Shape Matching of Two-Dimensional Objects

BIR BHANU, MEMBER, IEEE, AND OLIVIER D. FAUGERAS, MEMBER, IEEE

Abstract—In this paper we present results in the areas of shape matching of nonoccluded and occluded two-dimensional objects. Shape matching is viewed as a "segment matching" problem. Unlike the previous work, the technique is based on a stochastic labeling procedure which explicitly maximizes a criterion function based on the ambiguity and inconsistency of classification. To reduce the computation time, the technique is hierarchical and uses results obtained at low levels to speed up and improve the accuracy of results at higher levels. This basic technique has been extended to the situation where various objects partially occlude each other to form an apparent object and our interest is to find all the objects participating in the occlusion. In such a case several hierarchical processes are executed in parallel for every object participating in the occlusion and are coordinated in such a way that the same segment of the apparent object is not matched to the segments of different actual objects. These techniques have been applied to two-dimensional simple closed curves represented by polygons and the power of the techniques is demonstrated by the examples taken from synthetic, aerial, industrial and biological images where the matching is done after using the actual segmentation methods.

Index Terms—Coordination, hierarchical relaxation, occlusion, optimization, penalty function approach, projection gradient method, recognition, relaxation, segment matching, shape matching, stochastic labeling, 2-D objects.

I. Introduction

THE PROBLEM of assigning names or labels to a set of units/objects is the key problem in computer vision, image analysis, and pattern recognition. Since all the labels are not possible for a given unit, constraints based on contextual information, called the world model, are used to obtain a consistent and unambiguous valid assignment of the units. Local parallel processes are a very efficient way of assigning labels. The features of such algorithms include the propagation of local contextual information in a paradigm of competition and cooperation, locality, and speed. In general, the task of assigning names to units only on the basis of features of the units is very difficult since any segmentation based on low-level analysis is bound to contain errors and the computed features are noisy. The solution to this problem is to delay any firm commitment until all the contextual information has been used. Depending upon the type of constraints embodying the world model, the problem can be attacked by discrete methods (discrete relaxation) or continuous methods (continuous relaxation, also called stochastic labeling). Recent theoretical development and surveys on these algorithms applied to low level vision and symbolic matching can be found in [1]-[3].

In this paper we solve the "segment matching" problem [4] of shape matching, defined as the recognition of a piece of a shape as an approximate match to a part of larger shape, by using a hierarchical stochastic labeling technique [5]. The technique explicitly maximizes a criterion function based on the ambiguity and inconsistency of classification. The hierarchical nature of the algorithm reduces the computation time and uses results obtained at low levels to speed up and improve the accuracy of results obtained at higher levels. The class of shapes that we consider are represented by simple closed curves and are two-dimensional in nature such as the boundary of a region in an aerial image, outlines of biological cells, industrial parts, etc. These shapes are approximated by polygons. The shapes are allowed to undergo translation, rotation, scaling, and in general significant changes. Taking these variations into account is essential if one is concerned with real images, because in practice the results of a segmentation technique will be different when it is applied to the images of the same scene taken under different conditions. In the next section we present the basic hierarchical stochastic labeling technique to do shape matching of 2-D nonoccluded objects. Section III extends this technique to the shape matching of partially occluded objects. Several synthetic and real examples are presented. Finally, Section IV presents the conclusions of the paper.

II. Shape Matching of 2-D Nonoccluded Objects

Past techniques used in the shape matching of 2-D nonoccluded objects are: chain code cross-correlation, Fourier descriptors and moments, statistical pattern recognition techniques, symbolic matching, syntactic and relaxation methods. A review of these techniques can be found in [5], [6]. We solve the segment matching problem by extending the stochastic labeling technique [7], [8] in a hierarchical manner. The shape matching algorithm is hierarchical in the sense that at the higher levels, there are more constraints and world knowledge. In the following we present a two stage hierarchical stochastic labeling method for matching the segments of a template/model against the segments of an observed object.

Hierarchical Stochastic Labeling Algorithm: Let $T = (T_1, T_2, \cdots, T_N)$ and $O = (O_1, O_2, \cdots, O_{L-1})$ be the polygonal representation of the model and the object, respectively, where T_i and O_j are line segments, $i = 1, \cdots, N$ and $j = 1, \cdots, L-1$. In general, L may be greater, equal to or less than N. Model elements will be referred to as units and object elements

Manuscript received August 5, 1982; revised September 19, 1983. This work was supported in part by DARPA Contract F-33615-80-C-1080.

B. Bhanu is with the Aeronutronic Division, Ford Aerospace and Communications Corporation, Newport Beach, CA 92660.

O. D. Faugeras is with INRIA, Rocquencourt, France, and the University of Paris XI, Paris, France.

as classes. We identify part of the model T within the observation O. We are therefore trying to label each of the segments T_i ($i = 1, \cdots, N$) either as a segment O_j ($j = 1, \cdots, L - 1$) or as not belonging to O (label O_L = Nil). Thus each segment T_i can have L possible labels. Using a technique described subsequently we compute for every segment T_i a set of L positive numbers $p_i(l)$, $l = 1, \cdots, L$, forming a vector $\vec{p}_i = [p_i(1), \cdots, p_i(L)]^T$. $p_i(l)$ can be thought of as the probability of labeling the segment T_i as O_l, so $\Sigma_{l=1}^{L} p_i(l) = 1$ and $p_i(l) \geq 0$. The set of all vectors $\vec{p}_i$ ($i = 1, \cdots, N$) is called a stochastic labeling of the set of units.

Initially the stochastic labeling is ambiguous (except in some very special cases) and we make it evolve toward a less ambiguous labeling by comparing the local structure of T and O. From now on the indexes i are taken modulo N. To every segment T_i, we associate the two neighboring segments T_{i-1} and T_{i+1}. In order to compare the local structures of T and O, a world model is represented by two compatibility functions C_1 and C_2 mapping $S_2 \times O^2$ and $S_3 \times O^3$ into [0, 1] where, S_2 and S_3 are two subsets of T^2 and T^3 defined by

$$S_2 = \{(T_i, T_j)\}, \quad i = 1, \cdots, N, \quad j = i - 1 \text{ or } i + 1$$

$$S_3 = \{(T_i, T_{i-1}, T_{i+1})\}, \quad i = 1, \cdots, N.$$

The compatibility function $C_1(T_i, O_k, T_j, O_l)$ ($j = i - 1$ or $i + 1$) and $C_2(T_i, O_k, T_{i-1}, O_l, T_{i+1}, O_m)$ will be denoted more simply by $C_1(i, k, j, l)$ and $C_2(i, k, i - 1, l, i + 1, m)$. In certain situations C_1 and C_2 may have conditional probability interpretation [1], but here they do not. $C_1(i, k, i - 1, l)$ measures the resemblance of the set $\{T_i, T_{i-1}\}$ with the set $\{O_k, O_l\}$. Similarly, $C_2(i, k, i - 1, l, i + 1, m)$ measures the resemblance of the set $\{T_i, T_{i-1}, T_{i+1}\}$ with the set $\{O_k, O_l, O_m\}$. A good (bad) match means that the value of C_1 or C_2 is close to 1 (0). We associate to every segment T_i a compatibility vector $\vec{q}_i = [q_i(1), \cdots, q_i(L)]^T$. Intuitively, this vector represents what the neighbors of segment T_i (that is to say segment T_{i-1} and T_{i+1}) "think" about the way it should be labeled whereas $\vec{p}_i$ represents what the segment T_i "thinks" about its own labeling. Mathematically speaking we compute

$$Q_{ij}(k) = \sum_{l=1}^{L} C_1(i, k, j, l)\, p_j(l), \quad \begin{array}{l} j = i - 1, i + 1 \\ i = 1, \cdots, N \\ k = 1, \cdots, L \end{array} \tag{1}$$

$$Q_i^{(1)}(k) = \tfrac{1}{2}\left(Q_{ii-1}(k) + Q_{ii+1}(k)\right) \tag{2}$$

$$Q_i^{(2)}(k) = \sum_{l_1, l_2 = 1}^{L} C_2(i, k, i - 1, l_1, i + 1, l_2) \cdot p_{i-1}(l_1)\, p_{i+1}(l_2). \tag{3}$$

Using vector-matrix notation (1), (2), and (3) can be written more simply as

$$\vec{Q}_{ij} = A_{ij}\vec{p}_j \tag{4}$$

$$\vec{Q}_i^{(1)} = \tfrac{1}{2}(A_{ii-1}\vec{p}_{i-1} + A_{ii+1}\vec{p}_{i+1}) \tag{5}$$

$$Q_i^{(2)}(k) = \vec{p}_{i-1}^{\,T} B_{ik}\vec{p}_{i+1} \tag{6}$$

where A_{ij} and B_{ik} are $L \times L$ matrices. The (k, l)th element A_{ij} and (l_1, l_2)th element of B_{ik} are $C_1(i, k, j, l)$ and $C_2(i, k, i - 1, l_1, i + 1, l_2)$, respectively. The numbers $Q_i^{(1)}(k)$ and $Q_i^{(2)}(k)$, $k = 1, \cdots, L$ are positive. The idea is that they are large when the probabilities of the labels of the neighbors of T_i compatible with label O_k are large and small otherwise. The numbers $Q_i^{(1)}(k)$ and $Q_i^{(2)}(k)$ are normalized so that they add up to 1 yielding two vectors $\vec{q}_i^{(1)}$ and $\vec{q}_i^{(2)}$ such that

$$q_i^{(n)}(k) = \frac{Q_i^{(n)}(k)}{\sum_{l=1}^{L} Q_i^{(n)}(l)}, \quad \begin{array}{l} n = 1, 2 \\ k = 1, \cdots, L. \end{array} \tag{7}$$

It is desired to decrease the discrepancy between what every segment T_i thinks about its own labeling ($\vec{p}_i$) and what its neighbors think about it ($\vec{q}_i^{(n)}, n = 1, 2$). We can therefore define local consistency as the amount of difference between $\vec{p}_i$ and $\vec{q}_i^{(n)}$ ($n = 1, 2$). A good measure of it is the angle between these two vectors,

$$C_i^{(n)} = \cos\theta_i = \frac{\vec{p}_i \cdot \vec{q}_i^{(n)}}{\|\vec{p}_i\|_2\, \|\vec{q}_i^{(n)}\|_2}, \quad n = 1, 2 \tag{8}$$

when $\vec{p}_i = \vec{p}_i^{(n)}$, $\cos\theta_i = 1$ and is smaller than 1 if $\vec{p}_i$ and $\vec{q}_i^{(n)}$ are different with a minimum value of 0 (because $\vec{p}_i$ and $\vec{q}_i^{(n)}$ are probability vectors). Similarly, a local measure of ambiguity can be defined as the quadratic entropy,

$$H_i = \sum_{l=1}^{L} p_i(l)(1 - p_i(l)) = 1 - \|\vec{p}_i\|_2^2. \tag{9}$$

Since H_i is large when $\|\vec{p}_i\|$ is small and vice versa, we can use

$$H_i' = \|\vec{p}_i\|_2 \tag{10}$$

as a local measure of ambiguity. Further, since we want to maximize local consistency and minimize ambiguity, we can use the product of (8) and (10) as a measure of both quantities, i.e.,

$$C_i^{(n)} H_i' = \frac{\vec{p}_i \cdot \vec{q}_i^{(n)}}{\|q_i^{(n)}\|_2} \tag{11}$$

$C_i^{(n)} H_i'$ is maximum when $\vec{p}_i = \vec{q}_i^{(n)}$ (maximum consistency) and $\vec{p}_i$ is the unit vector (minimum ambiguity). We can define

$$J_i'^{(n)} = \|\vec{q}_i^{(n)}\|_2 C_i^{(n)} H_i' = \vec{p}_i \cdot \vec{q}_i^{(n)}, \quad n = 1, 2 \tag{12}$$

as a local measure of ambiguity and consistency. Note that $J_i'^{(n)}$ is maximum for $\vec{p}_i = \vec{q}_i^{(n)}$ and $\vec{p}_i$ = unit vector. Therefore, a good "local" measure of ambiguity and consistency is the inner product $\vec{p}_i \cdot \vec{q}_i^{(n)}$, $n = 1, 2$. By computing the average over the set T of these local measures we obtain two global criteria:

$$J^{(n)} = \sum_{i=1}^{N} \vec{p}_i \cdot \vec{q}_i^{(n)}, \quad n = 1, 2. \tag{13}$$

The problem of labeling the segments T_i is therefore equivalent to an optimization problem: given an initial labeling $\vec{p}_i^{(0)}$, $i = 1, \cdots, N$, find a local maximum of the criteria $J^{(n)}$ (n = 1, 2) closest to the original labeling $\vec{p}_i^{(0)}$ subject to the constraints that the $\vec{p}_i$'s are probability vectors. Since C_2 is a better measure than C_1 of the local match between T and O, we are actually interested in finding local maxima of the criterion $J^{(2)}$. On the other hand, maximizing $J^{(1)}$ is easier from the computational standpoint. We therefore use the following hierarchical approach: starting with an initial labeling $\vec{p}_i^{(0)}$, we look for a local maximum $\vec{p}_i^{(1)}$ of the criterion $J^{(1)}$. This labeling is less ambiguous than $\vec{p}_i^{(0)}$ in the sense that many labels have been dropped (their probabilities $p_i(k)$ are equal to zero). We then use the labeling $\vec{p}_i^{(1)}$ as an initial labeling to find a local maximum of the criterion $J^{(2)}$. The computational saving comes from the fact that the values $C_2(i, k, i-1, l_1, i+1, l_2)$ corresponding to probabilities $p_{i-1}(l_1)$ or $p_{i+1}(l_2)$ equal to zero are not computed.

The problem of maximizing (13) can be efficiently solved using the gradient projection method [9]. The gradient of the criterion at the first stage of hierarchy is given by

$$\frac{\partial J^{(1)}}{\partial \vec{p}_i} = \vec{q}_i^{(1)} + \sum_j \frac{1}{D_j} [A_{ji}^T \vec{p}_j - (\vec{p}_j \cdot \vec{q}_j^{(1)})(A_{ji}^T \vec{x})] \tag{14}$$

where

$$D_j = \sum_{l=1}^{L} Q_j^{(1)}(l) \tag{15}$$

and $\vec{x}$ is a $L \times 1$ vector of 1's, i.e., $\vec{x} = [1, 1, \cdots, 1]^T$. The first term in (14) corresponds to the simple maximization of the product $\vec{p}_i \cdot \vec{q}_i^{(1)}$ in the global criterion $J^{(1)}$, and the second term corresponds to the coupling between units through the compatibility function C_1. Note that in general $C_1(j, l, i, k) \neq C_1(i, k, j, l)$ since it depends upon the manner in which the compatibility is computed. At the second stage of hierarchy the gradient of the criterion $J^{(2)}$ is obtained as

$$\frac{\partial J^{(2)}}{\partial p_i(k)} = q_i^{(2)}(k) + \sum_j \frac{1}{D_j} [\tilde{Q}_j(k) - D_j' \vec{p}_j \cdot \vec{q}_j^{(2)}],$$
$$j = i-1, i+1 \tag{16}$$

where

$$\tilde{Q}_{i-1}(k) = \vec{p}_{i-1}^T C \vec{p}_{i-2}, \tilde{Q}_{i+1}(k) = \vec{p}_{i+1}^T D \vec{p}_{i+2} \tag{17}$$

$$D_{i-1}' = \frac{\partial D_{i-1}}{\partial p_i(k)} = \vec{x}^T C \vec{p}_{i-2}, D_{i+1}' = \frac{\partial D_{i+1}}{\partial p_i(k)} = \vec{x}^T D \vec{p}_{i+2} \tag{18}$$

$$D_{i-1} = \sum_{l=1}^{L} Q_{i-1}^{(2)}(l), D_{i+1} = \sum_{l=1}^{L} Q_{i+1}^{(2)}(l). \tag{19}$$

C and D are $L \times L$ matrices, whose (l_1, l_2)th element is given by $C_2(i-1, l_1, i-2, l_2, i, k)$ and $C_2(i+1, l_1, i+2, l_2, i, k)$, respectively. Now the iteration of $\vec{p}_i$'s is given by

$$\vec{p}_i^{(n+1)} = \vec{p}_i^{(n)} + \rho_i^{(n)} \vec{P}_i^{(n)} \left[\frac{\partial J^{(n)}}{\partial \vec{p}_i}\right], \quad \begin{matrix} i = 1, \cdots, N \\ n = 1, 2 \end{matrix} \tag{20}$$

where the projection of the gradient is given by

$$\vec{P}_i^{(n)} \left[\frac{\partial J^{(n)}}{\partial \vec{p}_i}\right] = \frac{\partial J^{(n)}}{\partial \vec{p}_i} - \frac{1}{L} \sum_{k=1}^{L} \frac{\partial J^{(n)}}{\partial p_i(k)} = \vec{A}_i^{(n)} \tag{21}$$

if none of the components of $\vec{p}_i$ are zero. The computation of the projection when some of the components of $\vec{p}_i$ are zero can be found in [7], [9]. Normally, $\rho_i^{(n)}$ is kept constant for all units during each iteration and is determined to have the largest possible value such that $\vec{p}_i$'s at the $(n+1)$th iteration still lie in the bounded convex region of the LN-dimensional Euclidean space defined by $\Sigma_{k=1}^{L} p_i(k) = 1$ and $p_i(k) \geqslant 0, i = 1, \cdots, N$. However, to obtain a faster convergence rate $\rho_i^{(n)}$ is obtained as

$$\rho_i^{(n)} = \alpha \cdot \min_i [\max_k \rho_i^{(n)}(k)] \tag{22}$$

where α is a constant between 0 and 1 and can be used to control the rate of convergence. Normally it is taken as 0.99. $\rho_i^{(n)}(k)$ is given by

$$\rho_i^{(n)}(k) = \begin{cases} \dfrac{1 - p_i^{(n)}(k)}{A_i^{(n)}(k)}, & A_i^{(n)}(k) > 0 \\[2ex] \dfrac{p_i^{(n)}(k)}{A_i^{(n)}(k)}, & A_i^{(n)}(k) < 0. \end{cases} \tag{23}$$

A side effect of computing $\rho_i^{(n)}$ for every unit is that we may not be following the gradient exactly. However, it can be expected that we are approximately in the direction of the gradient and the criterion (13) is still maximized. It is evidenced by a large number of experiments. Now we present the details of the shape matching algorithm.

Polygonal Approximation and Features: Polygonal approximation for the model and object is obtained by detecting the points of high curvature [10]. The features derived from the polygonal approximation of the boundary of an object are: length of a segment, intervertices distance, slope of a segment, angle between the two segments called the interior angle, and angle between the two segments called the exangle as shown in Fig. 1. The exangle corresponding to a vertex is equal to the angle between the two straight lines, where one line is obtained by extending the line joining this vertex and its neighboring counterclockwise vertex and the other line is obtained by extending the line joining the two clockwise neighboring vertices.

Initial Assignment of Probabilities: The initial assignment of probabilities for a unit is obtained by comparing its feature values with the feature values of all the segments of the object. In general, the quality of correspondence of a unit i to an object segment k is given by

$$M(T_i, O_k) = \sum_{p=1}^{P} |f_{tp} - f_{op}| W_p \tag{24}$$

where P is the total number of features

f_{tp} = pth feature value for the model segment

f_{op} = pth feature value for the object segment

W_p = weight factor for the pth feature.

Note that for a perfect match $M(T_i, O_k) = 0$ and for a poor match $M(T_i, O_k)$ will be large. The initial probabilities chosen proportional to $1/(1 + M(T_i, O_k))$, for $k = 1, \cdots, L-1$ are normalized so that they sum to 1. The weights of features are needed to account for their importance and the different range of values.

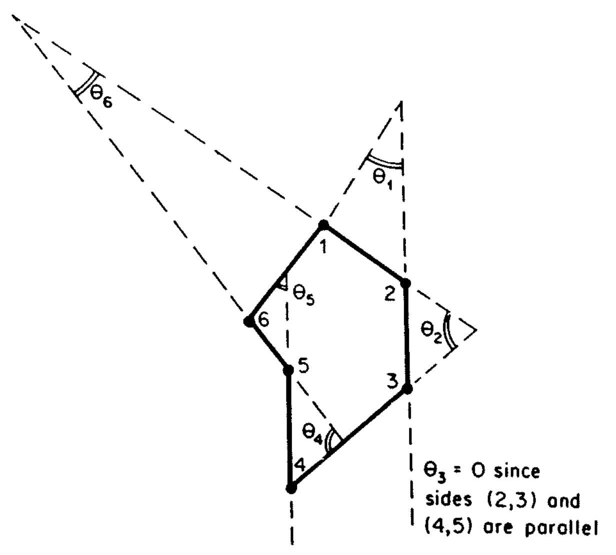


Fig. 1. Illustration of the definition of exangle.

Computation of Compatibilities: The compatibility function determines the degree by which the assignments of two or three neighboring units are compatible with each other. C_1 and C_2 take values between 0 and 1. There are at least 4 ways of computing the compatibilities C_1 and C_2.

First Method: At the first stage, we compute a transformation TR from a unit T_i to label O_k, i.e., $TR: T_i \rightarrow O_k$. TR consists of scaling, rotation, and translation in the X and Y directions. This transformation is applied to the unit $T_j (j = i - 1$ or $i + 1)$ and the error between the transformed T_j and O_l is

$$M(TR(T_j), O_l) = \sum_{p=1}^{P} |f_{t'p} - f_{op}| W_p \tag{25}$$

where $f_{t'p}$ = pth feature value for the transformed unit, and the other quantities are similar to those defined in (24).

Note that here the features may be slope and length of a segment, so we shall need the weights for these features. However, it is possible to avoid these parameters if we use only the distance between the ends of O_l and transformed T_j as the matching error between two segments, i.e., matching error $M(TR(T_j), O_l) = AB + CD$ [refer Fig. 2(a)]. In practice we have used this approach for computing the matching error. The compatibility at the first stage is given by

$$C_1(i, k, j, l) = \frac{1}{1 + M(TR(T_j), O_l)}.$$

The problem with this method of computing the compatibilities is that they are not symmetric, i.e., $C_1(i, k, j, l) \neq C_1(j, l, i, k)$. As we have seen, the computation of the gradient requires $C_1(j, l, i, k)$, so if this method is used we shall also require the computation of $C_1(j, l, i, k)$. Moreover, since we are using only one transformation, compatibilities so obtained will not be very accurate compared to the other three methods described below.

Second Method: Unlike the first method, here we find two transformations $TR1$ and $TR2$ such that

$$TR1: T_i \rightarrow O_k \text{ and } TR2: T_j \rightarrow O_l.$$

Now the average rotation, average scale and average translation of these two transformations are computed. The transformation associated with these parameters called TV, is now applied to the unit i and unit j and the matching errors between the transformed units and the segments O_k and O_l are computed as in the first method and finally,

$$C_1(i, k, j, l) = \frac{1}{1 + \text{total error}}.$$

At the second stage instead of finding two transformations, we find three transformations and take the average of these values. This average transformation is then applied to units $i, i - 1, i + 1$ and the total error between the transformed units and object segments k, l_1, and l_2 is computed to get $C_2(i, k, i - 1, l_1, i + 1, l_2)$ as in the first stage compatibility computation.

Third Method: This method is similar to the second method in that we compute two transformations $TR1$ and $TR2$. Now $TR1$ is applied to T_j giving matching error $M(TR1(T_j), O_l)$ and $TR2$ is applied to T_i giving matching error $M(TR2(T_i), O_k)$. Average of this error is taken and

$$C_1(i, k, j, l) = \frac{1}{1 + \text{average error}}.$$

At the second stage, we will find three transformations and the average error will be the average of six error terms and the compatibility

$$C_2(i, k, i - 1, l_1, i + 1, l_2) = \frac{1}{1 + \text{average error}}$$

Fourth Method: In this method we compute mathematically the best transformation from units i and j such that the sum of the squares of the error between the transformed units and the object segments is minimum. Here we can use only the distance in the computation of the matching error (unlike the first three methods where in principle we could use a combination of slope and length) so that the error criterion is quadratic and linear least squares techniques can be applied. For example, let the beginning and end coordinates of the segments i, j, k, and l be given by $(X1, Y1)$, $(X2, Y2)$, $(DX1, DY1)$, $(DX2, DY2)$, $(R1, S1)$, $(R2, S2)$, $(U1, V1)$ and $(U2, V2)$, respectively, then

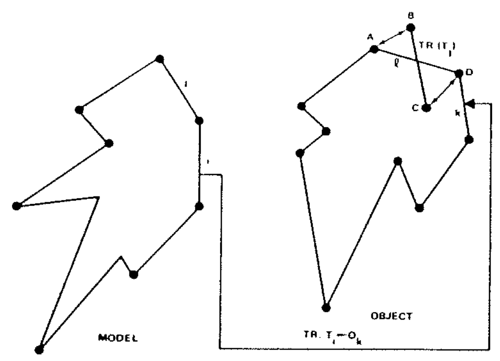
$$MX = b$$

where

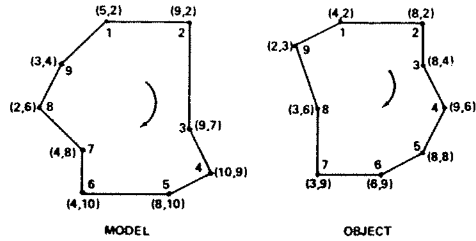
$$M = \begin{bmatrix} DX1 & -DY1 & 1 & 0 \\ DY1 & DX1 & 0 & 1 \\ DX2 & -DY2 & 1 & 0 \\ DY2 & DX2 & 0 & 1 \\ X1 & -Y1 & 1 & 0 \\ Y1 & X1 & 0 & 1 \\ X2 & -Y2 & 1 & 0 \\ Y2 & X2 & 0 & 1 \end{bmatrix}, \quad X = \begin{bmatrix} \lambda \cos \theta \\ \lambda \sin \theta \\ X_0 \\ Y_0 \end{bmatrix}, \quad b = \begin{bmatrix} U1 \\ V1 \\ U2 \\ V2 \\ R1 \\ S1 \\ R2 \\ S2 \end{bmatrix}$$

and λ is a scaling factor, θ is the rotation, and X_0 and Y_0 are the translations in the X and Y directions, respectively. The

(a)



(b)



METHOD	FIRST	SECOND	THIRD	FOURTH
$C_1(i,k,j,l)$ $i=1,k=6,j=9,l=5$				
Transformation- (Scale, Rotation, X-Trans., Y-Trans.)	TR: $T_1 \rightarrow O_6$ (0.75,-180° 9.75,10.5)	TR1, Same as TR TR2: $T_9 \rightarrow O_5$ (0.79,-161.56° 9.25,11.75) Avg. Trans., TV (0.77,-170.78° 9.50,11.12)	TR1 and TR2 as in the second method	Least Squares Best Transfo- rmation, from T_1 to O_6 and T_9 to O_5 (0.77,-173.29° 9.77,11.18)
Transformation Applied to Units	9	1,9	TR1 to 9 TR2 to 1	1,9
Beginning & End Coordinates (X,Y) of the Transformed Units	Unit 9 (7.5,7.5), (6,9)	Unit 1 (5.94,8.98), (2.91,8.49) Unit 9 (7.71,7.71), (5.94,8.98)	Unit 9 (7.5,7.5), (6,9) Unit 1 (6,9), (3,8)	Unit 1 (6.09,9.18), (3,8.81) Unit 9 (7.81,7.81), (6.09,9.18)
Total Error Between the Transformed Units and object Labels	0.707	1.033	1.707	0.844
Average Error			0.845	
Compatibility	0.586	0.492	0.540	0.542
$C_2(i,k,i-1,l_1,$ $i+1,l_2), i=1,k=6,$ $l_1=5, l_2=7)$				
Transformation- (Scale, Rotation, X-Trans., Y-TRans.)	TR as above	TR1 and TR2 as above TR3: $T_2 \rightarrow O_7$ (0.6,-180°, 8.4,10.2) Avg. Trans., TV (0.72,-173.85°, 9.13,10.81)	TR1, TR2 and TR3 as in the Second Method	Least Squares Best Transfo- rmation, from T_1 to O_6, T_9 to O_5 and T_2 to O_7 (0.72,-179.2° 9.63,10.68)
Transformation Applied to Units	9,2	1,9,2	TR1 to 9,2 TR2 to 1,2 TR3 to 1,9	1,9,2
Beginning & End Coordinates (X,Y) of the Transformed Units	Unit 9, Same as Above, Unit 2 (3,9) (3,5.25)	Unit 1 (5.73,9.01), (2.9,8.71) Unit 9 (7.31,7.75), (6.73,9.01) Unit 2 (2.9,8.71) (3.28,5.16)	TR1: Unit 9 (7.5,7.5),(6,9) Unit 2 (3,9),(3,5.25) TR2: Unit 1 (6,9),(3,8) Unit 2 (3,9),(3,5.25) TR3: Unit 1 (5.4,9),(3,9) Unit 9 (6.6,7.8), (5.4,9)	Unit 1 (6.03,9.19) (3.13,9.15) Unit 9 (7.49,7.76), (6.03,9.19) Unit 2 (3.13,9.15) (3.1,5.53)
Total Error Between the Transformed Units and object Labels	1.457	2.752	5.820	1.856
Average Error			0.970	
Compatibility	0.407	0.267	0.508	0.350

Fig. 2. (a) Matching distance error = $AB + CD$. (b) An example illustrating the four methods of computing compatibilities C_1 and C_2.

previous set of equations is an overdetermined system. It can be transformed as $M^TMX = M^Tb$ which can usually be uniquely solved for λ, θ, X_0, Y_0. This computed transformation can then be applied to obtain compatibilities at the first stage by using the approach used in the second method. At the second stage M is a 12×4 mátrix and b a 12×1 column vector. It can be solved exactly as in the first stage case to obtain the best transformation and compatibilities at the second stage. This method requires more computation time than any of the other methods. Note that the transformations used in this method allow polygons to undergo rotation, translation and scaling deformations only. However, a more general transformation can be easily computed. Also note that the second, third, and fourth method lead to symmetric compatibilities. In practice we have chosen the third method. Fig. 2(b) demonstrates the four methods of computing C_1 and C_2 on a specific example.

Initial Probability and Compatibility for the Nil Class: $p_i(\text{Nil})$ is assigned a small constant value, depending upon the *a priori* information that we may have about the possible number of matches. Normally, we have taken $p_i(\text{Nil})$ between 0.05 to 0.30. The actual value is not critical, however, it affects the convergence of probabilities, hence the number of iterations. The compatibilities involving the Nil class are assigned as follows.

$$\begin{aligned} C_1(i,k,i-1,\text{Nil}) &= C_1(i,k,i+1,\text{Nil}) \\ &= C_2(i,k,i-1,\text{Nil},i+1,\text{Nil}) \\ &= p_i(k) \\ C_1(i,\text{Nil},i-1,l) &= C_1(i,\text{Nil},i+1,l) \\ &= C_2(i,\text{Nil},i-1,l_1,i+1,l_2) \\ &= p_i(\text{Nil}) \end{aligned}$$

$$C_2(i,k,i-1,\text{Nil},i+1,l_2) = C_1(i,k,i+1,l_2)$$

$$C_2(i,k,i-1,l_1,i+1,\text{Nil}) = C_1(i,k,i-1,l_1).$$

Strategies That Lead to Faster Computation:

1) We set a probability value $p_i(k)$ to zero if it is less than a specified percentage (normally 5 percent) of the largest component of $\vec{p}_i$ at a given iteration. When some of the components of $\vec{p}_i$ become zero, we do not compute the gradients and compatibilities for them.

2) We set a probability vector $\vec{p}_i$ to the unit vector if any of the components of $\vec{p}_i$ becomes greater than a certain threshold (normally 0.9). The compatibilities and gradients are not subsequently computed for this unit.

3) We compute compatibility and gradient for a limited number of most likely assignments of neighbors (normally 1 or 2) for a given unit.

Control of the Stochastic Labeling Process: Two critical questions related to any relaxation process in general [11] are: 1) how do we evaluate the progress of the labeling process? and 2) how many iterations do we need and how do we stop? The answer to the first question is provided by the formulation of our technique. It guarantees that consistency will increase and ambiguity will decrease as the process progresses. This is not true for many relaxation schemes which often converge to results which are quite poor although the first few iterations provide significant improvement. To answer the second question, the number of iterations at the first and second stages are determined such that all the units are firmly assigned. The process then stops.

Examples: In using the shape matching algorithm in principle it does not matter whether we match an object with the model or vice versa. Generally, it may be preferred to label the segments of the object or model, whichever has the smaller number of segments, because in that case the labeling of a smaller number of units is required. We show the matching results in the form of a table. Only the label and the probability of the most likely assignment are shown.

Example 1: Fig. 3 shows a model and an object. The perimeter is shown as dotted points. Note that the upper portion of the model is a noisy version of the object so their polygonal approximations are different. This makes the matching problem somewhat more complicated than in [12] where Davis introduces the noise after the polygonal approximation so that the number of segments remains the same before and after the introduction of noise. Table I shows the results of labeling. Only the interior angle is used in the initial probability assignment. The results of labeling are good. Label 11 is the Nil class. Table II shows the results when only 6 iterations of the first stage are used. Now the label of unit 4 is wrong. This illustrates the need for the second stage which corrects the mistakes of the first stage. Subsequent examples support this fact.

Example 2: Fig. 4 shows two models and an object. Model 1 and Model 2 occlude each other to form an apparent object shown in Fig. 4(c). Note that there is a change of scale for the Model 1 in the apparent object. Tables III and IV show the results of labeling for Model 1 and Model 2. Note that all the assignments of Model 1 and Model 2 are correct except the assignment of unit 1 for Model 2. Although the probability of assigning label 19 to unit 1 increases, label 1 is finally assigned to this unit. Label 19 is the Nil class. This example is also described in Section III to illustrate the occlusion algorithm. There it will be seen that unit 1 of Model 2 is not assigned to label 1.

Example 3: Fig. 5 shows three regions consisting of the golden gate park obtained by using the recursive region splitting method [13] of segmentation applied to two color images of San Francisco. Regions in Fig. 5(a) and (b) are obtained from the same image with slightly different parameters in the segmentation scheme. Fig. 5(c) shows the region obtained from the image taken at a different time and rotated with respect to the other image. Since the region in Fig. 5(c) did not change with slightly different parameters, we consider it as the model and regions in Fig. 5(a) and (b) as objects. In order to reduce the computational complexity we reduce the regions shown in Fig. 5 by a factor of 14. The polygonal approximations of the reduced regions are shown in Fig. 6. We want to match the shape of objects [Fig. 6(a) and (b)] against the model [Fig. 6(c)], so object segments are units in this example. These shapes appear to be very different from each other. The left side of the golden gate park in Fig. 5(a) and (b) did not close, whereas in Fig. 5(c) it is closed. This is typical of shape matching complexity when we deal with real images. There is a clue of similarity of shapes in Figs. 5 or 6. Segments 13 and 17 of object 1 match with segments 7 and 14 or 16 of the model respectively. Similarly segments 10 and 15 or 16 of object 2 match with the segments 7 and 14 or 16 of the model. Results of shape matching are shown in Tables V and VI. Most of the assignments are very reasonable and correct although a few are incorrect. Label 30 is the Nil class.

From the results of labeling, the relative rotation between

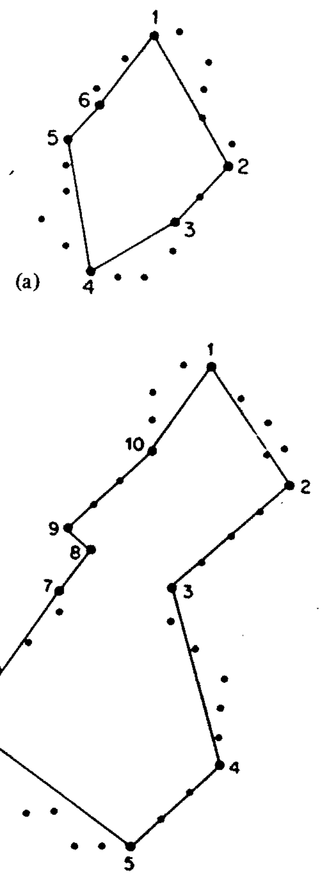


Fig. 3. (a) Model, perimeter = 21, number of segments = 6. (b) Object, perimeter = 38, number of segments = 10.

TABLE I
LABEL OF UNITS OF THE MODEL. EXAMPLE 1.

Units of the Model	Labels at different iterations				
	0	1	3	1	4
1	1(.38)	1(.48)	1(.70)	1(.81)	1(1.0)
2	5(.31)	5(.30)	2(.41)	2(.63)	2(1.0)
3	7(.53)	7(.55)	11(.61)	11(1.0)	11(1.0)
4	1(.44)	1(.42)	1(.60)	11(.61)	9(1.0)
5	4(.32)	4(.29)	11(.33)	11(.46)	10(1.0)
6	10(.75)	10(.83)	10(1.0)	10(1.0)	10(1.0)
Value of Criteria	-	.71	1.42	.93	1.14
		$J^{(1)}$		$J^{(2)}$	

Total Computation Time = 18.27 seconds

TABLE II
LABEL OF UNITS OF THE MODEL. EXAMPLE 1.

Units of the model	Labels at different iterations				
	0	1	3	5	6
1	1(.38)	1(.48)	1(.70)	1(1.0)	1(1.0)
2	5(.31)	5(.30)	2(.41)	2(1.0)	2(1.0)
3	7(.53)	7(.55)	11(.61)	11(1.0)	11(1.0)
4	1(.44)	1(.42)	1(.60)	1(1.0)	1(1.0)
5	4(.32)	4(.29)	11(.33)	9(.75)	9(1.0)
6	10(.75)	10(.83)	10(1.0)	10(1.0)	10(1.0)
Value of Criterion	-	.71	1.42	1.70	2.03

$J^{(1)}$

Total Computation Time = 4.24 seconds

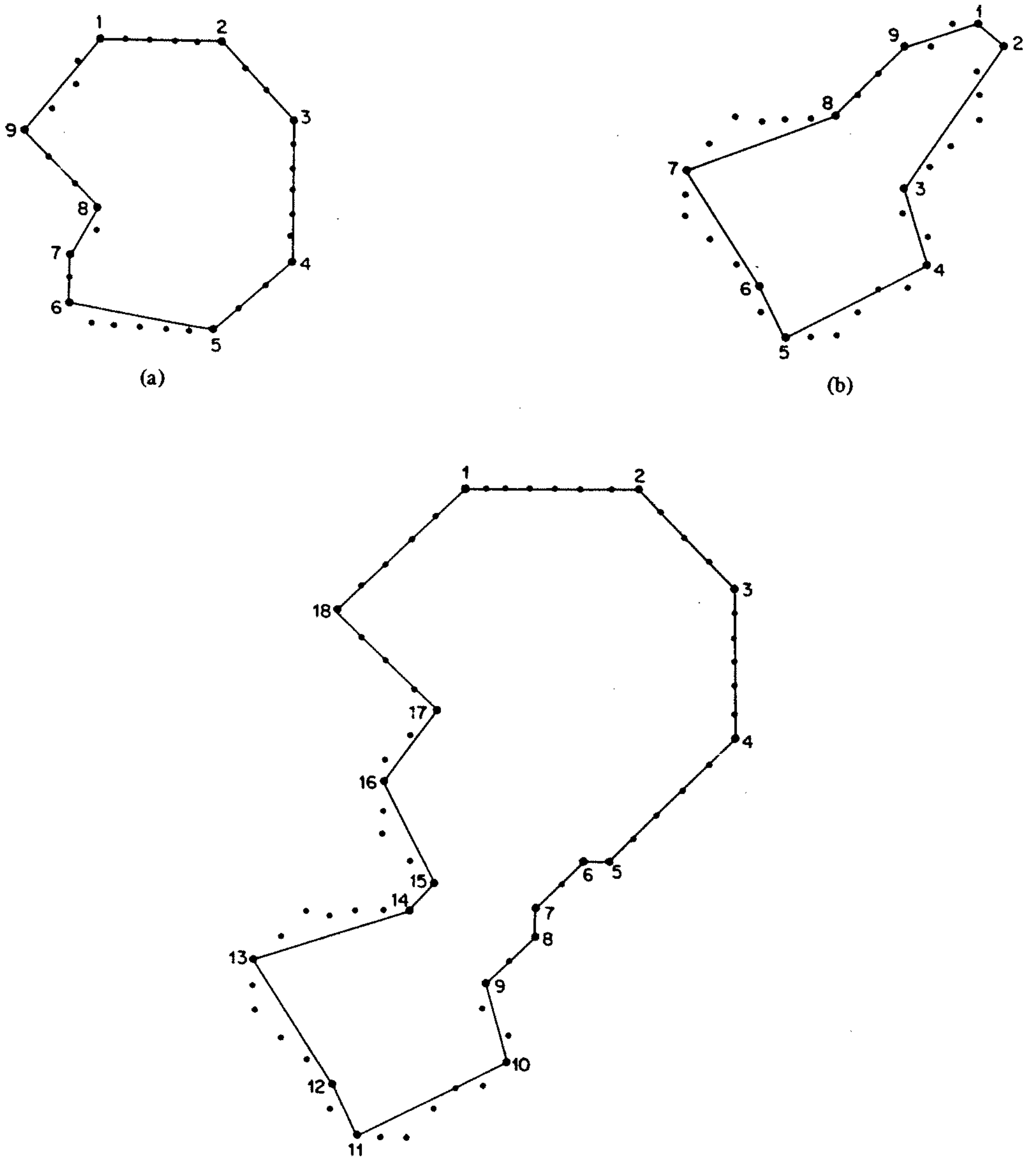


Fig. 4. (a) Model 1, perimeter = 34, number of segments = 0. (b) Model 2, perimeter = 35, number of segments = 9. (c) Object, perimeter = 67, number of segments = 18.

TABLE III
LABEL OF UNITS OF THE MODEL 1. EXAMPLE 2.

Units of the Model 1	Labels at different iterations 0	1	3	1	3	6
1	1(.20)	1(.33)	1(.50)	1(.53)	1(1.0)	1(1.0)
2	2(.51)	2(.62)	2(1.0)	2(1.0)	2(1.0)	2(1.0)
3	3(.63)	3(.74)	3(1.0)	3(1.0)	3(1.0)	3(1.0)
4	4(.28)	4(.43)	4(.71)	4(1.0)	4(1.0)	4(1.0)
5	19(.15)	19(.24)	19(.30)	19(.33)	19(.37)	5(1.0)
6	19(.15)	19(.21)	19(.27)	19(.31)	19(.45)	19(1.0)
7	19(.15)	19(.26)	16(.39)	16(.51)	16(.74)	16(1.0)
8	17(.30)	17(.43)	17(1.0)	17(1.0)	17(1.0)	17(1.0)
9	18(.29)	18(.46)	18(.68)	18(1.0)	18(1.0)	18(1.0)
Value of Criteria	-	1.03	1.56	1.33	1.44	1.44
		$J^{(1)}$			$J^{(2)}$	

Total Computation Time = 52.79 seconds

TABLE IV
LABEL OF UNITS OF THE MODEL 2. EXAMPLE 2.

Units of the Model 2	Labels at different iterations 0	1	3	1	3	5
1	19(.15)	19(.26)	19(.36)	19(.43)	19(.65)	1(1.0)
2	10(.15)	19(.20)	19(.24)	19(.30)	19(.45)	19(1.0)
3	9(.24)	9(.56)	9(1.0)	9(1.0)	9(1.0)	9(1.0)
4	10(.67)	10(1.0)	10(1.0)	10(1.0)	10(1.0)	10(1.0)
5	11(.66)	11(1.0)	11(1.0)	11(1.0)	11(1.0)	11(1.0)
6	12(.65)	12(1.0)	12(1.0)	12(1.0)	12(1.0)	12(1.0)
7	13(.66)	13(1.0)	13(1.0)	13(1.0)	13(1.0)	13(1.0)
8	19(.15)	19(.22)	19(.27)	19(.32)	14(.42)	14(1.0)
9	1(.16)	1(.23)	1(.33)	1(.36)	19(.40)	19(1.0)
Value of Criteria	-	1.81	2.89	2.74	2.36	3.19
		$J^{(1)}$			$J^{(2)}$	

Total Computation Time = 42.9 seconds

the model and the object can be computed. This is done by using the following formula:

average relative rotation

$$= \frac{\left(\sum_{\substack{i=1 \\ \text{label} \notin \text{Nil}}}^{N} \left| \begin{array}{c}\text{slope of the } i\text{th unit} - \text{slope of the} \\ \text{assigned label to the } i\text{th unit}\end{array} \right| \right)}{\text{number of units not assigned to Nil class}} \tag{26}$$

where N is the number of units. The idea behind this formula to compute the rotation is that although some labels may be wrong, it is expected that the slope of the matching segments will not be widely different. In practice we consider two cases when using this formula.

Case 1: If it is given that the relative rotation is small ($<90°$), we subtract any term greater than $180°$ in the summation of (26), from $360°$. So all the terms contributing to the sum in (26) will be less than $180°$.

Case 2: If the relative rotation is greater than $90°$, we do not subtract any term in the summation of (26) from $360°$ as in case 1, but we neglect the terms contributing less than $30°$

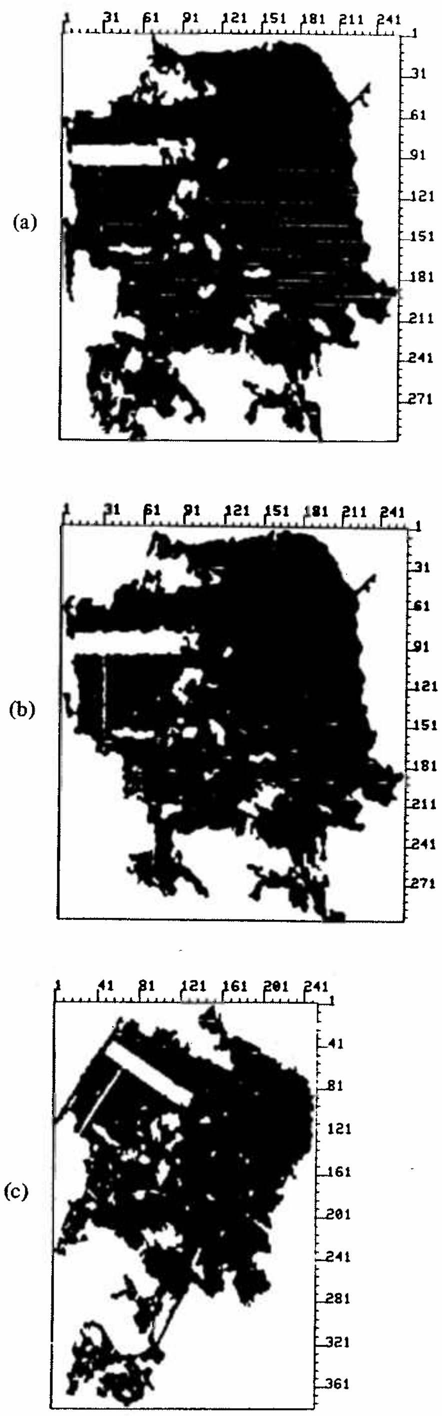


Fig. 5. Regions obtained using a recursive region splitting technique of segmentation. Regions in (a) and (b) are obtained from the same aerial image of San Francisco with slightly different parameters in the segmentation scheme. Region in (c) is obtained from another image, taken at a different time, by using the same parameters used to obtain (a) and (b). Regions shown are at different scales. (a) Size = 299 rows by 258 columns. (b) Size = 297 rows by 261 columns. (c) Size = 381 rows by 253 columns.

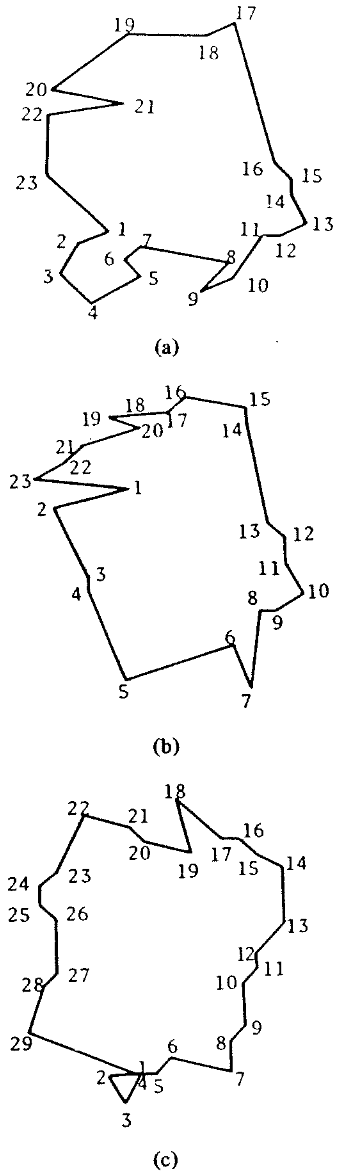


Fig. 6. Polygonal approximation of the regions shown in Fig. 5 after reducing them by 14 times. (a), (b), and (c) correspond to Fig. 5(a), (b), and (c), respectively. (a) Object 1, number of segments = 23. (b) Object 2, number of segments = 23. (c) Model, number of segments = 29.

and by the same amount the number of units is reduced in the denominator of (26).

Tables VII and VIII show calculations of the relative rotation for object 1 and object 2. The relative rotation between object 1 and the model and between object 2 and the model are found to be 36.1° and 35.5°, respectively. The actual rotation is 35°.

Example 4: Fig. 7(a) shows the top view of a piece of car shock absorber and 7(b) the superposition of two such pieces, the one below being the one of Fig. 7(a). The polygonal approximation is shown in Fig. 7(c) and (d). From a practical standpoint, it is important to identify in the shape of Fig. 7(d) (the observation) the visible part of the shape of Fig. 7(c) (the model). In this example $N = L = 28$. Table IX shows the results. Here the units 8, 9, 10, 11 and 26, 27, 28, 1 are

TABLE V
Label of Segments of the Object 1. Example 3.

Segments of the Object 1	Labels at different iterations					
	0	1	5	2	4	7
1	30(.08)	27(.20)	27(1.0)	27(1.0)	27(1.0)	27(1.0)
2	28(.16)	28(.22)	28(.78)	28(1.0)	28(1.0)	28(1.0)
3	25(.08)	29(.11)	29(.25)	28(.44)	28(1.0)	28(1.0)
4	5(.08)	30(.10)	3(.33)	3(.39)	3(1.0)	3(1.0)
5	30(.08)	5(.11)	5(.23)	5(.32)	5(.51)	5(1.0)
6	30(.08)	4(.14)	4(.39)	5(.56)	5(1.0)	5(1.0)
7	6(.10)	6(.11)	6(.77)	6(1.0)	6(1.0)	6(1.0)
8	19(.10)	19(.12)	1(.25)	7(.44)	7(.67)	7(1.0)
9	30(.08)	3(.13)	3(.30)	5(.31)	5(1.0)	5(1.0)
10	5(.09)	3(.20)	3(1.0)	3(1.0)	3(1.0)	3(1.0)
11	4(.21)	4(.35)	4(1.0)	4(1.0)	4(1.0)	4(1.0)
12	30(.08)	5(.15)	5(.47)	5(.79)	5(1.0)	5(1.0)
13	7(.11)	7(.20)	7(1.0)	7(1.0)	7(1.0)	7(1.0)
14	30(.08)	8(.13)	8(.58)	8(1.0)	8(1.0)	8(1.0)
15	30(.08)	30(.10)	9(.16)	9(.36)	9(.50)	8(1.0)
16	17(.08)	17(.14)	17(.63)	13(1.0)	13(1.0)	13(1.0)
17	30(.08)	30(.11)	30(.17)	18(.19)	16(.28)	16(1.0)
18	30(.08)	30(.12)	20(.35)	20(.71)	20(1.0)	20(1.0)
19	30(.08)	30(.12)	30(.17)	23(.26)	23(.44)	23(1.0)
20	18(.10)	2(.12)	18(.40)	30(.63)	30(1.0)	30(1.0)
21	1(.11)	21(.12)	21(.44)	19(1.0)	19(1.0)	19(1.0)
22	22(.14)	22(.17)	22(.53)	22(.77)	22(1.0)	22(1.0)
23	30(.08)	30(.10)	26(.24)	26(.33)	26(.53)	26(1.0)
Value of Criteria	-	.898	2.236	1.604	1.688	1.718
		$J^{(1)}$			$J^{(2)}$	

Total Computation Time = 243.75 seconds

TABLE VI
Label of Segments of the Object 2. Example 3.

Segments of the Object 2	Labels at different iterations					
	0	1	6	1	4	6
1	1(.15)	1(.20)	1(.64)	1(1.0)	1(1.0)	1(1.0)
2	2(.11)	2(.19)	2(.49)	2(.57)	2(1.0)	2(1.0)
3	30(.08)	27(.12)	27(.36)	27(.34)	3(.40)	3(1.0)
4	28(.09)	28(.15)	28(.68)	28(.79)	28(1.0)	28(1.0)
5	30(.08)	30(.12)	29(.31)	29(.35)	29(1.0)	29(1.0)
6	30(.08)	2(.14)	2(.66)	2(1.0)	2(1.0)	2(1.0)
7	3(.12)	3(.28)	3(1.0)	3(1.0)	3(1.0)	3(1.0)
8	4(.12)	4(.20)	4(1.0)	4(1.0)	4(1.0)	4(1.0)
9	30(.08)	5(.13)	5(.40)	5(.66)	5(1.0)	5(1.0)
10	7(.11)	7(.20)	7(1.0)	7(1.0)	7(1.0)	7(1.0)
11	30(.08)	8(.11)	8(.38)	8(.42)	8(1.0)	8(1.0)
12	11(.08)	11(.12)	9(.49)	9(.65)	9(1.0)	9(1.0)
13	30(.08)	30(.10)	12(.27)	12(.30)	11(.62)	11(1.0)
14	30(.08)	13(.15)	13(.55)	14(1.0)	14(1.0)	14(1.0)
15	14(.14)	14(.31)	14(1.0)	14(1.0)	14(1.0)	14(1.0)
16	30(.08)	16(.16)	16(.61)	16(1.0)	16(1.0)	16(1.0)
17	30(.08)	30(.13)	17(.33)	19(.38)	19(1.0)	19(1.0)
18	30(.08)	15(.10)	19(.20)	19(.26)	19(.45)	20(1.0)
19	18(.12)	18(.13)	18(1.0)	18(1.0)	18(1.0)	18(1.0)
20	1(.09)	19(.18)	19(.80)	19(1.0)	19(1.0)	19(1.0)
21	30(.08)	23(.13)	22(.41)	20(.63)	20(1.0)	20(1.0)
22	23(.08)	23(.11)	23(.44)	23(.49)	22(.61)	22(1.0)
23	18(.11)	29(.11)	29(.35)	29(.72)	29(1.0)	29(1.0)
Value of Criteria	-	0.917	1.985	1.606	1.658	1.672
		$J^{(1)}$			$J^{(2)}$	

Total Computation Time = 219.20 seconds

assigned to the same labels because of the great similarity of their local structure. The labeling could correctly identify the visible part of the model within the observation. The relative rotation is found to be 6.3°. The actual rotation is 5°.

III. Shape Matching of 2-D Occluded Objects

Matching of occluded objects is one of the prime capabilities of any shape analysis system. We view the occlusion problem

TABLE VII
COMPUTATION OF THE RELATIVE ROTATION BETWEEN THE OBJECT 1 AND THE MODEL. EXAMPLE 3.

Units of the object 1	Slope of the units of object 1 in degrees a	Label of the units of object 1	Slope of the segments of the labels in degrees b	|a-b| in degrees
1	207	27	225	18
2	243	28	252	9
3	315	28	252	63
4	33.7	3	63.4	29.7
5	135	5	45	90
6	45	5	45	0.0
7	351	6	346	5
8	225	7	90	180
9	26.6	5	45	18.4
10	56.3	3	63.4	7.1
11	0.0	4	0.0	0.0
12	26.6	5	45	18.4
13	117	7	90	27
14	90	8	45	45
15	135	8	45	90
16	107	13	90	17
17	207	16	180	27
18	180	20	135	45
19	219	23	225	6
20	349	30	-	-
21	191	19	162	29
22	270	22	243	27
23	315	26	270	45

Relative Rotation = 796.6÷22 = 36.1° Sum = 796.6

TABLE VIII
COMPUTATION OF THE RELATIVE ROTATION BETWEEN THE OBJECT 2 AND THE MODEL. EXAMPLE 3.

Units of the object 2	Slope of the units of object 2 in degrees a	Label of the units of object 2	Slope of the segments of the labels in degrees b	|a-b| in degrees
1	191	1	180	11
2	292	2	297	5
3	270	3	63.4	206.6+
4	288	28	252	36
5	15.9	29	337	321.1+
6	288	2	297	9
7	78.7	3	63.4	15.3
8	0.0	4	0.0	0
9	26.6	5	45	18.4
10	117	7	90	27
11	90	8	45	45
12	135	9	90	45
13	98.1	11	90	8
14	90	14	153	63
15	166	14	153	13
16	225	16	180	45
17	180	19	162	18
18	180	20	135	45
19	333	18	284	49
20	194	19	162	32
21	225	20	135	90
22	207	22	243	36
23	351	29	337	14

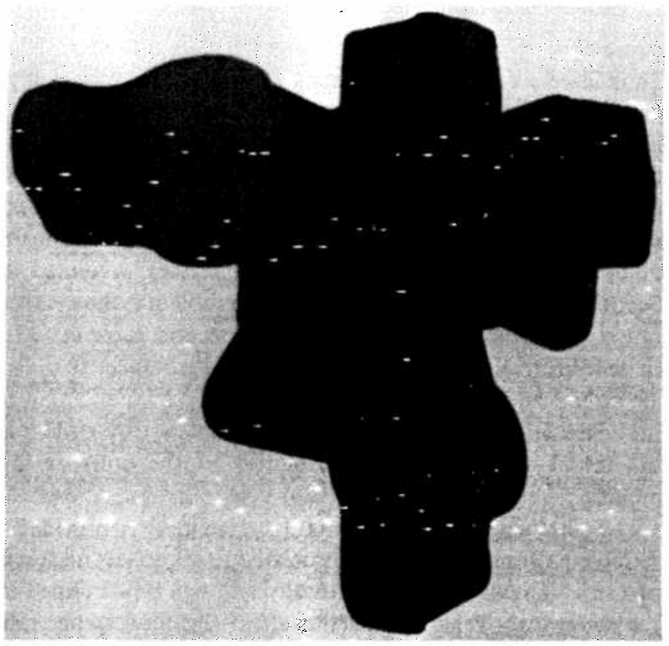
+The contribution of these terms in the sum will be 360-value Sum = 8 17.1
Relative rotation = 817.1:23 = 35.5°

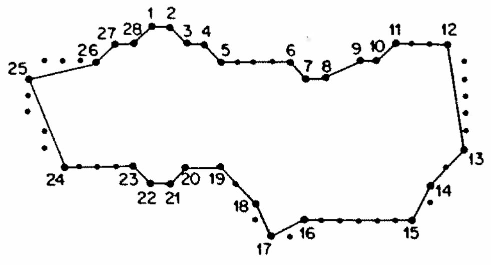
in 2-D basically as a boundary matching problem [14], [15]. However, compared to the previous studies, the framework presented here provides a firm mathematical basis for the solution of the occlusion problem. The occlusion problem treated as a segment matching problem involves matching the segments of two or more actual objects with the apparent object, which is formed by the occlusion of these objects. Some segments of the actual objects may not match with any of the segments of the apparent object. Also the matching algorithm should not assign the same segment of the apparent object to segments of different actual objects. In this section we extend the algorithm presented in Section II such that several hierarchical processes are executed in parallel for every object participating in the occlusion and are coordinated in such a way that the same segment of the apparent object is not matched to the segments of different actual objects. This is done by combining the gradient projection method with the penalty function approach [16]. In the following we formulate this problem as an optimization problem, discuss the occlusion algorithm and present several examples in which 2 or 3 objects partially occlude.

Problem Formulation: Consider a general case in which $M(\geqslant 2)$ actual objects, called models $(X_1, \cdots, X_M)$ occlude one another to form a single apparent object called the object. Let a model X_m be represented by $X_m = (T_1, T_2, \cdots, T_{N_m})$, where N_m is the number of segments in the polygonal path representation of the model X_m. Similarly, let $O = (O_1, O_2, \cdots, O_{L-1})$ be the polygonal path representation of the object. The object has $L - 1$ segments. We want to match the segments of the models against the segments of the object such that the following two conditions are satisfied.

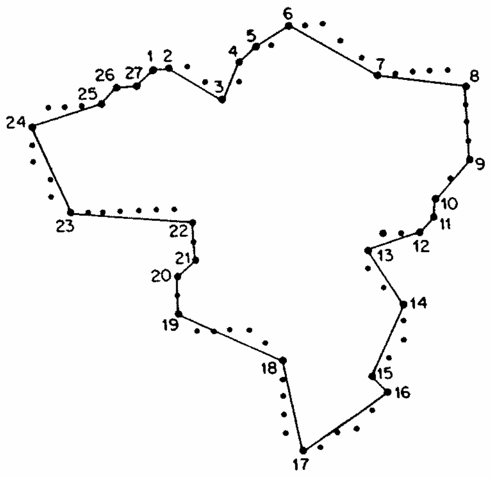
1) None of the segments of the different models are assigned

(a)


(b)



(c)



(d)

Fig. 7. (a) An automobile piece. (b) Superposition of two such pieces. (c) Model, number of segments = 28. (d) Object, number of segments = 27.

TABLE IX
LABEL OF UNITS OF THE MODEL. EXAMPLE 4.

Units of the Model	Labels at different iterations			
	0	1	3	2
1	28(.25)	1(.40)	1(1.0)	1(1.0)
2	28(.25)	28(.54)	28(.72)	28(1.0)
3	26(.25)	23(.49)	28(.68)	23(1.0)
4	28(.25)	28(.54)	28(.67)	28(1.0)
5	28(.25)	28(.49)	28(.73)	28(1.0)
6	28(.25)	28(.54)	28(.67)	28(1.0)
7	28(.25)	28(.57)	28(.10)	28(1.0)
8	28(.25)	28(.51)	28(.52)	25(1.0)
9	28(.25)	28(.37)	26(.60)	26(1.0)
10	27(.46)	27(.65)	27(1.0)	27(1.0)
11	28(.25)	28(.37)	28(.53)	1(1.0)
12	28(.25)	28(.51)	28(.59)	8(1.0)
13	28(.25)	28(.43)	28(.62)	28(1.0)
14	28(.25)	28(.65)	28(1.0)	28(1.0)
15	28(.25)	28(.51)	28(1.0)	23(1.0)
16	28(.25)	28(.66)	28(1.0)	28(1.0)
17	28(.25)	28(.55)	26(.76)	28(1.0)
18	28(.25)	28(.77)	28(1.0)	28(1.0)
19	28(.25)	28(.62)	28(1.0)	28(1.0)
20	28(.25)	28(.56)	28(1.0)	28(1.0)
21	28(.25)	28(.59)	28(1.0)	28(1.0)
22	28(.25)	28(.59)	28(1.0)	28(1.0)
23	28(.25)	28(.58)	28(.73)	22(1.0)
24	28(.25)	23(.52)	23(1.0)	23(1.0)
25	24(.48)	24(.67)	24(1.0)	24(1.0)
26	25(.44)	25(.62)	25(1.0)	25(1.0)
27	26(.46)	26(.67)	26(1.0)	26(1.0)
28	27(.46)	27(.62)	27(1.0)	27(1.0)
Value of Criterion	-	3.14	12.67	14.23
		$J^{(1)}$		$J^{(2)}$

Total Computation Time = 74.56 seconds

to the same segment of the object. This is called the occlusion condition. It is necessary for the labeling to be unambiguous.

2) Those segments of the models which do not match to any of the segments of the object are assigned to the Nil class.

We are thus trying to identify parts of the models within the object. We designate the object segments as classes, and the model segments as units. As discussed in the last section the global criterion that measures the consistency and ambiguity of the labeling over the set of units of a model X_m is given by

$$J_m^{(n)} = \sum_{i=1}^{Nm} \vec{p}_{im} \cdot \vec{q}_{im}^{(n)}, \quad n = 1, 2 \tag{27}$$

where n denotes the first or second stage of the hierarchy. Let $\vec{v}_m$ be the vector of $R^P = R^L \times \cdots \times R^L$ $(P = N_m L)$ equal to $(\vec{p}_{1m}, \vec{p}_{2m}, \cdots, \vec{p}_{N_m})$. Then (27) can be written as

$$J_m^{(n)} = \sum_{i=1}^{N_m} J_{im}^{(n)}(\vec{v}_m) \tag{28}$$

where

$$J_{im}^{(n)}(\vec{v}_m) = \vec{p}_{im} \cdot \vec{q}_{im}^{(n)}, \quad n = 1, 2.$$

Now the total criterion of consistency and ambiguity for all the M models is given by

$$F(\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_M) = \sum_{m=1}^{M} \sum_{i=1}^{N_m} J_{im}^{(n)}(\vec{v}_m), \quad n = 1, 2. \tag{29}$$

The occlusion condition can be written as

$$G(\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_M) = \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} g(\vec{s}_i, \vec{s}_j) = 0 \tag{30}$$

where $\vec{s}_l$ is obtained from $\vec{v}_l$ with the elements corresponding to the Nil class set equal to zero for all the units of the model X_l and $g(\vec{s}_i, \vec{s}_j)$ is given by

$$g(\vec{s}_i, \vec{s}_j) = \left(\sum_{k=1}^{N_i} \vec{s}_{ki}\right) \cdot \left(\sum_{l=1}^{N_j} \vec{s}_{lj}\right)$$

$$\text{with } \vec{s}_{ki} = [p_{ki}(1), \cdots, p_{ki}(L-1), 0]^T.$$

What this condition essentially means is that if a unit k of a model X_i matches the class O_l (where $l \neq L$), then the sum of the inner product of the probability vector of this unit $\vec{p}_{ki}$ with the probability vectors of all the units of all the other models should be zero. The Nil class components have been excluded by using $\vec{s}_l$ rather than $\vec{v}_l$ because one or more segments of different models may match to the Nil class. Now the occlusion problem viewed as a segment matching problem can be stated as follows.

Problem Statement (A): Given an initial labeling $\vec{v}_1^{(0)}$, $\vec{v}_2^{(0)}, \cdots, \vec{v}_M^{(0)}$ for the set of M models $(X_1, X_2, \cdots, X_M)$, find the labeling $\vec{u}_1, \vec{u}_2, \cdots, \vec{u}_M$ that corresponds to the local maximum of the criterion (29) which is closest to $\vec{v}_1^{(0)}, \vec{v}_2^{(0)}, \cdots, \vec{v}_M^{(0)}$ subject to the following constraints.

a) If $\vec{u}_m = (\vec{p}_{1m}, \vec{p}_{2m}, \cdots, \vec{p}_{N_m m})$ then $\vec{p}_{lm}$ is a probability vector for $l = 1, 2, \cdots, N_m$ and $m = 1, 2, \cdots, M$. For a particular unit y of the model X_m, this means that if

$$\vec{p}_{ym} = [p_{ym}(1), p_{ym}(2), \cdots, p_{ym}(L)]^T,$$

then

$$\sum_{l=1}^{L} p_{ym}(l) = 1 \quad \text{and } p_{ym}(l) \geq 0, \qquad \text{for } l = 1, \cdots, L.$$

b) $G(\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_M)$ as defined by (30) is equal to zero.

Note that criterion (29) is nonlinear. Constraint a) involves linear equality and nonnegativity restriction, and constraint b) is nonlinear. In order to solve this optimization problem we use the penalty function concept [16] and extend the hierarchical shape matching technique of Section II.

Occlusion Algorithm: To solve problem (A) using the penalty function approach, we define the penalized objective function as

$$\psi_c(\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_M) = F(\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_M) + \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} d_{ij}\, \phi_{ij}[g(\vec{s}_i, \vec{s}_j)] \qquad (31)$$

where ϕ_{ij} is a penalty function and $\{d_{ij}\}$ are penalty constants. Since the constraint b) given by (30) is an equality constraint, the penalty function is taken as the simple quadratic loss function, i.e.,

$$\phi_{ij}(a) \triangleq -a^2. \qquad (32)$$

Now problem (A) becomes equivalent to that of maximizing (31) subject to the constraints a). It can be solved by using the gradient projection method applied to the linear constraints as it has been used in the last section. The maximization of (31) subject to the constraints a) is equivalent to maximizing

$$\begin{cases} \max\limits_{\vec{v}_1} \; F(\vec{v}_1) + S(\vec{v}_1, \cdots, \vec{v}_M) \\ \max\limits_{\vec{v}_2} \; F(\vec{v}_2) + S(\vec{v}_1, \cdots, \vec{v}_M) \\ \quad \vdots \\ \max\limits_{\vec{v}_m} \; F(\vec{v}_M) + S(\vec{v}_1, \cdots, \vec{v}_M) \end{cases} \qquad (33)$$

where $S(\vec{v}_1, \cdots, \vec{v}_M)$ corresponds to the second term of (31). Thus in effect there is a hierarchical process for every model participating in the occlusion. These processes are executed in parallel and coordinated in such a way that the occlusion condition is satisfied. The algorithm has been implemented in a serial fashion on the computer, first we maximize with respect to $\vec{v}_1$, then with respect to $\vec{v}_2$ and so on. The main modification of the shape matching algorithm presented in Section II with respect to the occlusion problem is the computation of the gradient relating to the second term in (33). In general, to solve (33) by maximizing with respect to $\vec{v}_i$ the algorithm can be stated as follows.

1) Pick an initial estimate of $(\vec{v}_1^{(0)}, \vec{v}_2^{(0)}, \cdots, \vec{v}_M^{(0)})$. This is the initial assignment of probabilities to the units of the models.

2) Pick the penalty constants $\{d_{ij}\}$ so that it provides a suitable balance between the associated first and second terms of (33). This is done automatically and will be described in the following. Penalty constants affect the convergence rate of the algorithm.

3) Determine the maximum $\vec{v}_m^{(n+1)}$ $(m = 1, 2, \cdots, M)$ of the penalized objective function (33) subject to the constraints a) by using the value at the present iteration $\vec{v}_m^{(n)}$ and the gradient projection method.

4) Pick new penalty constants $\{d_{ij}\}$ in order to rebalance the magnitude of the penalty terms. The magnitude of the penalties should be increased to force a closer approach to the boundary; replace n by $n + 1$ and return to 3.

Under the assumption of the continuity of function F in (29) and constraints (30) inherent in (31), the sequence of maxima $\{\vec{v}_m^{(n+1)}\}$ for $m = 1, \cdots, M$ generated by the above algorithm approaches a constrained maximum of the problem defined in (A). The iteration is terminated when all the units of the models are firmly assigned. Since we are seeking only local maxima, ill-conditioning problems near the boundary do not occur [16].

Examples: In the examples presented here we have taken penalty constants associated with various terms in (31) to be the same. We determine its value at every iteration such that the penalty term (second term) of (33) is a fixed percentage (between 10-90 percent) of the first term in (33). Since the criterion increases at every iteration, the penalty constant also increases and when the occlusion condition is satisfied, the penalty constant effectively becomes infinite. In Example 4 of Section II, we identified a partial view of a model within the object. In the examples presented in the following we know *a priori* all the models which are occluding one another and we want to identify all of them based on their partial views.

TABLE X
ASSIGNMENT OF THE UNITS OF MODEL X_1 USING THE OCCLUSION ALGORITHM. EXAMPLE 5.

Units of Model X_1	Labels at different iterations							
	First Stage			Second Stage				
	0	1	3	1	4	7	10	11
1	1(.20)	1(.22)	1(.23)	1(.24)	1(.34)	1(.56)	1(1.0)	1(1.0)
2	2(.51)	2(.62)	2(.68)	2(1.0)	2(1.0)	2(1.0)	2(1.0)	2(1.0)
3	3(.63)	3(.72)	3(.75)	3(1.0)	3(1.0)	3(1.0)	3(1.0)	3(1.0)
4	4(.28)	4(.32)	4(.33)	4(.37)	4(.56)	4(1.0)	4(1.0)	4(1.0)
5	19(.15)	19(.19)	19(.20)	19(.21)	19(.23)	19(.33)	5(.43)	5(1.0)
6	19(.15)	19(.19)	19(.20)	19(.23)	19(.32)	19(.50)	15(.77)	15(1.0)
7	19(.15)	19(.18)	19(.19)	19(.20)	19(.27)	16(.47)	16(1.0)	16(1.0)
8	17(.30)	17(.34)	17(.38)	17(.46)	17(1.0)	17(1.0)	17(1.0)	17(1.0)
9	18(.29)	18(.31)	18(.33)	18(.37)	18(.58)	18(1.0)	18(1.0)	18(1.0)
First term of the objective function	-	1.037	1.252	1.095	1.243	1.352	1.516	1.323
Penalty terms of objective function	-	.9334	1.127	.9862	1.119	0	0	0
Criterion	-	.1037	.1252	.1095	.1243	1.352	1.516	1.323
Penalty Constant	-	.1009	1.071	2.758	16738.4	-	-	-

TABLE XI
ASSIGNMENT OF THE UNITS OF MODEL X_2 USING THE OCCLUSION ALGORITHM. EXAMPLE 5.

Units of Model X_2	Labels at different iterations							
	First Stage			Second Stage				
	0	1	3	1	4	7	10	11
1	19(.15)	19(.20)	19(.22)	19(.25)	19(.28)	19(.45)	19(.71)	19(1.0)
2	10(.15)	19(.17)	19(.20)	19(.21)	19(.25)	10(.39)	7(1.0)	7(1.0)
3	9(.24)	9(.29)	9(.31)	9(.34)	9(.36)	9(1.0)	9(1.0)	9(1.0)
4	10(.67)	10(1.0)	10(1.0)	10(1.0)	10(1.0)	10(1.0)	10(1.0)	10(1.0)
5	11(.66)	11(1.0)	11(1.0)	11(1.0)	11(1.0)	11(1.0)	11(1.0)	11(1.0)
6	12(.65)	12(1.0)	12(1.0)	12(1.0)	12(1.0)	12(1.0)	12(1.0)	12(1.0)
7	13(.66)	13(1.0)	13(1.0)	13(1.0)	13(1.0)	13(1.0)	13(1.0)	13(1.0)
8	19(.15)	19(.17)	19(.20)	19(.25)	19(.27)	14(.79)	14(1.0)	14(1.0)
9	1(.16)	19(.22)	19(.24)	19(.29)	19(.31)	19(.71)	19(1.0)	19(1.0)
First term of the objective function	-	1.812	2.581	2.312	2.388	1.974	2.202	2.212
Penalty term of objective function	-	1.631	2.323	2.081	2.149	0	0	0
Criterion	-	.1812	.2581	.2312	.2388	1.974	2.202	2.212
Penalty Constant	-	.1764	2.207	5.820	32158.9	-	-	-

Example 5: Fig. 4 shows two models X_1 and X_2 which occlude each other to form an apparent object. This is the figure used in Example 2. Here we use the occlusion algorithm to identify models X_1 and X_2 within the apparent object. The results of labeling the units of the models are shown in Tables X and XI. Here we have used the same parameters as the ones used to obtain Table III and Table IV. We also show values of the unpenalized objective function and penalty function terms [first and second terms of (31)], criteria and penalty constants at various iterations. Note that the criteria and penalty constants increase with the iterations. The assignment of the units of both models are correct and the conflicting labeling of the two units which occurred in Tables III and IV does not occur (unit 1 of both X_1 and X_2 are assigned to the label 1 in the Tables III and IV). The total computation time for the results shown in Tables X and XI is 235 s.

Example 6: Fig. 8 presents a synthetic example, where three models X_1, X_2, and X_3 occlude one another to form an apparent object. We want to identify each of the models within the apparent object. The problem is a kind of "jig-saw puzzle." The labeling results are shown in Tables XII-XIV. All the lables of all the units of X_1, X_2, and X_3 are correct. The total computation time for matching is 152.7 s.

Example 7: Fig. 9 shows 512 X 512, 8 bits gray scale images of industrial parts which occlude each other to form an apparent object. The images in Fig. 9(a) and (b) are reduced by a factor of 16 and the image in Fig. 9(c) by a factor of 18. The reduced images are thresholded and their polygonal approximation are shown in Fig. 9(d)-(f). Only the rotation and scale invariant features are used in the initial probability assignment. Label 25 is the Nil class. The results are shown in Table XV and XVI. Note that all the key assignments of the units are correct. The units 5, 6, 7, 8 of model X_1 are not matched to the segments 9, 10, 11, 12. The reason for this is the presence of ambiguity between segments 5 to 17 of Fig. 9(d) and 8 to 16 of Fig. 9(f); the number of segments is different as a result of change in scale. The total computation time for this example is 530 s.

Example 8: This example is provided in order to critically evaluate the occlusion algorithm when the segmentation is difficult and the polygonal approximation is crude. Fig. 10 shows two 128 X 128, 8 bit images of cells. These images have been taken 15 minutes apart. The background in these images consists of human skin cancer cells and the small circular shaped objects are human lymphocytes and red blood cells. One cancer cell in the image of Fig. 10(a) is undergoing

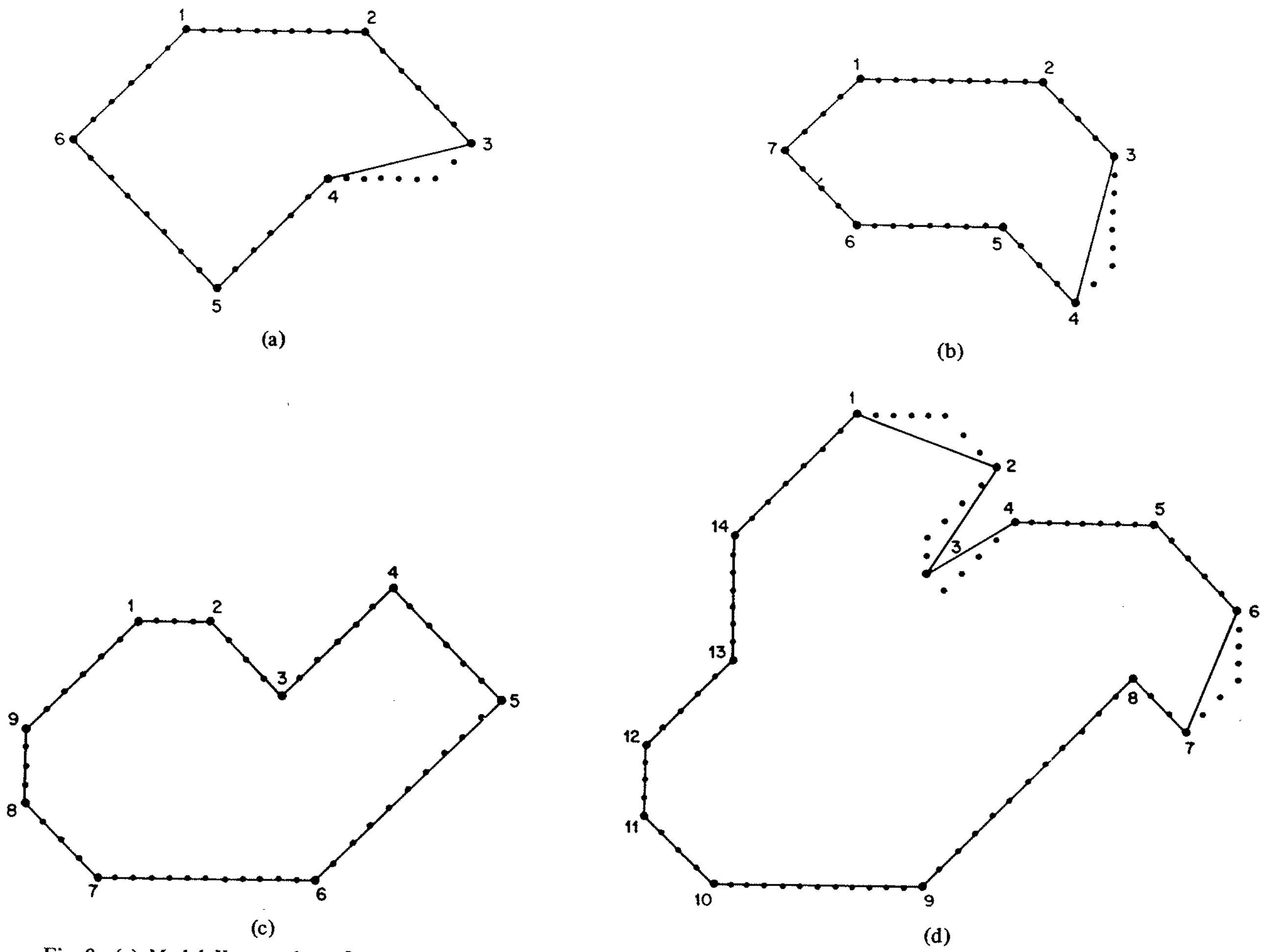


Fig. 8. (a) Model X_1, number of segments = 6. (b) Model X_2, number of segments = 7. (c) Model X_3, number of segments = 9. (d) Apparent object, number of segments = 14.

TABLE XII
RESULTS OF LABELING FOR THE MODEL X_1 USING THE OCCLUSION ALGORITHM. EXAMPLE 6.

Units of Model X_1	Labels at different iterations					
	First Stage			Second Stage		
	0	1	3	1	4	6
1	4(.17)	15(.23)	15(.34)	15(.37)	15(.68)	15(1.0)
2	1(.20)	1(.24)	1(.30)	1(.34)	1(.71)	1(1.0)
3	15(.15)	15(.33)	15(.47)	15(.54)	15(1.0)	15(1.0)
4	15(.15)	15(.30)	15(.36)	15(.41)	15(1.0)	15(1.0)
5	15(.15)	15(.23)	15(.27)	15(.33)	15(.60)	15(1.0)
6	15(.15)	15(.29)	15(.35)	15(.40)	15(1.0)	15(1.0)
First term of the objective function	-	.5183	.9785	1.074	1.6071	2.437
Penalty term of the objective function	-	.1554	.2935	.3223	.4821	0
Criterion	-	.3628	.6849	.7522	1.125	2.437
Penalty Constant	-	.00624	.0427	.0758	1.131	-

mitosis. In the image of Fig. 10(b), the cell of Fig. 10(a) undergoing mitosis (parent cell) has been divided into two cells (daughter cells). Note that significant changes in shape have taken place. We use a gradient relaxation technique of segmentation [17] to obtain the cell boundaries. Fig. 10(c)-(e) show the polygonal approximation. We want to match the daughter cells of Fig. 10(c) and (d) with the parent cell of Fig. 10(e). The results of labeling are shown in Table XVII and XVIII. The assignment of units 18 and 19 of model X_1 are correct, but the label 23 of unit 20 is wrong. The unit 13 should have been labeled as 23, but a careful examination of this figure shows that the local structure of unit 20 matches 23 better than to 13. Other units are assigned to the Nil class. All the assignments of the units of model X_2 are correct

TABLE XIII
RESULTS OF LABELING FOR THE MODEL X_2 USING THE OCCLUSION ALGORITHM. EXAMPLE 6.

Units of Model X_2	Labels at different iterations					
	First Stage			Second Stage		
	0	1	3	1	4	6
1	4(.17)	4(.33)	4(.42)	4(.48)	4(1.0)	4(1.0)
2	5(.31)	5(.53)	5(.73)	5(1.0)	5(1.0)	5(1.0)
3	6(.40)	6(.62)	6(.77)	6(1.0)	6(1.0)	6(1.0)
4	15(.15)	7(.40)	7(.55)	7(.56)	15(1.0)	15(1.0)
5	13(.20)	13(.30)	13(.38)	13(.39)	15(.68)	15(1.0)
6	15(.15)	15(.50)	15(.71)	15(1.0)	15(1.0)	15(1.0)
7	15(.15)	15(.20)	15(.33)	15(.44)	15(.61)	3(1.0)
First term of the objective function	-	.8197	1.5709	1.3875	2.7729	3.451
Penalty term of the objective function	-	.2459	.4712	.4162	.8318	0
Criterion	-	.5738	1.0996	.9712	1.9410	3.451
Penalty Constant	-	.009875	.0685	.0979	1.953	-

TABLE XIV
RESULTS OF LABELING FOR THE MODEL X_3 USING THE OCCLUSION ALGORITHM. EXAMPLE 6.

Units of Model X_3	Labels at different iterations					
	First Stage			Second Stage		
	0	1	3	1	4	6
1	4(.16)	15(.26)	15(.32)	15(.39)	15(.52)	15 (1.0)
2	5(.21)	15(.33)	15(.46)	15(.52)	15(1.0)	15 (1.0)
3	13(.17)	15(.33)	15(.49)	15(.53)	15(.59)	13 (1.0)
4	1(.19)	15(.35)	15(.44)	15(.47)	15(.56)	15 (1.0)
5	2(.22)	15(.28)	15(.37)	15(.41)	15(.61)	15 (1.0)
6	9(.66)	9(.77)	9(1.0)	9(1.0)	9(1.0)	9 (1.0)
7	10(.63)	10(.73)	10(1.0)	10(1.0)	10(1.0)	10 (1.0)
8	11(.63)	11(.76)	11(1.0)	11(1.0)	11(1.0)	11 (1.0)
9	14(.16)	15(.20)	15(.25)	15(.29)	12(.40)	12 (1.0)
First term of the objective function	-	1.721	3.031	3.079	4.100	3.692
Penalty term of the objective function	-	.5165	.9095	.9237	1.230	0
Criterion	-	1.205	2.122	2.155	2.870	3.692
Penalty Constant	-	.0207	.1323	.2172	2.887	-

except the assignment of unit 9 which is matched to the label 21. This is again because of the close resemblance of the local structure. The total computation time for this example is 1100 s.

IV. CONCLUSIONS

The success or failure of the hierarchical stochastic labeling technique can be measured on the basis of two facts: 1) the final labeling should be as unambiguous as possible and 2) it should be consistent with any *a priori* knowledge that we may have about the set of possible labelings. The matching results are not "perfect," and some wrong labels or multiple assignments do occur. However, the key assignments are correctly obtained. In the case of multiple assignments, an interpretation of the results may be required. In the examples presented, our objective has been to evaluate how well the technique performs when the segmentation results are very different and the polygonal approximation is very crude rather than providing some simple examples like the island examples of Davis [12] which are claimed to be trivially solved by the syntactic shape analyzer of Pavlidis [18]. In such situations, the technique depends to some extent on the similarity of the local structures of the model and object. Many times an incorrect label occurs because the local structure of the incorrect match is more similar than that of the correct one. In cases where an object/model segment is broken into more segments, the algorithm is not able to account for it completely. But, the technique is quite robust and effective when applied to real images and it is able to cope to a certain extent with common problems in scene analysis such as noisy features, extra and missing segments and a large number of segments [5]. Moreover, in this regard the algorithm works better than the Hough transform technique of Davis [19] and Ballard [20]. The first stage does not correct all the mistakes of the initial assignment because it uses less world knowledge and some correct assignments are not among the early candidates. The second stage

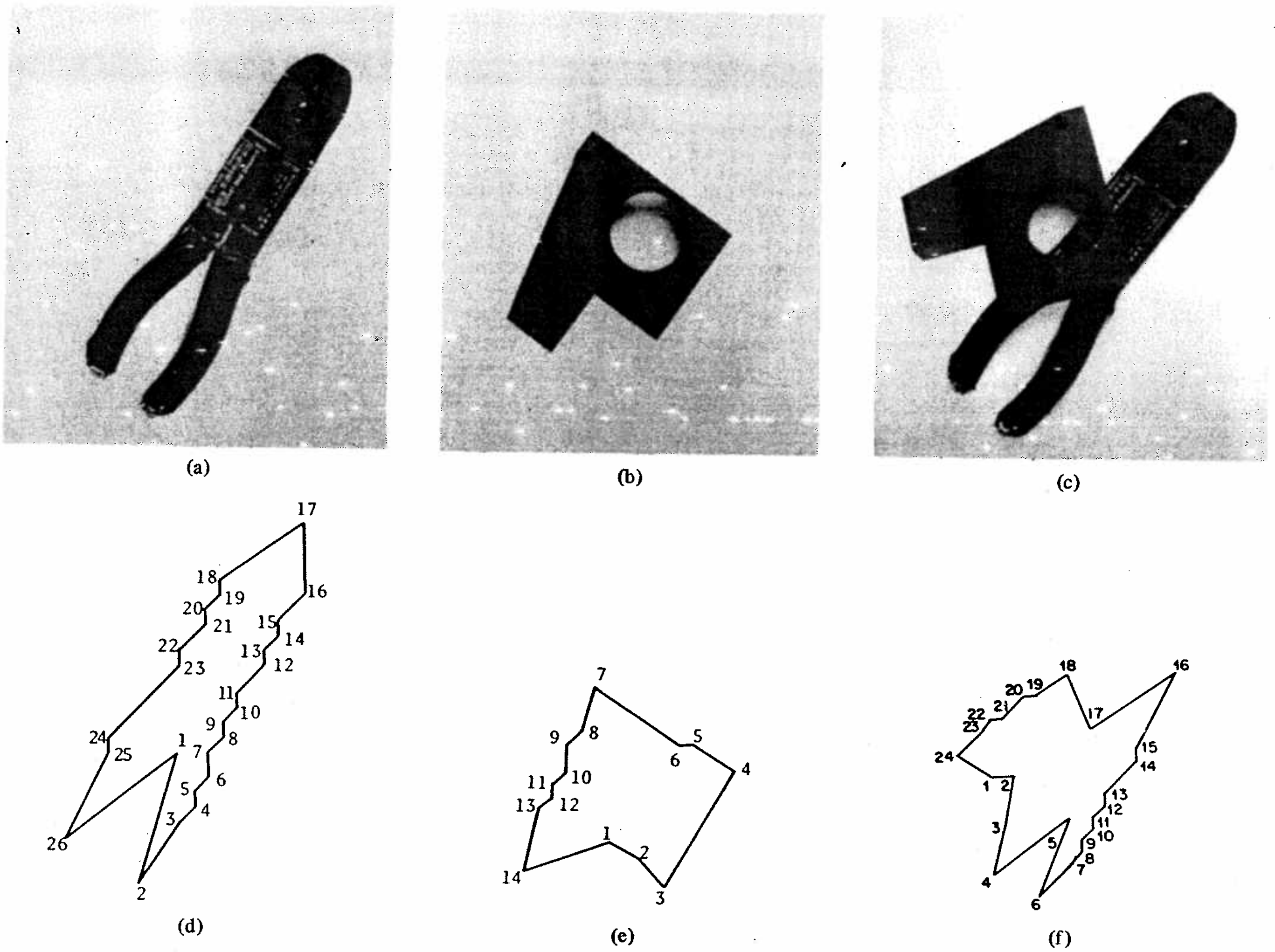


Fig. 9. (a) An industrial piece. (b) An industrial piece. (c) Partial occlusion of industrial pieces in (a) and (b). (d) Model X_1, number of segments = 26. (e) Model X_2, number of segments = 14. (f) Apparent object, number of segments = 24.

corrects mistakes in the labeling of the first stage, and since it uses more world knowledge, it increases the confidence in the matching results. It is possible to generalize the shape matching algorithm to include higher levels of hierarchy at the expense of increased complexity. The technique has been extended for the shape matching of 3-D objects [21].

The shape matching technique is truly hierarchical in the following sense: at the first or second stage the criterion $J^{(1)}$ or $J^{(2)}$ increases of course, but at the first stage if we compute the criterion $J^{(2)}$ or at the second stage if we compute the criterion $J^{(1)}$, it also continually increases with the iteration. This has been verified experimentally. Thus our computation of compatibilities is sound. The algorithms are implemented in SAIL and the computation time varied from 4 s to 20 min on a PDP-10 (KL-10 processor, which performs about 1.8 million operations/s) for matching of varying complexity and number of occluding objects. It includes the timing for segmentation, boundary following, feature computation, matching and several I/O operations. About 70 percent of this time is spent in computing the gradient. The computation of the gradient at the second stage is the most expensive. In the present implementation of the programs we do not store the compatibility values when we compute the compatibility vector. We recompute them when the gradient is needed. The computation time can be reduced by 60 percent if we store compatibility values. It can also be reduced in certain situations if we assume that the objects are rigid bodies. Further savings come by allowing the global maximization of the criteria $J^{(n)}(n = 1, 2)$ by local computations only, i.e., a processor for a unit communicates only with the neighboring processors to compute the gradient and the projection operator. Thus a large amount of parallelism can be introduced. The gradient of the criteria can be computed in two sequential steps. Within each step the processors can work in parallel. In the first step we determine the compatibility vectors and in the second step actually compute the gradients. A processor associated with a unit performs simple operations mostly in parallel while the sequential process of going from one iteration to the next allows these processors to work towards the final goal in a coordinated fashion. The amount of computation per processor is of the order of L^2, where L is the number of classes. It is difficult to develop any useful absolute model for the complexity of the stochastic labeling process, because as the procedure iterates, many labels for a unit become zero and the complexity goes down. As a result the computation time per iteration for the later stages of the process is

TABLE XV
RESULTS OF LABELING FOR THE MODEL X_1 USING THE OCCLUSION ALGORITHM. EXAMPLE 7.

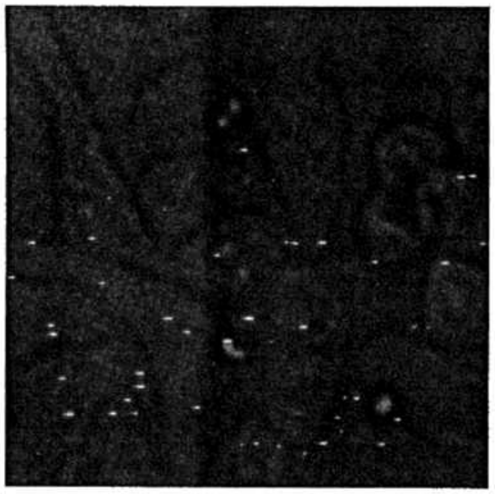
Units of Model X_1	Labels at different iterations					
	First Stage			Second Stage		
	0	1	3	1	3	5
1	5(.25)	5(.37)	5(.62)	5(1.0)	5(1.0)	′5(1.0)
2	6(.26)	6(.42)	6(.60)	6(.64)	6(1.0)	6(1.0)
3	25(.20)	25(.25)	7(.34)	25(.39)	7(.55)	7(1.0)
4	25(.20)	25(.23)	8(.59)	8(.61)	8(1.0)	8(1.0)
5	25(.20)	25(.23)	25(1.0)	25(1.0)	25(1.0)	25(1.0)
6	25(.20)	25(.23)	25(1.0)	25(1.0)	25(1.0)	25(1.0)
7	25(.20)	25(.23)	25(1.0)	25(1.0)	25(1.0)	25(1.0)
8	25(.20)	25(.23)	25(1.0)	25(1.0)	25(1.0)	25(1.0)
9	25(.20)	25(.23)	25(1.0)	25(1.0)	25(1.0)	25(1.0)
10	25(.20)	25(.23)	25(1.0)	25(1.0)	25(1.0)	25(1.0)
11	25(.20)	25(.23)	25(1.0)	25(1.0)	25(1.0)	25(1.0)
12	25(.20)	25(.23)	25(1.0)	25(1.0)	25(1.0)	25(1.0)
13	25(.20)	25(.23)	25(1.0)	25(1.0)	25(1.0)	25(1.0)
14	25(.20)	25(.23)	25(1.0)	25(1.0)	25(1.0)	25(1.0)
15	25(.20)	25(.23)	25(1.0)	25(1.0)	25(1.0)	25(1.0)
16	25(.20)	25(.28)	25(.32)	25(.51)	14(1.0)	14(1.0)
17	25(.20)	25(.33)	25(.51)	25(.74)	25(1.0)	25(1.0)
18	25(.20)	25(.29)	25(.41)	25(.73)	25(1.0)	25(1.0)
19	25(.20)	25(.23)	25(1.0)	25(1.0)	25(1.0)	25(1.0)
20	25(.20)	25(.23)	25(1.0)	25(1.0)	25(1.0)	25(1.0)
21	25(.20)	25(.23)	25(1.0)	25(1.0)	25(1.0)	25(1.0)
22	25(.20)	25(.23)	25(1.0)	25(1.0)	25(1.0)	25(1.0)
23	25(.20)	25(.23)	25(.80)	25(1.0)	25(1.0)	25(1.0)
24	14(.22)	14(.31)	14(.51)	14(.56)	14(1.0)	14(1.0)
25	25(.20)	25(.28)	25(.51)	25(.60)	25(1.0)	25(1.0)
26	25(.20)	25(.28)	4(.39)	25(.40)	25(1.0)	25(1.0)
First term of the objective function	-	3.350	4.715	16.21	17.04	17.20
Penalty term of the objective function	-	0.3350	.4715	1.621	0	0
Criterion	-	3.015	4.244	14.59	17.04	17.20
Penalty Constant	-	.0025	.0085	1.696	-	-

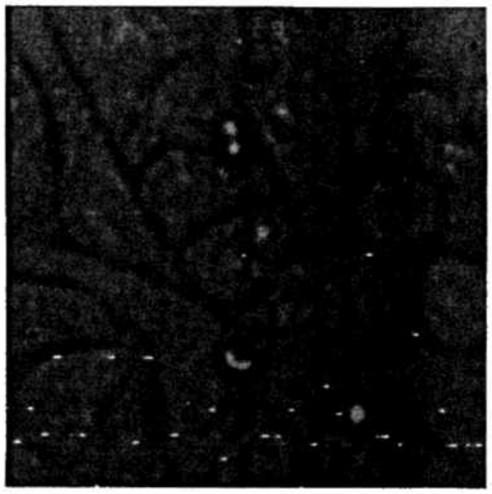
TABLE XVI
RESULTS OF LABELING FOR THE MODEL X_2 USING THE OCCLUSION ALGORITHM. EXAMPLE 7.

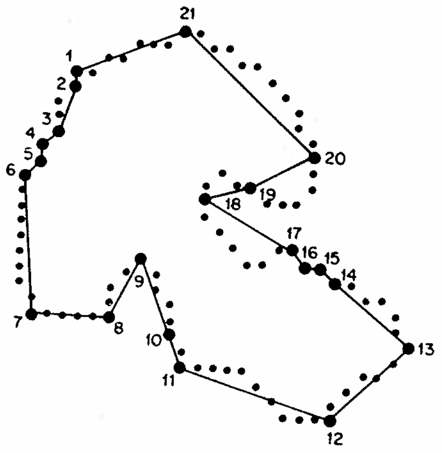
Units of Model X_1	Labels at different iterations					
	First Stage			Second Stage		
	0	1	3	1	3	5
1	25(.20)	25(.31)	25(.50)	25(.51)	25(1.0)	25(1.0)
2	25(.20)	25(.34)	25(.44)	25(.47)	23(.54)	23(1.0)
3	25(.20)	25(.32)	25(.52)	25(.55)	25(.62)	25(1.0)
4	25(.20)	25(.28)	25(.38)	25(.39)	25(.44)	25(1.0)
5	25(.20)	25(.28)	25(.39)	25(.49)	25(1.0)	25(1.0)
6	25(.20)	25(.30)	25(.39)	25(.42)	25(.44)	15(1.0)
7	25(.20)	25(.29)	18(.44)	18(.49)	18(1.0)	18(1.0)
8	25(.20)	25(.33)	25(.49)	25(.57)	25(1.0)	25(1.0)
9	25(.20)	25(.23)	25(1.0)	25(1.0)	25(1.0)	25(1.0)
10	25(.20)	25(.23)	25(1.0)	25(1.0)	25(1.0)	25(1.0)
11	25(.20)	25(.23)	25(1.0)	25(1.0)	25(1.0)	25(1.0)
12	25(.20)	25(.28)	25(.36)	25(.51)	25(.54)	19(1.0)
13	25(.20)	25(.33)	25(.64)	25(.75)	22(1.0)	22(1.0)
14	25(.20)	25(.30)	25(.43)	25(.44)	25(1.0)	25(1.0)
First term of the objective function	-	1.377	2.821	5.684	5.450	7.136
Penalty Term of the objective function	-	.1377	.2821	.5684	0	0
Criterion	-	1.239	2.539	5.116	5.450	7.136
Penalty Constant	-	.00103	.00511	.5947	-	-

less than the early iterations. Normally, for 42 units and 31 classes, we never needed more than a total of 15 iterations of the first and second stages of stochastic labeling. In the worst case, only one label is set equal to zero at every iteration.

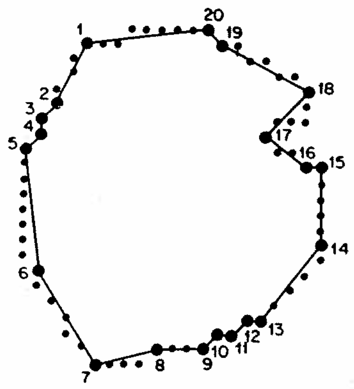
We also presented an extension of the hierarchical stochastic labeling technique to do shape matching of partially occluded 2-D objects by combining the gradient projection method and the penalty function approach. Penalty constants are chosen in an automatic manner. The computation time varies linearly with the number of objects occluding one another. If the objects are rigid as has been mostly assumed in the past work, matching will be relatively simple. After matching actual objects with the apparent object, it will be easier to track them and carry out the motion analysis.


(a)

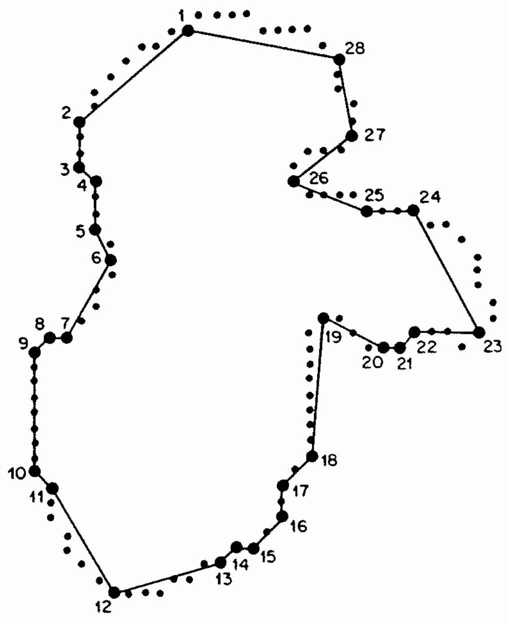

(b)



(c)



(d)



(e)

Fig. 10. (a) The cancer cell undergoing mitosis. (b) The cell in (a) is divided into 2 daughter cells after 15 min. (c) Model X_1, number of segments = 21. (d) Model X_2, number of segments = 20. (e) Apparent object, number of segments = 28.

Pilani, India. In the spring of 1975 he was a graduate research student at the Rensselaer Polytechnic Institute, Troy, NY. From 1975 to 1977 he was a member of the Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge. From 1977 to 1981 he was a Research Assistant at the Image Processing Institute, University of Southern California, Los Angeles. In 1978 he worked as an Academic Associate with the Computer Science Department of IBM Research Laboratory at San Jose, CA. During the winter of 1980, he was a research fellow at INRIA (Institut National de Recherche en Informatique et en Automatique), France. Since July 1981 he has been with the Aeronutronic Division of Ford Aerospace and Communications Corporation, Newport Beach, CA, where he is an engineering specialist in the digital systems department. His current research interests include computer vision, pattern recognition, artificial intelligence, robotics, VLSI, and industrial applications.

Dr. Bhanu is a member of the Association for Computing Machinery, Sigma Xi, IEEE Computer Society, and SPIE.

Olivier D. Faugeras (S'76-M'76) is the Scientific Director of the Computer Vision and Robotics group at INRIA (Institut National de Recherche en Informatique et Automatique), France, and a Senior Lecturer at the University of Paris XI and Ecole Polytechnique. His interests are in computer vision, graphics, robotics, pattern recognition, and artificial intelligence.

Mr. Faugeras is a member of the Association for Computing Machinery.

A Representation for Shape Based on Peaks and Ridges in the Difference of Low-Pass Transform

JAMES L. CROWLEY, MEMBER, IEEE, AND ALICE C. PARKER, MEMBER, IEEE

Abstract—This paper defines a multiple resolution representation for the two-dimensional gray-scale shapes in an image. This representation is constructed by detecting peaks and ridges in the difference of low-pass (DOLP) transform. Descriptions of shapes which are encoded in this representation may be matched efficiently despite changes in size, orientation, or position.

Motivations for a multiple resolution representation are presented first, followed by the definition of the DOLP transform. Techniques are then presented for encoding a symbolic structural description of forms from the DOLP transform. This process involves detecting local peaks and ridges in each bandpass image and in the entire three-dimensional space defined by the DOLP transform. Linking adjacent peaks in different bandpass images gives a multiple resolution tree which describes shape. Peaks which are local maxima in this tree provide landmarks for aligning, manipulating, and matching shapes. Detecting and linking the ridges in each DOLP bandpass image provides a graph which links peaks within a shape in a bandpass image and describes the positions of the boundaries of the shape at multiple resolutions. Detecting and linking the ridges in the DOLP three-space describes elongated forms and links the largest peaks in the tree.

The principles for determining the correspondence between symbols in pairs of such descriptions are then described. Such correspondence matching is shown to be simplified by using the correspondence at lower resolutions to constrain the possible correspondence at higher resolutions.

Manuscript received January 26, 1983; revised May 17, 1983 and August 25, 1983. This work was supported in part by the Robotics Institute, Carnegie-Mellon University, in part by the National Science Foundation under Grant APR75-08154, and in part by the Naval Electronics System Command under Grant N00039-79-Z-0169.

J. L. Crowley is with the Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA 15213.

A. C. Parker is with the Department of Electrical Engineering-Systems, University of Southern California, Los Angeles, CA 90007.

Index Terms—Correspondence matching, difference of Gaussian, difference of low-pass transform, image pyramid, multiresolution representation, shape matching.

I. Introduction

A REPRESENTATION is a *formal system* for making explicit certain entities or types of information, and a specification of how the system does this [20]. Representation plays a crucial role in determining the computational complexity of an information processing problem.

This paper describes a representation for two-dimensional shape which can be used for a variety of tasks in which the shapes (or gray-level forms) in an image must be manipulated. An important property of this representation is that it makes the task of comparing the structure of two shapes to determine the correspondence of their components computationally simple. However, this representation has other desirable properties as well. For example, the network of symbols that describe a shape in this representation have a structure which, except for the effects of quantization, is invariant to the size, orientation, and position of a shape. Thus a shape can be compared to prototypes without having to normalize its size or orientation. An object can be tracked in a sequence of images by matching the largest peak(s) in its description in each image. This representation can also describe a shape when its boundaries are blurred or poorly defined or when the image has been corrupted by various sources of image noise.

This representation is based on a reversible transform referred to as the "difference of low-pass" (DOLP) transform. From