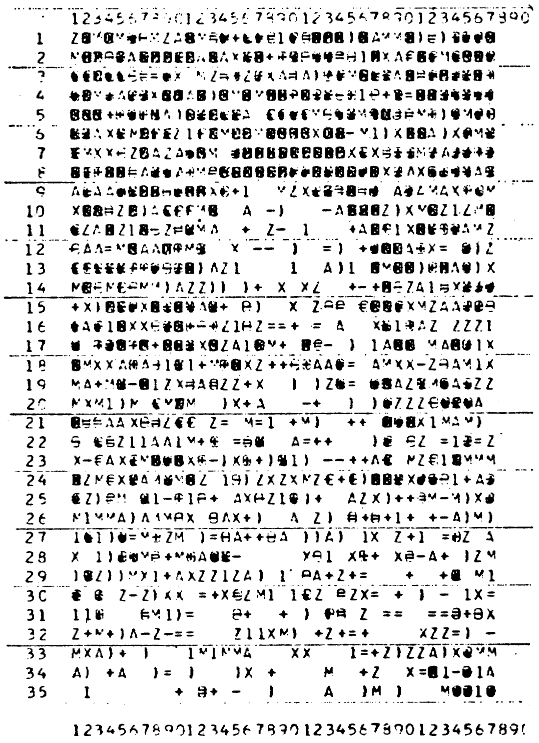
```
HAIR WAS LOCATED AT (11, 21)
L/EDGE WAS LOCATED AT (25, 11)
R/EDGE WAS LOCATED AT (25, 24)
L/EYE WAS LOCATED AT (21, 15)
R/EYE WAS LOCATED AT (21, 21)
NOSE WAS LOCATED AT (26, 18)
MOUTH WAS LOCATED AT (29, 17)
```

(d)

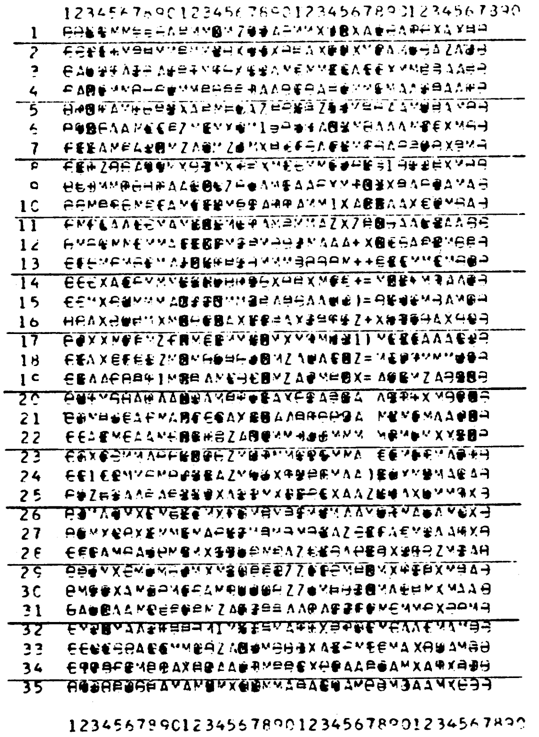
Fig. 4 (continued). (d) Successful embedding under random noise.



Original picture.



Noisy picture (sensed scene) as used in experiment.

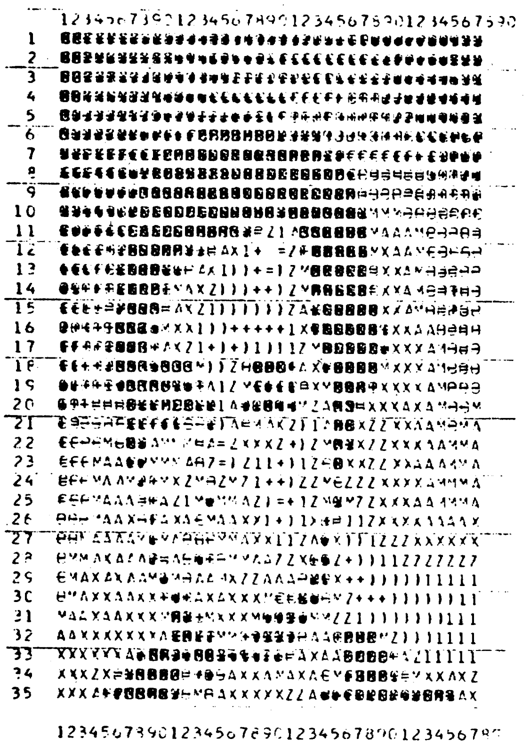


L(EV)A for L/EDGE. (Density at a point is proportional to probability that L/EDGE is present at that location.)

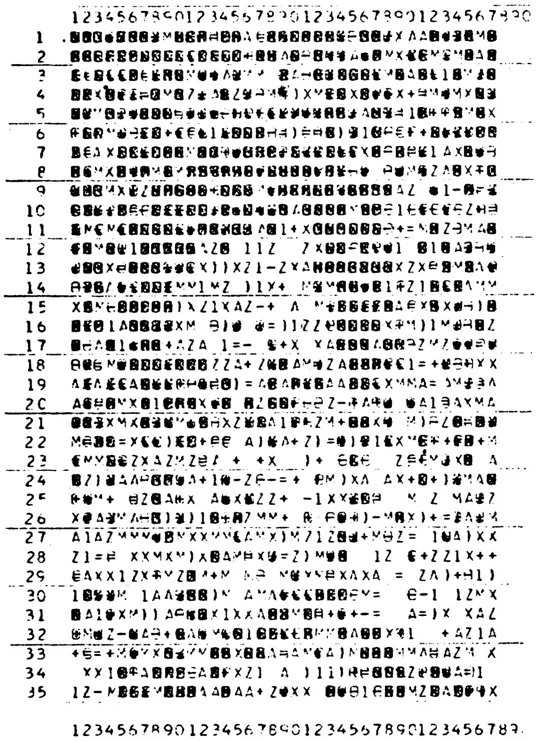
```
HAIR WAS LOCATED AT (8, 20)
L/EDGE WAS LOCATED AT (20, 11)
R/EDGE WAS LOCATED AT (20, 27)
L/EYE WAS LOCATED AT (18, 15)
R/EYE WAS LOCATED AT (18, 23)
NOSE WAS LOCATED AT (23, 18)
MOUTH WAS LOCATED AT (25, 18)
```

(e)

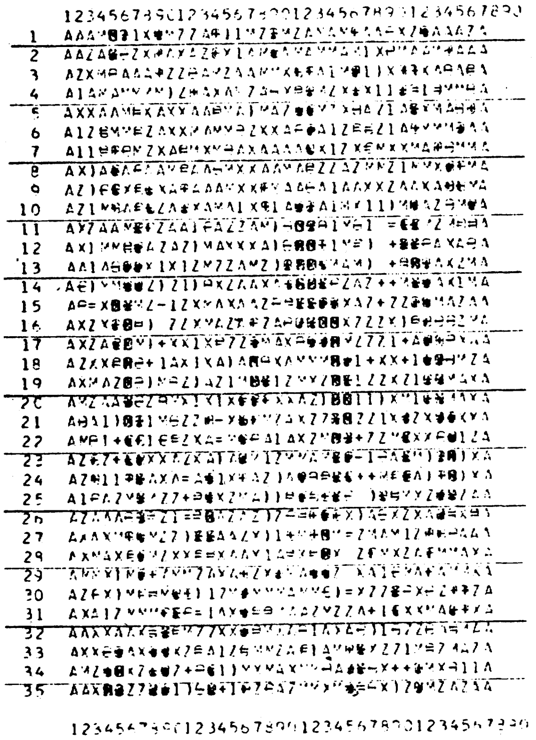
Fig. 4 (continued). (e) Successful embedding under random noise.



Original picture.



Noisy picture (sensed scene) as used in experiment.

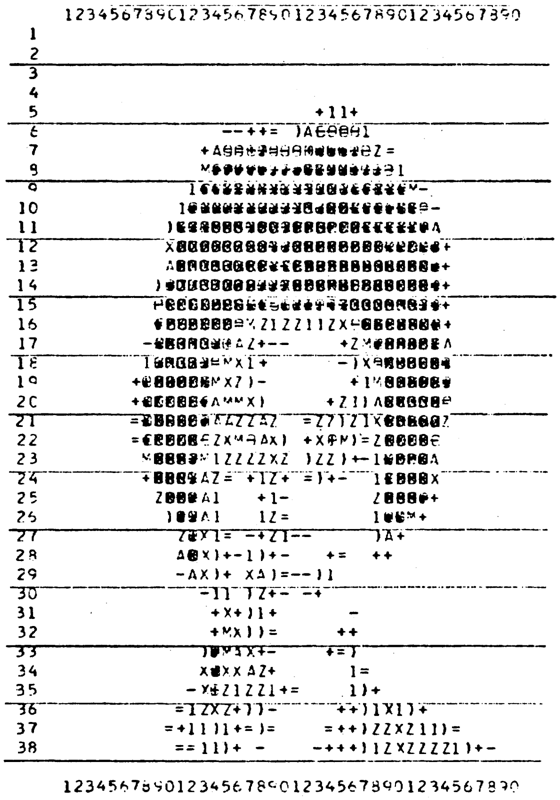


L(EV)A for R/EDGE. (Density at a point is proportional to probability that R/EDGE is present at that location.)

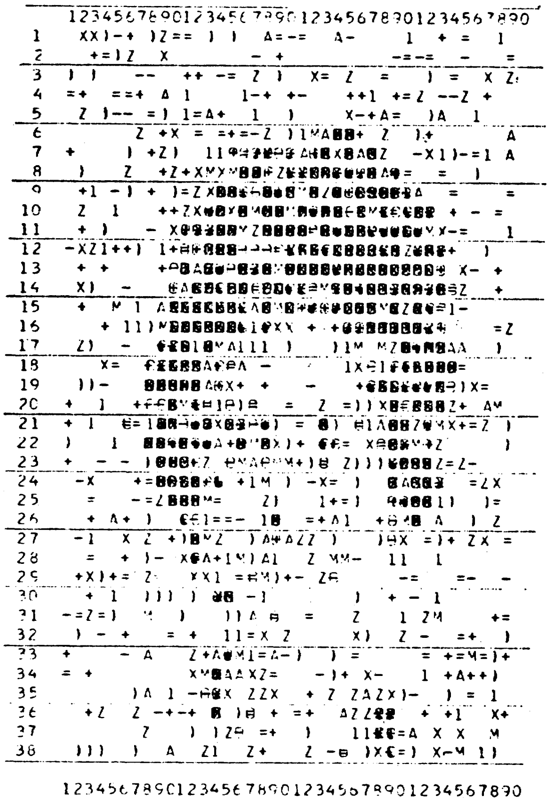
```
HAIR WAS LOCATED AT (11,15)
L/EDGE WAS LOCATED AT (19,8)
R/EDGE WAS LOCATED AT (19,24)
L/EYE WAS LOCATED AT (19,12)
R/EYE WAS LOCATED AT (19,20)
NOSE WAS LOCATED AT (25,15)
MOUTH WAS LOCATED AT (28,15)
```

(f)

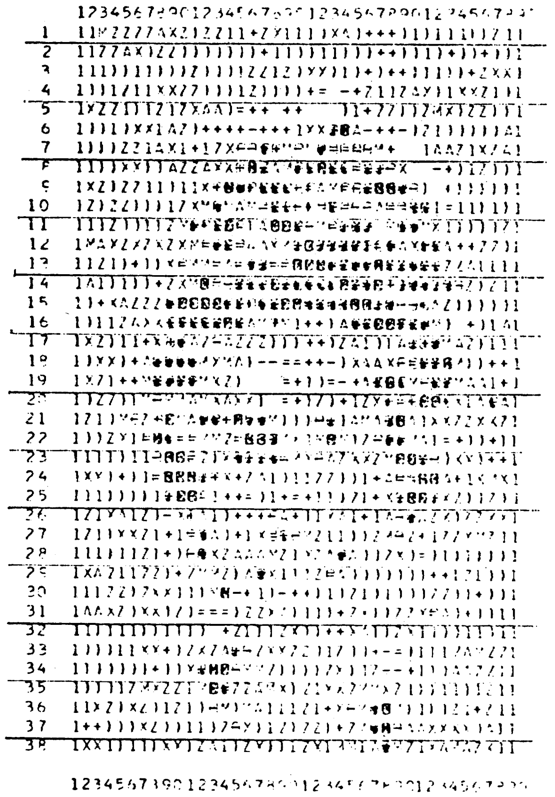
Fig. 4 (continued). (f) Successful embedding under random noise.



Original picture.



Noisy picture (sensed scene) as used in experiment.

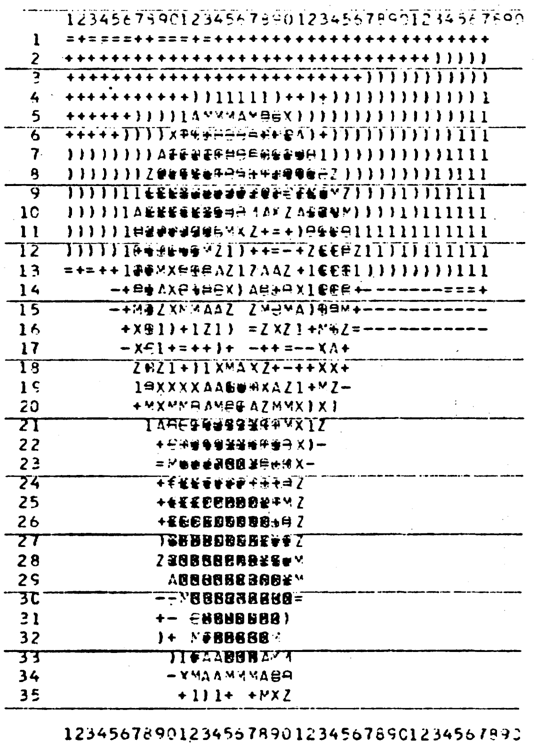


L(EV)A for eye. (Density at a point is proportional to probability that eye is present at that location.)

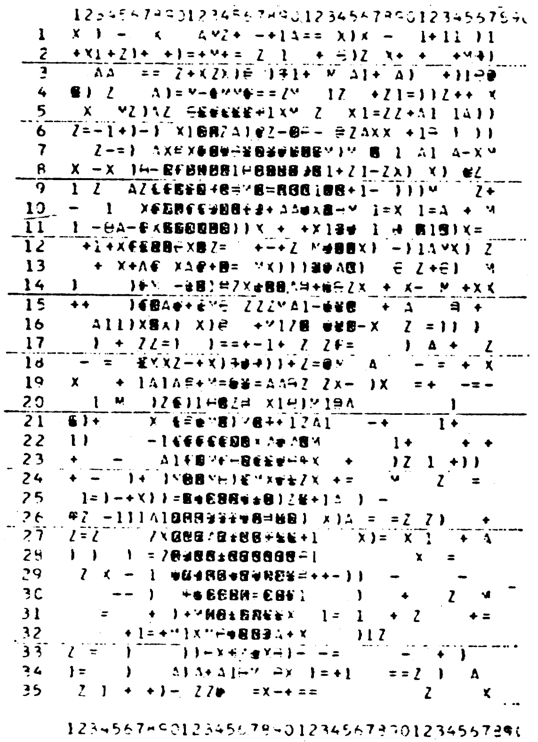
HAIR WAS LOCATED AT (13, 23)
L/EDGE WAS LOCATED AT (25, 13)
R/EDGE WAS LOCATED AT (25, 28)
L/EYE WAS LOCATED AT (22, 16)
R/EYE WAS LOCATED AT (22, 23)
NOSE WAS LOCATED AT (27, 20)
MOUTH WAS LOCATED AT (29, 19)

(g)

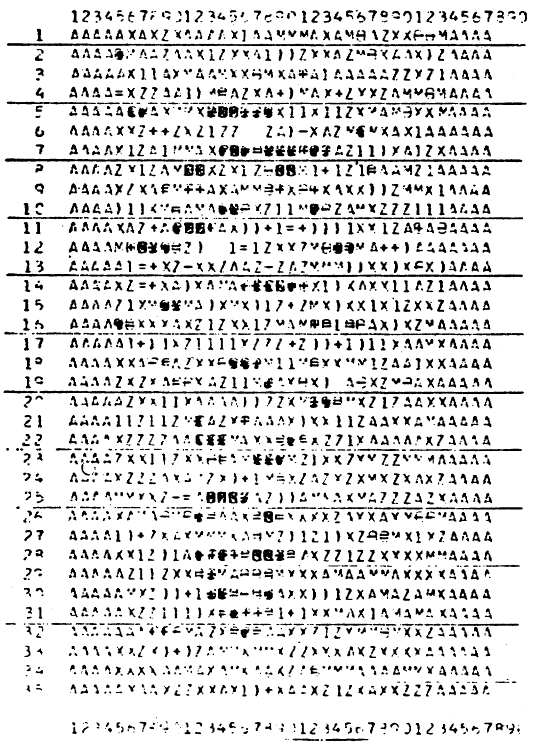
Fig. 4 (continued). (g) Successful embedding under random noise.



Original picture.



Noisy picture (sensed scene) as used in experiment.

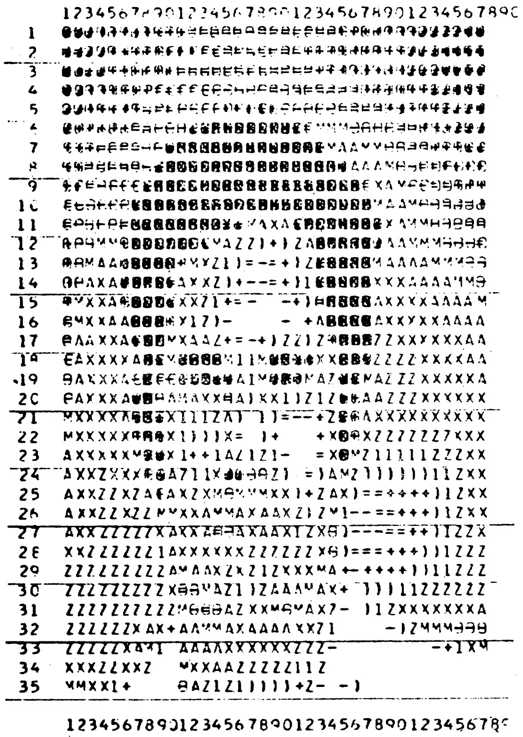


L(EV)A for mouth. (Density at a point is proportional to probability that mouth is present at that location.)

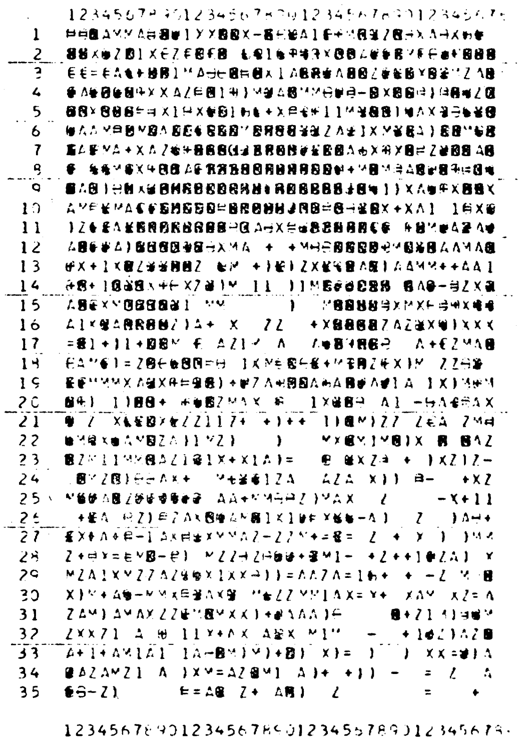
```
HAIR WAS LOCATED AT (8, 19)
L/EDGE WAS LOCATED AT (16, 9)
R/EDGE WAS LOCATED AT (16, 23)
L/EYE WAS LOCATED AT (14, 12)
R/EYE WAS LOCATED AT (14, 18)
NOSE WAS LOCATED AT (18, 16)
MOUTH WAS LOCATED AT (21, 16)
```

(h)

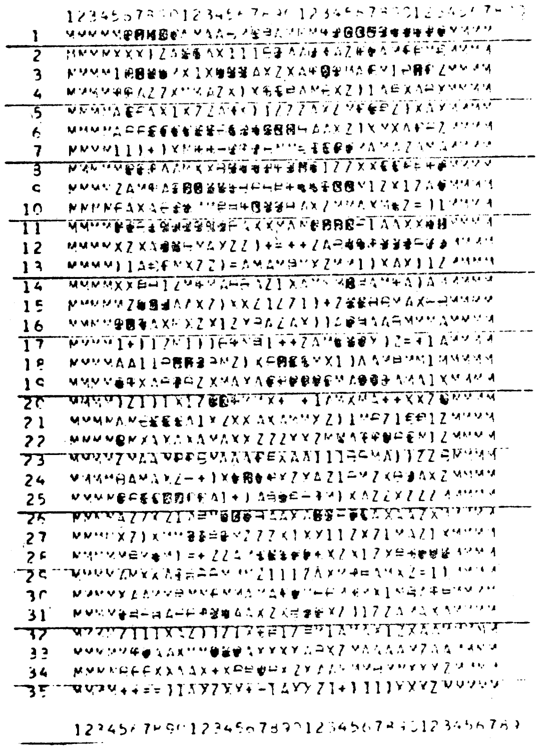
Fig. 4 (continued). (h) Successful embedding under random noise.



Original picture.



Noisy picture (sensed scene) as used in experiment.

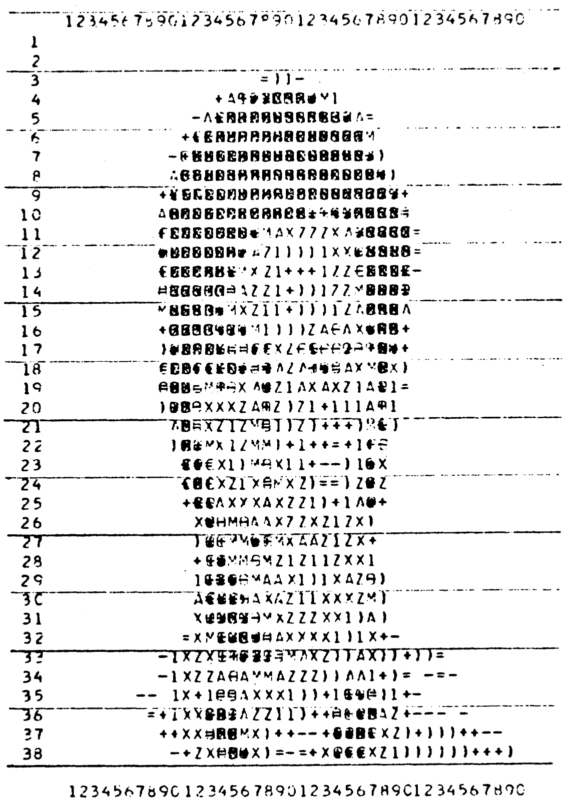


L(EV)A for nose. (Density at a point is proportional to probability that nose is present at that location.)

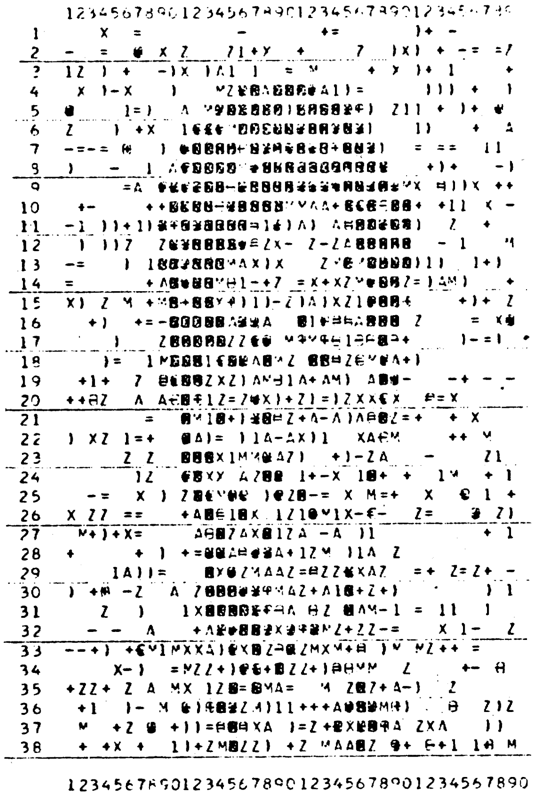
```
HAIR WAS LOCATED AT (9,19)
L/EDGE WAS LOCATED AT (22,9)
R/EDGE WAS LOCATED AT (22, 22)
L/EYE WAS LOCATED AT (19, 12)
R/EYE WAS LOCATED AT (19, 19)
NOSE WAS LOCATED AT (24, 16)
MOUTH WAS LOCATED AT (26,15)
```

(i)

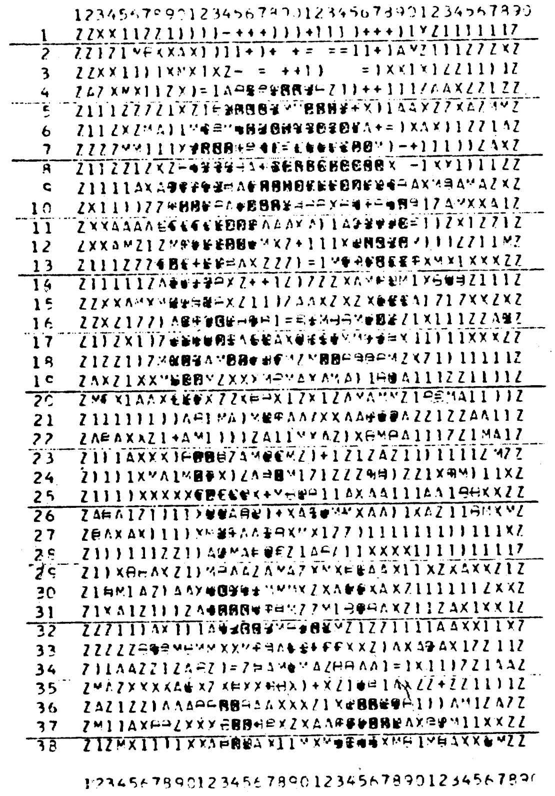
Fig. 4 (continued). (i) Successful embedding under random noise.



Original picture.



Noisy picture (sensed scene) as used in experiment.

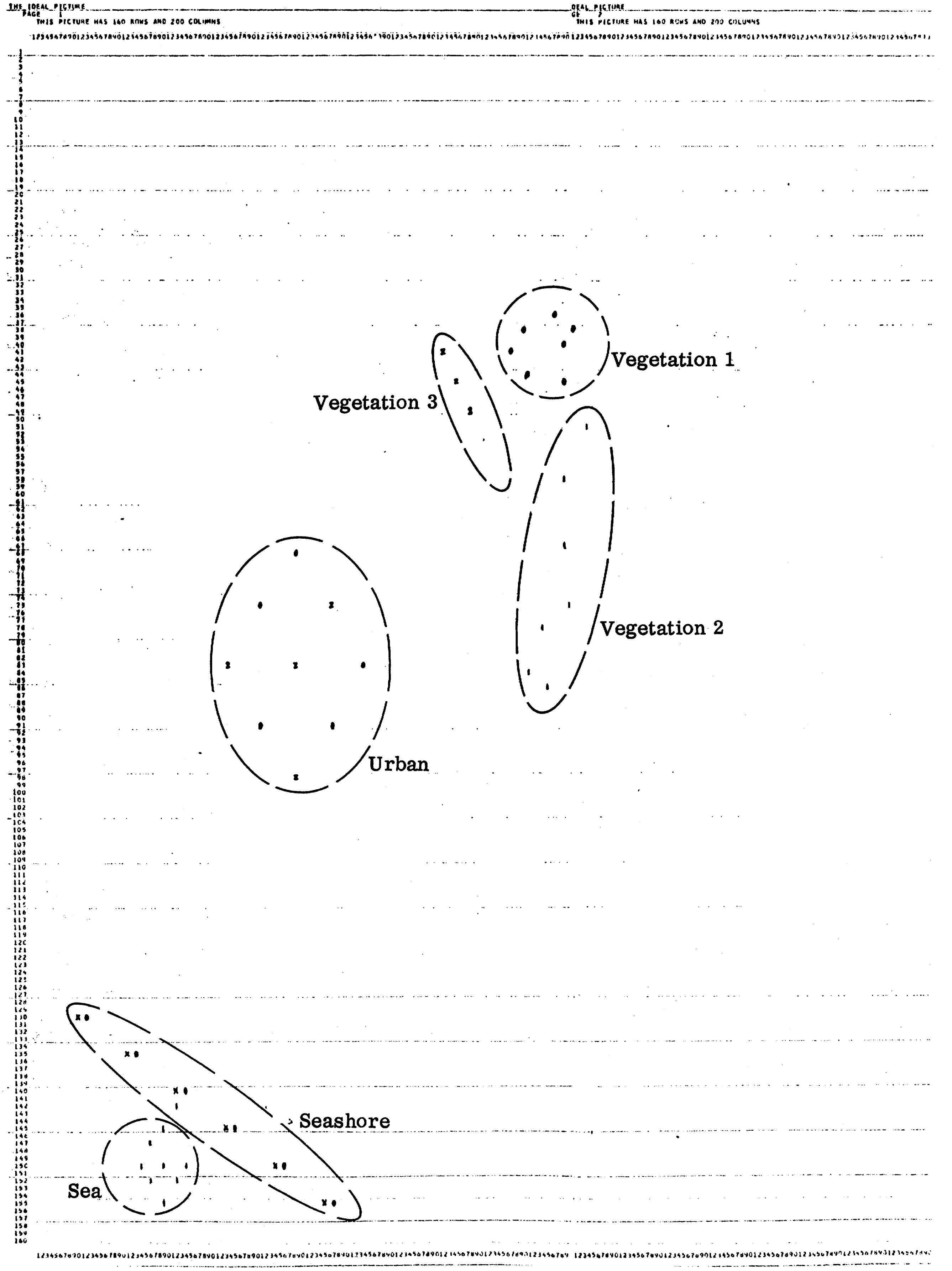


L(EV)A for eye. (Density at a point is proportional to probability that eye is present at that location.)

```
HAIR WAS LOCATED AT (9, 20)
L/EDGE WAS LOCATED AT (20,12)
R/EDGE WAS LOCATED AT (20,27)
L/EYE WAS LOCATED AT (18,15)
R/EYE WAS LOCATED AT (18, 22)
NOSE WAS LOCATED AT (24, 18)
MOUTH WAS LOCATED AT (27, 19)
```

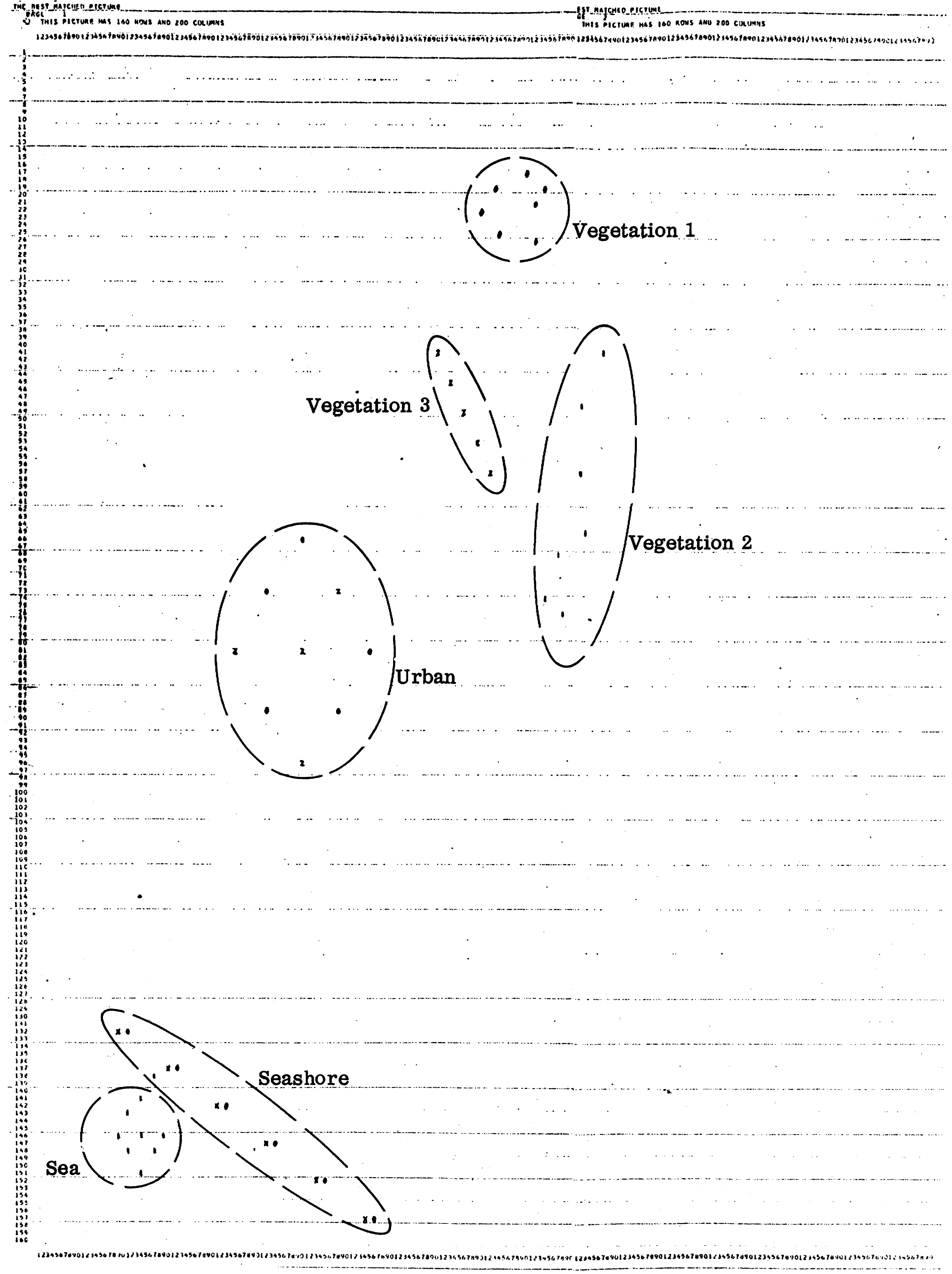
(j)

Fig. 4 (continued). (j) Successful embedding under random noise.



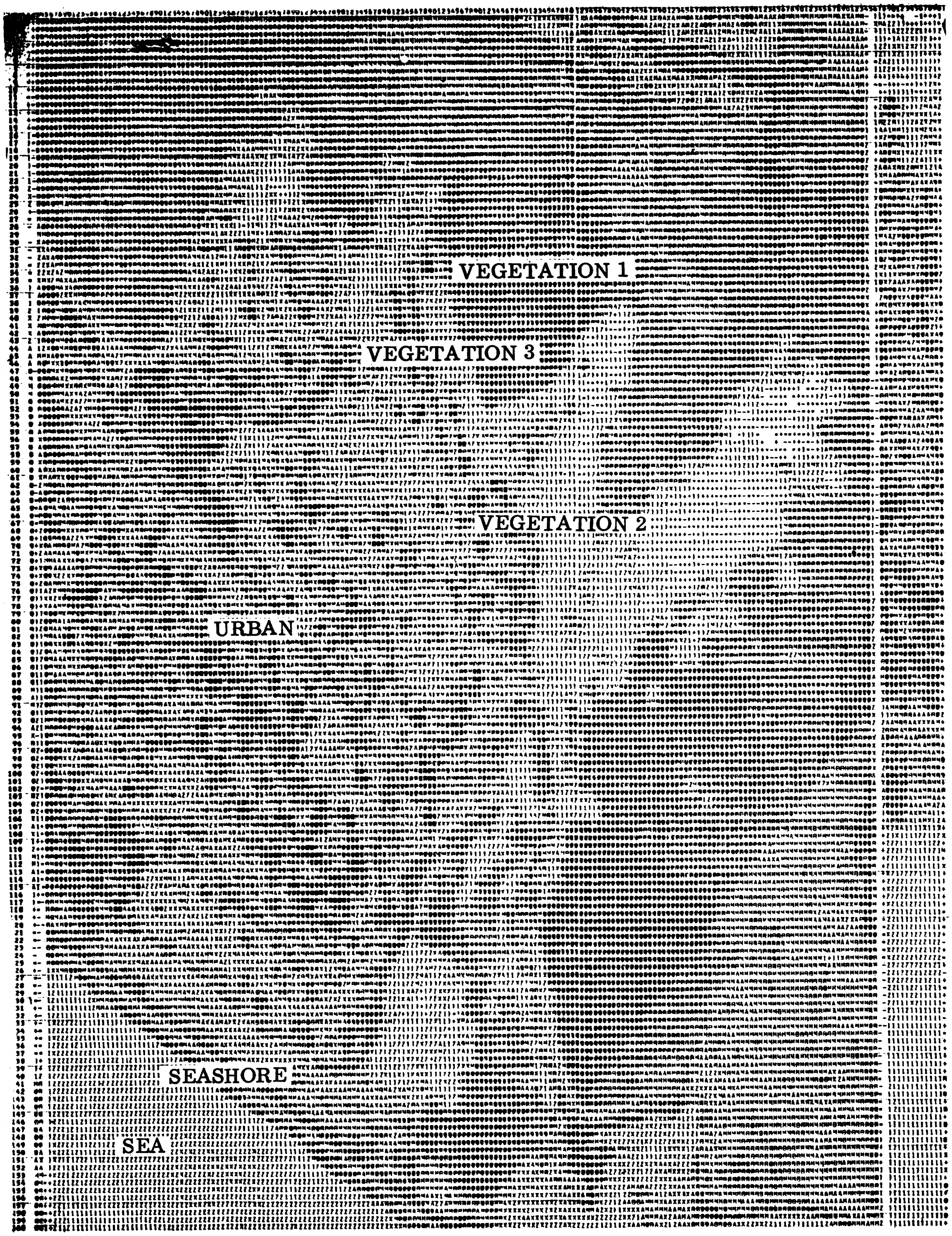
(a)

Fig. 5. Example of image-matching experiment using a terrain scene. (a) Reference for terrain.



(b)

Fig. 5 (continued). (b) Embedding of reference in sensed image for terrain.



(c)

Fig. 5 (continued). (c) Sensed image for terrain.

posed a rigid template which is correlated at the different positions of the sensed scene in order to get the local evaluation array. The correlation is performed only at the indicated points of the template. Reproduction difficulties make it hard to see the intensities of these enclosed points. For "Vegetation 1," all the points are of intensity 40, for "Vegetation 2," 20. For "Urban," the points were randomly assigned intensities of 25 and 35, and so on for the other pieces.

The terrain-matching experiments represent a continuing area of investigation. Our current efforts are primarily directed toward obtaining a satisfactory set of primitives which can be used as the basis for reference component description. In low-noise experiments, with limited geometric distortion, the simple descriptions (i.e., ad hoc shape and texture components linked by springs) produced correct embeddings. Experiments under more severe conditions remain to be performed.

Implementation Details

In all of the face and terrain experiments presented in this paper, the springs were assigned the values

$$g_{ij}(x_j - x_i) = \begin{cases} 0, & \text{if } A \leq \text{row } (x_j - x_i) \leq B \text{ and} \\ & C \leq \text{column } (x_j - x_i) \leq D \\ \infty, & \text{otherwise.} \end{cases}$$

The values A, B, C, and D were typically set by taking a number of sample pictures and determining the smallest box encompassing the variation in the relative position between the components i and j. A subroutine was written to perform this task and automatically set the derived parameters into the LEA.

In our current implementation, for a typical 35×39 (face) picture, we require 13 s to compute all the L(EV)A's, and 35 s to execute the LEA (these times include picture input from a disk library, and storage of results on the disk). Most of the programming is in Fortran, and the computer is an IBM/360/40 (the addition instruction on the 360/40 executes in 11.88 μs). Total core and disk storage in bytes for all arrays is $M(4f+2h)$ and $NM(f+2h)$, respectively, where M, N, f, h are the number of resolution cells in the sensed image, the number of L(EV)A's, the amount of core needed for a floating point number, and the amount of core required for a (small) integer. For a typical picture in the face experiments ($M=1600$, $N=8$, $f=4$, $h=2$), these requirements work out to approximately 32 K bytes of core, and 100K bytes of disk storage.

Discussion

Many, though by no means all, visual objects can be described by breaking down the object into a number of more "primitive parts," and by specifying an allowable range of spatial relations which these "primitive parts" must satisfy for the object to be present.

As an example, suppose we want to describe a frontal view of a standing person. This visual object could be decomposed into six primitive pieces: a head, two arms, a torso, and two legs. For this visual object to be present in an actual picture, it is required that these six primitives occur (or at least that some significant subset of them occurs), and also that they occur within a certain spatial relationship one to the other—that is, the legs should be next to each other, and below the torso; the torso should be between and below the tops of the two arms; and the head should be on top of the torso.

It may be noticed that in the previous two paragraphs, we implicitly separated the local aspects from the global aspects of the description; the local aspects are the primitive parts of the picture, and the global aspects are the spatial relations between these parts. While at first glance it does not seem unnatural to make this separation, in practice there is a frequently encountered difficulty, that is, the feedback between the local and the global.

To illustrate this difficulty, let us go back to the example of the view of a standing person. Any method which detects torsos on a local level (that is, a method which detects torsos without using any knowledge of the positions of nearby arms and legs) might very well detect several torsos in a picture. In fact, the actual or "true" torso may be one of the weaker of the torsos detected by the method; it may even happen that the true torso is not detected at all. What does determine the position of the true torso is the position of the true arms, legs, and head. But, unfortunately, the reverse is true. The positions of, say, the true arms depends on the position of the true torso. Thus, the possible position of each piece affects the possible position of each other piece, making for a circular type of dependency.

It seems that whatever the visual object is, whenever we try to separate the global and the local, the same circular dependency occurs. Many times attempts to recognize visual objects described in such a way involve alternating between local and global analysis, heuristics, backup procedures, etc.

One conceptual way of avoiding this circularity is to evaluate simultaneously a complete interpretation of the picture; e.g., in the example given above, we would look at, and evaluate, complete configurations of head, arms, legs, torso, etc. The best complete interpretation could then be chosen. This approach, however, requires that we make an infeasibly large number of evaluations. It was just this computational problem that in the first place led to the decomposition approach.

The implication of the above discussion is that, in general, we cannot hope to decompose the global evaluation problem into a number of smaller *independent* problems, but rather must use something akin to the simultaneous evaluation, taking advantage of any reduction in total variable interdependency to reduce the required number of such evaluations.

In this paper, we accomplish this through the following machinery. First, an embedding metric is presented which sets the framework for evaluating how well any composition of primitive picture pieces (parts of the decomposed picture) matches the desired composite picture. Second, a sequential optimization (dynamic programing-type) algorithm is developed which takes advantage of the decomposition to reduce drastically the computational requirements (our computational requirements grow linearly with the size of the picture, rather than exponentially). The contribution of this paper is the simultaneous offering of the above two components and their suitability for application to a wide class of pictorial objects.

In addition to the image-matching application, which was the center of most of the development in this paper, we have also attempted to establish the utility of the representational aspects of the embedding metric for general picture description applications.

The work presented here is a continuation of the investigation described in [1] and [2], where Fischler uses sequential optimization for matching two-dimensional scenes, introduces the generic form of the embedding metric elaborated on here, and presents the concepts of coherent segmentation, arbitrary serialization, and sequential constraints. The relation of the heuristic embedding problem to formal decision theory is also discussed. The only other paper in which sequential optimization is applied to a broad *class* of problems involving two-dimensional scenes is where Martelli and Montanari [4] present a metric and matched algorithm for smoothing pictures. Kovalewsky [5] and Montanari [3] have applied dynamic programing to the detection of (one-dimensional) line-like pictures. Reference [3] is an outstanding paper, in which Montanari provides much insight into the characteristics of sequential optimization. Both Kovalewsky [5]' and Montanari [3] comment on the representational aspects associated with their optimization procedures. An embedding metric conceptually very similar to the one given in [1], [2], and this paper is discussed in a broad and interesting work by Bremermann[10] [12] with respect to its potential use in character recognition, speech recognition, and control of effectors (e.g., manipulators).

Acknowledgment

The authors wish to thank O. Firschein and J. Tenenbaum for many constructive suggestions relative to the organization of this paper.

References

[1] M. A. Fischler, "The detection of scene congruence," Lockheed Missiles & Space Company, Inc., Palo Alto, Calif., Rep. 6-83-71-2, Jan. 1971.

[2] M. A. Fischler, "Aspects of the detection of scene congruence," in *Proc. 2nd Int. Joint Conf. Artificial Intelligence* (Advance Paper), Sept. 1971, pp. 88–100.

[3] U. Montanari, "On the optimal detection of curves in noisy pictures," *Commun. Ass. Comput. Mach.*, vol. 14, pp. 335–345, May 1971.

[4] A. Martelli and U. Montanari, "Optimal smoothing in picture processing," in *Proc. IFIP Congr.* Amsterdam, The Netherlands: North-Holland, 1971.

[5] V. A. Kovalewsky, "Sequential optimization in pattern recognition and pattern description," in *Proc. IFIP Congr.* Amsterdam, The Netherlands: North-Holland, 1968, pp. 146–151.

[6] R. Bellman and S. Dreyfus, *Applied Dynamic Programming.* Princeton, N. J.: Princeton Univ. Press, 1962.

[7] U. Bertelé and F. Brioschi, "A new algorithm for the solution of the secondary optimization problem in nonserial dynamic programming," *J. Math. Anal. Appl.*, vol. 27, no. 3, pp. 565–574, 1969.

[8] O. Firschein and M. A. Fischler, "Describing and abstracting pictorial structures," *Pattern Recognition*, vol. 3, pp. 421–444, Nov. 1971.

[9] A. Rosenfeld, *Picture Processing by Computer.* New York: Academic, 1969.

[10] W. F. Miller and A. S. Shaw, "Linguistic methods in picture processing—A survey," in *1968 Fall Joint Comput. Conf., AFIPS Conf. Proc.*, vol. 33. Washington, D. C.: Thompson, 1968, pp. 279–290.

[11] M. D. Kelly, "Edge detection in pictures by computer using planning," in *Machine Intelligence 6*, B. Meltzer and D. Michie, Ed. New York: Elsevier, 1971, pp. 397–410.

[12] H. J. Bremermann, "Cybernetic functionals and fuzzy sets," in *Ann. Symp. Rec. 1971 IEEE Syst., Man Cybern. Group*, Oct. 1971, pp. 248–253.

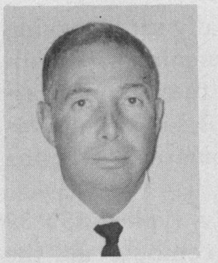
[13] ——, "Pattern recognition, functionals, and entropy," *IEEE Trans. Bio-Med. Eng.*, vol. BME-15, pp. 201–207, July 1968.

[14] ——, "What mathematics can and cannot do for pattern recognition," in *Pattern Recognition in Biological and Technical Systems*, O. J. Grüsser, Ed. Heidelberg, Germany: Springer, 1971, pp. 31–45.

[15] W. W. Bledsoe, "The model method in facial recognition," Panoramic Research, Inc., Palo Alto, Calif., Rep. PRI:15, Aug. 1966.

[16] ——, "Man-machine facial recognition," Panoramic Research, Inc., Palo Alto, Calif., Rep. PRI:22, Aug. 1966.

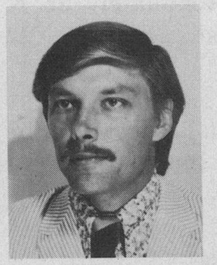
[17] A. J. Goldstein, L. D. Harmon, and A. B. Lesk, "Identification of human faces, " *Proc. IEEE*, vol. 59, pp. 748–760, May 1971.



Martin A. Fischler (S'57–M'58) was born in New York, N. Y., on February 15, 1932. He received the B.E.E. degree from the City College of New York, New York, in 1954 and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, Calif., in 1958 and 1962, respectively.

He served in the U. S. Army for two years and held positions at the National Bureau of Standards and at Hughes Aircraft Corporation during the period 1954 to 1958. In 1958 he joined the technical staff of the Lockheed Missiles & Space Company, Inc., at the Lockheed Palo Alto Research Laboratory, Palo Alto, Calif., and currently holds the title of Staff Scientist. He has conducted research and published in the areas of artificial intelligence, picture processing, switching theory, computer organization, and information theory.

Dr. Fischler is a member of the Association for Computing Machinery, the Pattern Recognition Society, the Mathematical Association of America, Tau Beta Pi, and Eta Kappa Nu. He is currently an Associate Editor of the journal *Pattern Recognition* and is a past Chairman of the San Francisco Chapter of the IEEE Society on Systems, Man, and Cybernetics.



Robert A. Elschlager was born in Chicago, Ill., on May 25, 1943. He received the B.S. degree in mathematics from the University of Illinois, Urbana, in 1964, and the M.S. degree in mathematics from the University of California, Berkeley, in 1969.

Since then he has been an Associate Scientist with the Lockheed Missiles & Space Company, Inc., at the Lockheed Palo Alto Research Center, Palo Alto, Calif. His current interests are picture processing, operating systems, computer languages, and computer understanding.

Mr. Elschlager is a member of the American Mathematical Society, the Mathematical Association of America, and the Association for Symbolic Logic.

[10] Other relevant publications by Bremermann include [13] and [14].