Supervised Learning of Large Perceptual Organization: Graph Spectral Partitioning and Learning Automata

Sudeep Sarkar, *Member*, *IEEE*, and Padmanabhan Soundararajan

Abstract—Perceptual organization offers an elegant framework to group low-level features that are likely to come from a single object. We offer a novel strategy to adapt this grouping process to objects in a domain. Given a set of training images of objects in context, the associated learning process decides on the relative importance of the basic salient relationships such as proximity, parallelness, continuity, junctions, and common region toward segregating the objects from the background. The parameters of the grouping process are cast as probabilistic specifications of Bayesian networks that need to be learned. This learning is accomplished using a team of stochastic automata in an N-player cooperative game framework. The grouping process, which is based on graph partitioning is, able to form *large* groups from relationships defined over a small set of primitives and is fast. We statistically demonstrate the robust performance of the grouping and the learning frameworks on a variety of real images. Among the interesting conclusions are the significant role of photometric attributes in grouping and the ability to form large salient groups from a set of local relations, each defined over a small number of primitives.

Index Terms—Perceptual organization, learning in vision, learning automata, Bayesian networks, feature grouping, object recognition, figure ground segmentation.

1 Introduction

One of the fundamental problems in intermediate level vision is the selection of low-level features that belong to one object. This process has been referred to by the computer vision community as perceptual organization, feature grouping, saliency detection, object hypothesis generation, or simply as segmentation. This is one of the largely unsolved, fundamental problems in vision and is central to the design of artificial vision systems. Clemens and Jacobs [1] have shown that recognition by "indexing is of very limited benefit unless accompanied by a grouping process." They showed that the performance of an indexing system is dependent on the ability of a grouping system to generate "pure" groups with no background clutter. In fact, the sizes of these groups are directly related to indexing speedup. On a similar note, Grimson [2] has shown that the combinatorics of the recognition process in cluttered environments using constrained search reduces from an exponential to a low order polynomial if we use an intermediate grouping process. What is remarkable is that, unlike for the indexing case, this grouping process need not be perfect!

Goal. Gestalt psychologists have offered a set of laws that are important in figure-ground segmentation, such as the laws of parallelism, continuity, similarity, symmetry, common region, and closure [3]. The use of these principles in computer vision is not new [4]. However, their usage has been limited in two aspects. First, the ability to tune the relative importance of these relationships has not been exploited. For example, in some domains, the parallelism relation might be a better discriminant between object and background than continuity. In such cases, we would like to weigh parallelism more than continuity. In addition, the definition of the salient relationships themselves entails uncertainty. We offer a framework that casts the grouping parameters as probabilities which are learned from a set of training images of objects in their natural contexts.

Second, most past efforts have been to form simple, small groups of features such as parallels [5], convex outlines [6], ellipses [7], and rectangles [8]. This is partly because of the rarity of fast frameworks to form large feature groups. The computational difficulty arises from the fact that the search space for large groups grows exponentially with the number of features in a group. But, large feature groups are important. It is highly unlikely for large organized groups to arise by chance. Hence, according to the law of accidentalness [4], the significance of a large organization is higher than a small organized form. We present a computational model that integrates a variety of salient relationships, such as parallelism, continuity, common region, and perpendicularity, among extended tokens to form large groups.

Approach to the Solution. One possible strategy for deciding on the relative importance of salient geometric relationships is to consider 2D or 3D object models. Statistical analysis of these models could provide estimates of the various grouping parameters. For example, one could look at the distribution of the angles between pairs of straight lines in the model and decide on the grouping angle tolerance. However, isolated object models do not

• *The authors are with the Computer Science and Engineering Department, University of South Florida, 4202 E. Fowler Ave., ENB 118, Tampa, FL 33620. E-mail: {sarkar, psoundar}@csee.usf.edu.*

Manuscript received 2 Dec. 1997; accepted 21 Oct. 1999.
Recommended for acceptance by G. Medioni.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 107780.

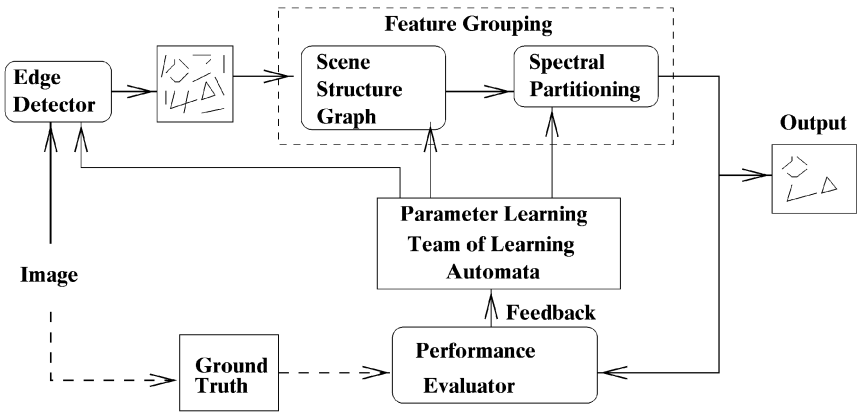


Fig. 1. System block diagram of the proposed framework.

constitute a sufficient basis for the grouping parameter decisions. It is the job of the grouping algorithm to *segregate* an object from not only the background clutter, but also from other objects in the scene. Based on isolated object models, we might be able to decide on the associative parameter values between features within a model, but these values will not guarantee segregation either from the background clutter or from other objects. We have to also consider the statistics of the background clutter and the scene context of the objects. However, the modeling of both these factors is an open and difficult problem. So, we suggest using a training set of images of objects in context, but with the objects of interest manually outlined. Based on this training set of images, the importance of each relationship is learned using an N-player stochastic automata game framework. Unlike the usual gradient descent algorithms, which can guarantee only a local minimum, the learning automata-based N-player game framework converges to the global optimum with the proper choice of its learning rate [9]. Observe that, in this framework, the influence of the object models on the grouping process is only statistical in nature and is implicit through the use of training images. We do not require explicit, detailed object models.

We assemble the contributions of the individual salient relationships over a small number of primitives using a graph—the scene structure graph. This graph is partitioned to form large groups of features using the graph spectrum. Graph spectrum refers to the *ordered* set of eigenvalues (along with their eigenvectors) of the matrix representation of a graph. The overall grouping process is fast. For a 512 by 512 image, it takes, on average 5 seconds, (on a Sparc Ultra) to compute the salient groups. As we shall see, the algorithm can cope with significant image clutter.

Fig. 1 depicts the overview of the system. The input to the grouping algorithm consists of low-level image features such as constant curvature edge segments (arcs and straight lines). The output consists of salient groups of low-level features. The feature grouping algorithm consists of two parts: scene structure graph construction and spectral partitioning. A weighted relational graph captures the salient relationships among the edge tokens. Probabilistic Bayesian networks quantify these salient relationships. The uncertainty in the definition of the relationships and their relative importance are captured using probability measures. Section 2 discusses in detail this relational graph construction. Section 3 outlines the graph spectral algorithm used to partition the relational graph into feature clusters. Because of the use of graph representations, the output groups do not have a single global functional description, such as elliptical, parallelogram, etc., but are described by the strong pairwise interactions between the features. This definition of groups tends to encompass a larger class of feature distributions than functional descriptions.

The probabilities that underly the relational graph, along with the other algorithm parameters, are learned using an N-player automata game framework. As we shall see, in addition to the six prior probabilities that capture the relative importance of the salient relations, we have nine grouping tolerance parameters and six feature detection parameters that need to be chosen. The learning framework, which decides on all these parameters is able to account for dependence among parameters in the search process. We believe that the learning framework can also be used to learn parameters of other vision algorithms. These learning automata need supervisory feedback. This feedback is automatically generated by comparing the output of the grouping algorithm with manually outlined training images. The learning algorithm is discussed in detail in Section 4. In Section 5, we present results and we conclude with Section 6.

Previous Approaches. Learning is one of the thrust areas in vision and has been used for feature selection [10], texture classification [11], shape classification [12], region segmentation [13], and object recognition and modeling [14], [15]. However, the use of learning in perceptual organization is new.

There is also work in parameter selection for vision systems, such as [16], [17], [18]. More recently, Peng and Bhanu [19] presented a closed loop recognition system whose segmentation parameters are tuned based on feedback from the recognition module. They employ a team of connectionist Bernoulli quasilinear units with one unit associated with each value that each parameter can take. Unlike the currently used search techniques, the team of learning automata that we advocate can take into account parameter dependencies in the search process.

Most work in perceptual organization for 2D images has concentrated on extracting continuous contours by grouping

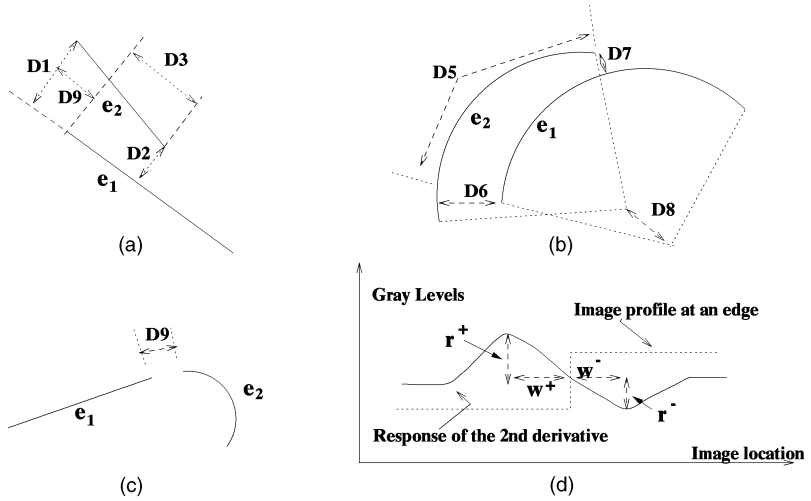


Fig. 2. The geometric attributes used to classify pairwise edge segment relationships are shown in (a), (b), and (c). The length of the segment e_1 is greater than the length of e_2. The photometric attributes of an edge pixel are shown in (d).

image pixels, using primarily the properties of proximity and continuity [20], [21], [22], [23], [24], [25], [26], [27]. Of the work in perceptual organization with extended primitives, such as lines or arcs, *the effort has been mostly to form simple, small groups of primitives* such as parallels [5], convex outlines [6], ellipses [7], [28], and rectangles [8], [28], [29]. Among the frameworks that can form large groups is the solution suggested by Herault and Horaud [30]. They use simulated annealing to solve the figure ground problem. They group edgels based on a quadratic cost function constructed out of terms based on cocircularity, smoothness, and proximity. McCafferty [31] combines different Gestalt principles into one energy functional which is optimized using simulated annealing. As with other energy minimization and cost function based methods, these strategies are necessarily computationally expensive.

In computer vision, Sarkar and Boyer [32], [33] used graph spectral techniques to capture the structure or organization in a scene for change detection. Shapiro and Brady [34] used the eigenvectors of a proximity graph to establish correspondence between two sets of features. Sengupta and Boyer [35] used the eigenvalues of the connectivity relation matrix of a 3D model to extract global attributes of an object. Recently, Shi and Malik [22] used normalized cut-based spectral techniques to partition graphs of image pixels into regions. In this paper, we use the graph spectral partitioning technique to group extended 2D features.

2 Scene Structure Graph Specification

We represent the full scene structure using a graph whose nodes are the image feature primitives, such as constant curvature segments, and the links between the nodes denote relations. We will refer to such a graph as the scene structure graph (**SSG**). The links denote the Gestalt inspired 2-ary relationships of parallelism, perpendicularity, continuity, proximity, and common region [3]. Parallelism can exist between two straight edge segments or between two arcs (ribbons). Similarly, we consider continuity between two straight lines or between two arcs. We consider two types of perpendicularity: T-junctions and L-junctions, both of which are usually considered important from an object recognition point of view. Common region refers to the relationship that two image primitives share or are embedded in the same image region (photometric property). This relation has been recently suggested by Gestalt psychologists as being important.

It is certainly logical to raise doubts about the sufficiency of 2-ary relationships in capturing the structure of a scene. In general, richer k-ary relations, represented using hypergraphs, might be necessary. The graph spectral-based grouping framework, outlined in the next section, can be easily modified to handle hypergraphs by first transforming them to equivalent 2-ary graphs. Recent work [36] indicates it is possible to convert hypergraph representations into a 2-ary graph (with additional dummy nodes) that has the same partitioning properties as the original hypergraph. The present use of 2-ary relations is primarily due to computational considerations. The order of search complexity increases with the size of the k-ary relations. Also, as we shall see in Section 5, even with just 2-ary relations, we are able to generate high quality large groups.

The classification and the quantification of the binary (2-ary) relations are as follows: For every pair of constant curvature edge segments (straight lines and arcs), we compute the distances as shown in Fig. 2. The computations are referenced with respect to the larger segment, thus ensuring that the relationship definitions are symmetric. The distances, D1 and D2 represent the maximum and the minimum distances, respectively, of the end points of smaller line to the larger line. Similarly, D6 and D7 represent the maximum and minimum distances of the end points of the smaller arc to larger arc, respectively. The overlap between two straight (arc) edge segments is represented by D3 (D5). The distance between the centers of two arcs is represented by D8. The minimum distance between the *end points* of two segments is denoted by D9. We also compute the difference in slopes, θ, between two straight lines. These distances are absolute quantities, and hence, are scale dependent. To make them scale independent, we form their ratio with the length of the larger segment of the pair, l_{max}, to arrive at the attributes shown in Table 1.

TABLE 1
Scale Invariant Geometric and Photometric Attributes Computed between Each Pair of Edge Segments

$d_{min} = \frac{D2}{l_{max}}$ (straight) or $d_{min} = \frac{D7}{l_{max}}$ (arc)	$d_{max} = \frac{D1}{l_{max}}$ (straight) or $d_{max} = \frac{D6}{l_{max}}$ (arc)
$o_{lap} = \frac{D3}{l_{max}}$ (straight) or $o_{lap} = \frac{D5}{l_{max}}$ (arc)	$c_{dist} = \frac{D8}{l_{max}}$ $e_{dist} = \frac{D9}{l_{max}}$
$r_{mag} = \max\left\{\frac{\lvert r_i^+ - r_j^+\rvert}{0.5(r_i^+ + r_j^+)}, \frac{\lvert r_i^- - r_j^-\rvert}{0.5(r_i^- + r_j^-)}\right\}$	$r_{width} = \max\left\{\frac{\lvert w_i^+ - w_j^+\rvert}{0.5(w_i^+ + w_j^+)}, \frac{\lvert w_i^- - w_j^-\rvert}{0.5(w_i^- + w_j^-)}\right\}$

The attributes that are specific to straight lines and arcs are labeled as such.

In addition to the geometric attributes, we compute two photometric attributes, r_{mag} and r_{width}, as listed in the last row of Table 1. We fashion these attributes after the photometric attributes used by Boyer et al. [37] for dynamic edge warping in the context of stereo matching as follows: To compute the edges, we use the Canny step edge detector. The maxima in the Canny detector response represents the edge location. These maxima are detected as zero crossings of the second directional derivatives. Along with a zero crossing, the second derivative produces two lobes on each side of the step edge, as shown in Fig. 2d. The heights and the distances of these lobes from the edge location form the photometric attributes of each edge pixel. The slope of the zero crossing is implicit in these four quantities. In addition, the average values of the four attributes capture the photometric properties of the region around an edge. The attributes r^+ and w^+ capture the properties on one side of the edge and r and w capture the properties on the other side. This is unlike the slope of the zero crossing, which just captures the local photometric property at the edge. The averages of r^+, r , w^+, and w over an edge segment form the photometric attributes of the segment. We define the photometric compatibility of two segments using r_{mag} and r_{width}, which are defined as the fractional differences between the average attributes of two edge segments, e_i and e_j (Table 1). Two segments that share a common region will tend to have low values for these two photometric attributes.

2.1 Bayesian Networks

Based on the values of the photometric and geometric attributes (Table 1), we classify each pair of edge segment into straight parallel, T-junction, L-junction, continuous, ribbon, proximal, sharing a common region, or none-of-the-above. This classification requires noncrisp representations of the relationships, which are, typically, uncertain. We use probabilistic Bayesian network representations to model these uncertainties. Bayesian networks are graphical representations of joint probability specifications [38]. The nodes of a Bayesian network represent the individual random variables and its directed links denote the direct dependencies between two variables. The links are quantified by the respective conditional probabilities. Not only does the network representation explicitly encode the dependencies between variables, but it also facilitates efficient probabilistic updating upon the arrival of new information.

Four Bayesian networks (see Fig. 3) classify the 2-ary relationships. One network classifies pairs of straight lines

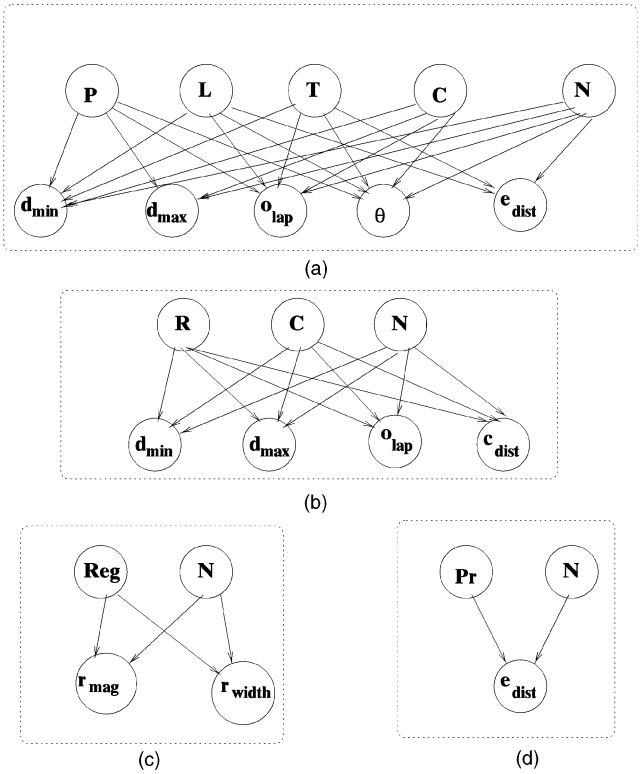


Fig. 3. Bayesian networks used to classify pairs of edge segments into the Gestalt inspired salient relationships. The networks in (a) and (b) classify pairs of straight lines and arcs, respectively. The network in (c) computes the significance of the photometric similarity and the proximity significance is computed using (d).

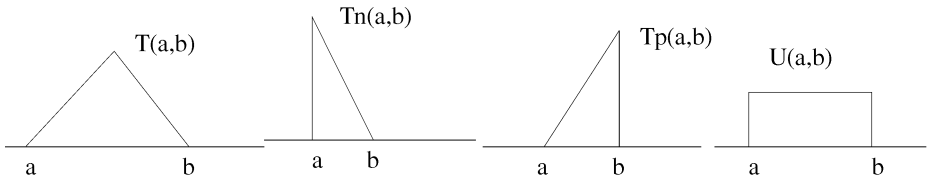


Fig. 4. Basic forms of the conditional probabilities of the Bayesian networks.

into parallels(P), T-junctions(T), L-junctions(L), or continuous lines(C). The second network classifies pairs of arcs into cocircular(C) and parallels (ribbons, R). The third network computes the significance of the region similarity (Reg) between the edge segment. The fourth network classifies proximity relations (Pr). The random variables of the Bayesian networks are the relational attributes and the relations themselves, denoted by P, T, L, C, R, Reg, and Pr. Note that there is a node in each net denoted by N. This node represents the none-of-the-above choice and captures the probability that the line arrangement could have arisen just by chance. Bayesian networks allow the consideration of the dependence of different relational types upon each another. As a consequence, the quantification of a relationship, such as parallelism, between a line pair takes into account not only the extent to which the line pair participates in the parallelism relation, but also the extent of its participation in other possible relationships such as continuity, perpendicularity, etc.

The probabilities that need to be specified in the Bayesian network are the prior probabilities of the various relations (P, L, T, R, C, Pr, N, Reg) and the conditional probabilities of the relational attributes given the relations. The prior probabilities constitute an efficient mechanism for incorporating the relative importance of the various relationships. A low prior value for a relation would result in low final probabilities, thus weighing down the effect of that relation. A high prior would indicate high importance of the relation. In the absence of evidence to the contrary, we assume equal prior for none-of-the-above (N) relation. Since the ribbon (R) and parallel (P) relations denote essentially the same relation, namely, parallelism, we use the same prior for both of them. Thus, we have six priors that can chosen (or learned), namely the priors for P, L, T, C, Pr, and Reg.

For the conditional probabilities, we need to specify the probability of an attribute given the state of its parents in the Bayesian network. For example, the relational attribute, d_{max}, has P, C, and N as its parents. So, we need to specify: $P(d_{max} = d|P = p, C = c, N = n)$, where p, c, and n denote the *binary* states of the parents. In the general case, this would require specifying eight conditional probabilities for $d_{max} = d$, corresponding to various combinations of the states of the parents. However, in our case, we know that a pair of straight line can exhibit only *one* of the three relations. Thus, we need to specify only

$$P(d_{max} = d|P = 1, C = 0, N = 0),$$
$$P(d_{max} = d|P = 0, C = 1, N = 0),$$

and

$$P(d_{max} = d|P = 0, C = 0, N = 1);$$

the probabilities for other combinations are zero. These three conditional probabilities represent the distribution of d_{max} for a parallel, continuous, and none-of-the-above relationships, respectively. For a parallel relation, d_{max} should neither be zero nor should it be very large. Recall that d_{max} is a measure of the distance between the two lines. Thus, the parallel lines should not be collinear, which is the case accounted for by the continuity relation, nor should they be very far apart. So, we represent the density using the triangular function, $T(0, b)$, shown in Fig. 4. However, d_{max} should be ideally zero for a continuity relationship, so we choose $P(d_{max} = d|P = 0, C = 1, N = 0)$ to be of the form $Tn(0, b)$, as shown in Fig. 4. The node N represents the completely random scenario, thus $P(d_{max} = d|P = 0, C = 0, N = 1)$ is a uniform density function over (0, 1).

Table 2 lists the forms of the conditional probability densities of the Bayesian networks. We construct all the conditional probabilities out of the density functions shown in Fig. 4. The differences are in the parameters of the functions. All the conditional density functions are characterized by seven parameters, d_{tol}, o_{tol}, θ_{tol}, c_{tol}, d_{cont}, p_{tol}, and r_{tol}. These parameters represent the effective tolerances used in the grouping process. Thus, d_{tol} is the distance tolerance, o_{tol} is the overlap tolerance, θ_{tol} is the orientation tolerance, c_{tol} is the tolerance between two arc centers, d_{cont} is the distance tolerance for continuity, p_{tol} is the proximity tolerance, and r_{tol} represents region tolerance.

2.2 Quantification of the Scene Structure Graph

The described Bayesian networks classify each edge segment pair into different salient Gestalt inspired relations. Each pair of edge segments instantiates the respective attribute nodes in the Bayesian networks. Messages propagate in the network according to the method of conditioning [38], updating the probabilities. The parent node with the highest probability determines the type of the relation between the pair of segments. The value of the probability quantifies the quality of the relation. Thus, $Prob(P_{ij})$ denotes the confidence that the relationship between the ith and jth features is parallelism. We combine the quantified relations to generate the link weights of the scene structure graph (**SSG**), w_{ij}, between two nodes as shown below:

$$\begin{aligned} w_{ij} &= \left\{ \begin{array}{l} \max\ Prob(P_{ij}), Prob(R_{ij}), Prob(L_{ij}), \\ Prob(C_{ij}), Prob(T_{ij})) + Prob(Pr_{ij}) + Prob(Reg_{ij}) \end{array} \right\} \\ &= 0 \text{ if } Prob(N_{ij}) \geq \{Prob(P_{ij}), Prob(R_{ij}), \\ &\quad Prob(L_{ij}), Prob(C_{ij}), Prob(T_{ij})\}. \end{aligned} \tag{1}$$

TABLE 2
Conditional Probabilities Used in the Bayesian Networks

$P(d_{min}\|P=1,L=0,T=0,N=0)=T(0,2d_{tol})$ $P(d_{min}\|P=0,L=1,T=0,N=0)=Tn(0,d_{tol})$ $P(d_{min}\|P=0,L=0,T=1,N=0)=T(0,d_{tol})$ $P(d_{min}\|P=0,L=0,T=0,N=1)=U(0,1)$	$P(d_{min}\|R=1,C=0,N=0)=T(0,2d_{tol})$ $P(d_{min}\|R=0,C=1,N=0)=Tn(0,d_{tol})$ $P(d_{min}\|R=0,C=0,N=1)=U(0,1)$
$P(d_{max}\|P=1,C=0,N=0)=T(0,2d_{tol})$ $P(d_{max}\|P=0,C=1,N=0)=Tn(0,d_{tol})$ $P(d_{max}\|P=1,C=0,N=0)=U(0,1)$	$P(d_{max}\|R=1,C=0,N=0)=T(0,2d_{tol})$ $P(d_{max}\|R=0,C=1,N=0)=Tn(0,d_{tol})$ $P(d_{max}\|R=0,C=0,N=1)=U(0,1)$
$P(o_{lap}\|P=1,L=0,T=0,C=0,N=0)=Tp(1-o_{tol},1)$ $P(o_{lap}\|P=0,L=1,T=0,C=0,N=0)=T(-o_{tol}/2,-o_{tol}/2)$ $P(o_{lap}\|P=0,L=0,T=1,C=0,N=0)=Tn(0,o_{tol}/2)$ $P(o_{lap}\|P=0,L=0,T=0,C=1,N=0)=Tp(-d_{cont},0)$ $P(o_{lap}\|P=0,L=0,T=0,C=0,N=1)=U(0,1)$	$P(o_{lap}\|R=1,C=0,N=0)=Tp(1-o_{tol},1)$ $P(o_{lap}\|R=0,C=1,N=0)=Tp(-d_{cont},0)$ $P(o_{lap}\|R=0,C=0,N=1)=U(0,1)$
$P(\theta\|P=1,L=0,T=0,C=0,N=0)=Tn(0,\theta_{tol})$ $P(\theta\|P=0,L=1,T=0,C=0,N=0)=Tp(1-\theta_{tol},1)$ $P(\theta\|P=0,L=0,T=1,C=0,N=0)=Tp(1-\theta_{tol},1)$ $P(\theta\|P=0,L=0,T=0,C=1,N=0)=Tn(0,\theta_{tol})$ $P(\theta\|P=0,L=0,T=0,C=0,N=1)=U(0,1)$	$P(c_{dist}\|R=1,C=0,N=0)=Tn(0,c_{tol})$ $P(c_{dist}\|R=0,C=1,N=0)=Tn(0,c_{tol})$ $P(c_{dist}\|R=0,C=0,N=1)=U(0,1)$
$P(e_{dist}\|L=1,T=0,N=0)=Tn(0,d_{tol})$ $P(e_{dist}\|L=0,T=1,N=0)=Tn(d_{tol}/2,1)$ $P(e_{dist}\|L=0,T=0,N=1)=U(0,1)$	$P(e_{dist}\|Pr=1,N=0)=Tn(0,p_{tol})$ $P(e_{dist}\|Pr=0,N=1)=U(0,1)$
	$P(e_{dist}\|Pr=1,N=0)=Tn(0,p_{tol})$ $P(e_{dist}\|Pr=0,N=1)=U(0,1)$
	$P(r_{mag}\|Reg=1,N=0)=Tn(0,r_{tol})$ $P(r_{mag}\|Reg=0,N=1)=U(0,1)$
	$P(r_{width}\|Reg=1,N=0)=Tn(0,r_{tol})$ $P(r_{width}\|Reg=0,N=1)=U(0,1)$

The functions $T, Tn, Tp,$ and U are as defined in Fig. 4.

This results in a single value for each edge, as opposed to a vector weight.

3 Graph Spectral Partitioning

We form large organized groups of primitives by searching for clusters of nodes that are loosely connected to the rest of the nodes in the scene structure graph **SSG**. To find these node clusters, we cast the problem of grouping image primitives into a partitioning problem of the scene structure graph, **SSG**(N, E), where N is the set of nodes and E is the set of weighted edges. We compute this partitioning recursively by first cutting the graph into two parts, N_1 and N_2, which are further bisected. The process continues until we have partitions that are small enough.

We represent a graph bisection by a vector $\mathbf{v}$ whose sign of the ith component (v_i) represents the membership of node i in one or the other set; positive components indicate the nodes for one set and the negative components indicate membership in the other. Denoting the weight of an edge between nodes i and j as w_{ij}, we cast our graph bisection problem as:

$$\min \sum_{ij} w_{ij}(v_i \quad v_j)^2 \tag{2}$$

subject to the constraints that 1) $\sum_i v_i = 0$ and 2) $\sum_i v_i^2 = 1$.

The minimization of above term will tend to assign similar weights (v_i, v_j) to nodes (i and j) between which there is a large link weight (w_{ij}). The difference between v_i and v_j for nodes that are weakly connected will tend to be large. The two constraints prevent the trivial solution of $\mathbf{v} = \mathbf{1}$ and $\mathbf{v} = \mathbf{0}$, respectively. The first constraint will force negative and positive values for v_i with the convenient consequence that the negative entries would correspond to one partition and the positive values would constitute the other partition.

We can merge the second constraint with the minimized term in (2) to recast the problem as:

$$\min \frac{\sum_{ij} w_{ij}(v_i \quad v_j)^2}{\sum_i v_i^2} \quad \text{such that } \sum v_i = 0. \tag{3}$$

The numerator of the minimized term can be rearranged as follows:

$$\begin{aligned}\sum_{ij} w_{ij}(v_i \quad v_j)^2 &= 2\sum_{i=1}^{N}\left(\sum_{j=1}^{N} w_{ij}\right)v_i^2 \quad 2\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}v_iv_j \\ &= 2\mathbf{v}^T\mathbf{L}_G\mathbf{v},\end{aligned} \tag{4}$$

where $\mathbf{L}_G$ (known as the Laplacian matrix) is an $N \times N$ sized array with the following entries:

$$\mathbf{L}_G(i,j) = \begin{cases} \sum_{j,j\neq i} w_{ij} & \text{if } i = j \\ w_{ij} & \text{if } i \neq j \end{cases} \tag{5}$$

for every $i, j = 1, \cdots, N$ and where w_{ij} is the weight of edge between nodes i and j. Thus, the minimization process can be compactly expressed using vector notation as:

$$\min 2\frac{\mathbf{v}^T\mathbf{L}_G\mathbf{v}}{\mathbf{v}^T\mathbf{v}} \qquad \text{such that } \mathbf{v}^T\mathbf{1} = 0, \tag{6}$$

where $\mathbf{1}$ is vector with all entries equal to one. The solution of this minimization can be easily constructed from the Courant Fischer Minimax Theorem [39, p. 179]. The following corollary of the Courant Fischer theorem gives a variational characterization of the eigenvalues of a matrix.

Corollary 1 (from Courant-Fisher). *Let A be a real symmetric matrix with eigenvalues, $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and let the corresponding eigenvectors be $\mathbf{v_1}, \cdots, \mathbf{v_n}$. Then,*

$$\lambda_k = \min_{\mathbf{v}, \mathbf{v} \neq 0, \mathbf{v} \perp \mathbf{v_1}, \cdots, \mathbf{v_{k-1}}} \frac{\mathbf{v}^T A \mathbf{v}}{\mathbf{v}^T \mathbf{v}}. \quad (7)$$

In particular, the first eigenvector, $\mathbf{v_1}$, minimizes the quadratic expression in (7) with the minimum value being λ_1. The second eigenvector provides a minimizing solution *orthogonal* to the first eigenvector. The third eigenvector provides a solution that is orthogonal to both the first and the second eigenvectors, and so on. Equation (7) with $k = 2$ determines the solution for (6). This follows from the fact that the first eigenvalue λ_1 of $\mathbf{L}_G$ is zero and $\mathbf{v_1} = [1, 1, \cdots, 1]$. Thus, the condition in (7) that the second eigenvector is orthogonal to $\mathbf{v_1}$ reduces to $\sum_i \mathbf{v}_2(i) = 0$, which is the constraint of the minimization. *Thus, the minimum value for the expression in (6) is $2\lambda_2$ and the solution is the second eigenvector of the Laplacian matrix.* Given the second eigenvector of the Laplacian, the partition is obtained by assigning the positive entries to one set and the negative ones to the other. This spectral partitioning technique was first introduced by Fiedler [40], later on reused by Pothen et al. [41], and is presently used to determine load assignment in parallel and distributed computing scenarios.

The second eigenvalue (λ_2) of the Laplacian matrix of a graph is also commonly used as a measure of the connectivity of the graph. It can also be shown [42] that if $G = (N, E)$ is a graph, and $G1 = (N, E1)$ a subgraph, i.e., with the same nodes and a subset of the edges, so that $G1$ is "less connected" than G, then $\lambda_2(\mathbf{L}_{G_1}) < \lambda_2(\mathbf{L}_G)$, i.e., the algebraic connectivity of $G1$ is also less than or equal to the algebraic connectivity of G.

It is interesting to note that, using the derivation in [22], one can show that the above partitioning technique offers us an approximate solution to the problem of minimizing the total link weight *between* the two partitions, N_1 and N_2 normalized by the size of the two sets—$Cut(N_1, N_2)(\frac{1}{|N_1|} + \frac{1}{|N_2|})$—and, hence, can be referred to as the *average cut* solution. In [22], Shi and Malik suggest spectral partitioning that approximate, another cut measure, namely, the *normalized cut*, for image region segmentation. The normalized cut minimizes the total link weight *between* the two partitions, N_1 and N_2, normalized by the association of the nodes within the two sets: $Cut(N_1, N_2)(\frac{1}{Assoc(N_1)} + \frac{1}{Assoc(N_2)})$. As we shall see in Section 5, normalized cut-based partitioning, with its added computational complexities, does not produce significantly different groups from average cut-based partitioning for extended edges.

In summary, the graph spectral partitioning solution operates by recursively partitioning each part. The stopping condition of the recursion involves a threshold on the maximum partition strength, which measures how strong a cluster one wants to break. The other stopping condition is the minimum cluster size beyond which we do not partition. We learn these two parameters, along with the six priors for the relations (see Section 2.1) and the seven tolerances that specify the conditional probability specification of the Bayesian network (see Section 2.1), using the automata-based learning algorithm discussed in Section 4.

Complexity of the grouping process. The spectral partitioning technique involves the computation of eigenvectors at each stage. Standard routines for eigenvector computations are $\mathcal{O}(N^3)$. At each stage of the recursion, the problem size reduces by two. Using the master theorem, we can show that the complexity of a size N partitioning problem is $\mathcal{O}(N^3)$. However, in practice, the scene structure graph is sparse and we can significantly improve the execution speed by using sparse matrix eigenvalue computation routines. Besides, we do not need to compute all the eigenvectors of a matrix; we need to compute only the second eigenvector at each iteration. The second eigenvector can be computed in $\mathcal{O}(N)$ using the Lanczos algorithm, thus resulting in an overall partitioning complexity of $\mathcal{O}(N \log N)$. The scene structure graph construction is $\mathcal{O}(N^2)$. So, the overall complexity of grouping is $\mathcal{O}(N^2)$.

4 Learning Grouping Parameters

As with any perceptual organization strategy, the spectral grouping algorithm also has parameters that need to be chosen. Specifically, we have 15 parameters: seven tolerance parameters used in the Bayesian network to construct the scene structure graph (see Section 2.2), six prior probabilities for the relations (see Section 2.1), and two parameters (see Section 3) used in spectral partitioning, namely, minimum cluster size and maximum partition strength. These parameters are in addition to the three edge detection parameters of edge scale, σ, edge strength threshold, and edge length threshold, and the three parameters of the constant curvature contour segmentation algorithm. Thus, there are 21 parameters that have to be chosen. The advantage of this large number of parameters is the flexibility of the grouping algorithm. The down side is that we need an effective strategy to choose these parameters. In this section, we present a strategy to learn these parameter given a training set of images.

This problem of automated parameter selection is also present in other computer vision contexts. The usual practice is to choose such parameters by trial and error or using heuristics. However, when we network a number of vision modules, the number of parameters grows and manual choice becomes difficult. The parameter choice problem has three characteristics that make it computationally expensive in practice. First, the search space is extremely large. Let N_p be the number of parameters to be chosen and r be the number of possible values for each parameter. Then, the total number of possible parameter combinations is r^{N_p}. Second, for a network of vision modules, tuning of parameters on a per module basis does not guarantee overall optimal choice. In practice, the choice of the parameters depends upon each other. This is true not only between parameters of a particular module, but also between parameters from different modules. Thus, not only can there be dependence between the choice of the scale, σ, and the thresholds of the edge detector, but there can also be dependence between the scale, σ, and, say, the distance tolerance factor (d_{tol}) used in the grouping algorithm. With the increase in edge scale, the contours move away from each other and edges become sparse, which can affect the distance tolerance. Third, the optimal parameter choices are typically dependent on the image domain.

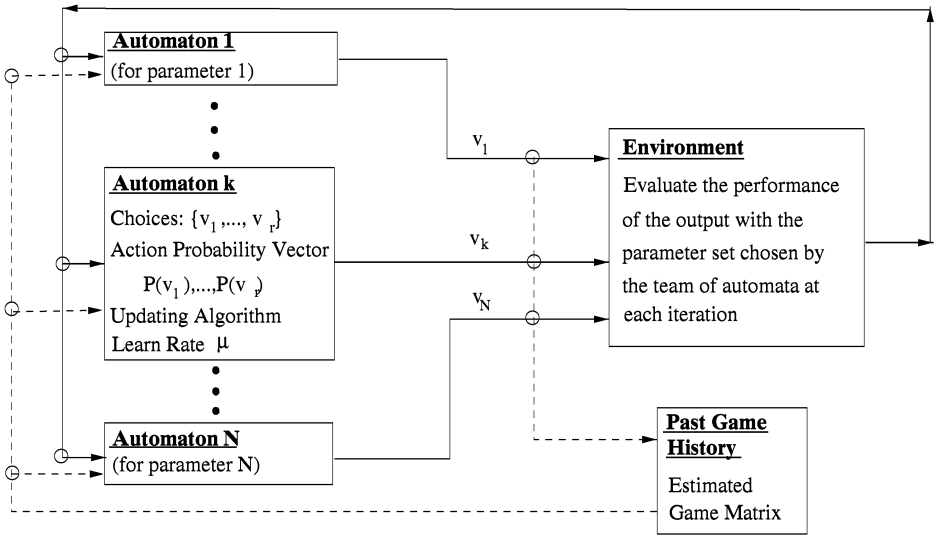


Fig. 5. Team of learning automata for learning parameter combinations.

The goal of the learning algorithm is to learn a set of parameter combinations that result in good performance on a class of images characterized by the training set. The learned set of parameters is composed of combinations that result in good performance on each of the training images. For the experiments presented in Section 5, we chose the size of the learned set of parameter combinations to be 100. Given a new image, all 100 parameter combinations would be tried, which of course is far better than trying 10^{21} combinations. Note that we implicitly encode the dependence of parameter on images; there are good parameter combinations for each training image in the chosen set. Thus, if the new image has characteristics similar to one of the training images, a subset of learned parameter combinations will result in good performance.

There are various other possible strategies for selecting the set of parameter combinations that result in good performance. Peng and Bhanu [19] employ a team of connectionist Bernoulli quasilinear units, with one unit associated with each value that each parameter can take. There are no interactions between the units. Sometimes, the optimal parameter selection process is cast as an optimization problem of an energy function [43], [44] and traditional optimization techniques for parameter search, such as hill climbing or gradient descent, are employed. We attack the parameter estimation problem using a suite of learning automata (LA) in an N-player stochastic game framework. We use the learning automata primarily for three reasons:

- It has been proven that a team of learning automata will converge to the global optimum [9] with the right learning rate.
- The N-player game model can easily accommodate, in the search process, interactions between different parameter choices. It accounts for the dependence of one parameter choice on another parameters to guide the search process.
- Although in this paper we use the automata team as an offline learning module, the team can also be used online. The automata team can incorporate new training data as they arrive. They are capable of incremental learning. It is possible to use such a team of automata to continuously enhance the performance of a vision algorithm with each run. However, this aspect is still a part of future work.

4.1 What Is a Learning Automaton?

A learning automaton (LA) is an algorithm that adaptively chooses from a set of possible actions on a random environment so as to maximize expected feedback. (The reader is referred to [45] for an excellent introduction to learning automata.) A learning automaton is coupled with the environment, which in our case is the feature grouping algorithm along with the image set. In response to the chosen action, the environment generates a stochastic output β, which is used by the learning automaton to decide on the next action. The goal is to ultimately choose the action that results in the maximum expected β.

A learning automaton decides on the next action by random sampling based on an action probability vector, $\mathbf{p^k} = \{p_1^k, \cdots, p_r^k\}$, defined over the set of actions, $\{\alpha_1^k, \cdots, \alpha_r^k\}$. In the beginning $p_1^k = \cdots = p_r^k = 1/r$, signifying that each action is equally likely. On receiving a feedback from the environment, this probability vector is updated using a learning algorithm. The exact nature of the updating algorithm varies. However, the common strategy is to increase the probability of the action that generates a favorable β and decrease the probability of the action that generates an unfavorable feedback. The change in the probabilities are such that $\sum_{i=1}^{r} p_i^k = 1$. With each iteration, the entropy of the action probability vector decreases until the probability of the optimal action converges to one. It can be shown that the LA will converge if the statistics of the environment are stationary and the updating functions satisfy some minimal conditions. For the grouping problem, this environmental stationarity assumption implies that the statistics of the image set are stationary or that the images are from one class.

In the present scenario, we associate one learning automaton with one algorithm parameter. The actions correspond to the various values of the parameter. However, the learning automata do not operate independently of each other, but they work as a team to capture the dependence between the parameters. The probabilistic

updating of each automaton takes into account the actions of other automata.

4.2 How Does a Team of Automata Operate?

We map the parameter estimation problem into an N-player game by associating with each parameter a player who has to choose from a range of parameter values (Fig. 5). We quantize each parameter into r levels ($r = 10$ in our experiments) so that each player has a finite set of "moves" or "plays" to make or "actions" to choose from. Let us denote this choice set for the kth player by $\alpha^k = \{\alpha_1^k, \cdots, \alpha_r^k\}$. Each player randomly makes a move, which forms part of the chosen parameter combination. This parameter combination extracts a reward from the environment, viz. from the grouping algorithm and the image set. To generate a reward, the environment applies the grouping algorithm with the parameter combination on the training image set and computes the average performance based on the measures discussed in Section 4.7. This reward is returned to each player as feedback. Based on this common feedback, β, each player chooses its next move. The objective of each player is to choose an action so as to maximize this feedback over time.

The updating strategy maintains estimates of the expected feedback for every combination of moves as a multidimensional matrix called the (estimated) game matrix $\hat{\mathbf{D}} = \{\hat{D}_{i_1,\cdots,i_N}\}$, whose dimension is $r \times r \times \cdots r(N\text{times})$. Let the i_kth possible action of the kth automaton be denoted by $\alpha_{i_k}^k$. Then, $\hat{D}_{i_1,\cdots,i_N}$ stores the average reward for the play $\{\alpha_{i_1}^1, \cdots, \alpha_{i_N}^N\}$. It might appear that we would require a significant amount of memory to store this game matrix estimate. However, in practice, this estimated game matrix is sparse and can be efficiently stored. The number of nonzero entries will be, at most, equal to the number of iterations, which is typically far less than the maximum possible size of the matrix. Each player updates its action probability vector based on this estimated game matrix.

This estimated game matrix $\hat{\mathbf{D}}$ is really an approximation of the underlying game matrix governing the game $\mathbf{D} = \{D_{i_1,\cdots,i_N}\}$, which is composed of the *expected* feedbacks for every combination of moves, $\{i_1, \cdots, i_N\}$,

$$D_{i_1,\cdots,i_N} = E(\beta | \alpha^1 = \alpha_{i_1}^1, \cdots, \alpha^N = \alpha_{i_N}^N), \tag{8}$$

where $\alpha_{i_k}^k$ is the i_kth action of the kth automaton.[1] We, of course, do not know the game matrix a priori. However, if the statistics of this game matrix are stationary, then we can design algorithms based on its estimates so that the game converges to the global optimum. In our case, this environmental stationarity assumption implies that the statistics of the image set are stationary or that the images are from one class.

Let E_j^k denote the maximum expected reward to the kth player when it plays α_j^k. We denote the vector composed of these rewards by $\mathbf{E^k} = \{E_j^k\}$ and term it the *individual game vector*. This individual game vector can be looked upon as a projection of the game matrix. Thus,

$$E_j^k = \max_{\{i_s, s \neq k\}} D_{i_1,\cdots,i_{k\ 1},j,i_{k+1},\cdots,i_N}. \tag{9}$$

The term $\{i_s, s \neq k\}$ denotes the set of possible combinations of moves of all the players except for the kth player. Let the globally optimal play for the the kth player be $\alpha_{m_k}^k$, then

1. We use $\hat{X}$ to denote estimate of the random variable X.

$$\max_{\{i_s\}} D_{i_1,\cdots,i_N} = \max_j E_j^k = E_{m_k}^k \qquad \text{for all } k = 1, \cdots, N.$$

The term $\{i_s\}$ denotes the set of possible combinations of moves of all the players. So, each player can reach the globally optimum point by choosing the plays according the individual game vector, $\mathbf{E^k} = \{E_j^k\}$. Of course, in practice, we only have an estimate of this vector, $\hat{\mathbf{E}}^{\mathbf{k}}$, which is computed from the estimated game matrix $\hat{D}(i_1, \cdots, i_N)$.

$$\hat{E}_j^k = \max_{\{i_s, s \neq k\}} \hat{D}(i_1, \cdots, i_{k\ 1}, j, i_{k+1}, \cdots, i_N). \tag{11}$$

Based on the environment feedback, β, and this individual game vector estimate, $\hat{E}_j^k$, each automaton chooses its actions. Let us denote the iteration number by n. We will denote the value of a variable at the nth iteration by appending n as an argument to its symbol. Thus, $\beta(n)$ denotes the environmental feedback at the nth iteration and $\alpha^k(n)$ denotes the play of the kth automaton at the nth iteration. Let $\{\alpha_{i_1}^1, \cdots, \alpha_{i_k}^k, \cdots, \alpha_{i_N}^N\}$ be the play of the N automata at the nth iteration. Following Thathachar and Sastry [9], we update the action probability vectors and other estimates, which essentially consists of two steps.

First is the update of the probability vector $\mathbf{p^k(n+1)}$. We increase the probabilities of the plays with estimates of the individual maximum feedback, $\hat{E}_j^k(n)$ that are larger than the feedback for the play chosen at the nth iteration $\hat{E}_{i_k}^k(n)$. We decrease probabilities of the other plays so that the total sum of probabilities remains one. Mathematically,

$$\begin{aligned}
&p_j^k(n+1) = \\
&p_j^k(n) \quad \mu\Big(\hat{E}_{i_k}^k \quad \hat{E}_j^k\Big) p_j^k(n) \qquad \text{if } (j \neq i_k) \text{ and } (\hat{E}_{i_k}^k \geq \hat{E}_j^k) \\
&\quad = p_j^k(n) + \mu\Big(\hat{E}_j^k \quad \hat{E}_{i_k}^k\Big) \\
&(1 \quad p_j^k(n)) \tfrac{p_{i_k}^k(n)}{r\ 1} \qquad \text{if } (j \neq i_k) \text{ and } (\hat{E}_{i_k}^k < \hat{E}_j^k) \\
&\quad = 1 \quad \sum_{j \neq i_k} p_j^k(n+1) \quad \text{if } (j = i_k).
\end{aligned} \tag{12}$$

Recall, at start $p_j^k(0) = \frac{1}{r}$, with all actions being equally likely. The extent of change in the action probability vector at each iteration depends on 1) μ—the learning rate, 2) the difference in the maximum feedback for an action and the action chosen at the nth iteration, and 3) the probability of each action. For cases where different actions result in drastically different feedbacks, the learning would be faster than for a case where different actions result in almost similar feedbacks. Also, in the beginning, when all the action probabilities are small, learning is slow, which allows the process to explore new actions.

The second step consists of updating the individual game vector estimates, $\hat{E}_j^k(n+1)$. We use two intermediate variables R and Z to first compute the game matrix estimate, $\hat{D}_{j_1,\cdots,j_N}(n+1)$, which, in turn, determines $\hat{E}_j^k(n+1)$. At each step, we need to update only one entry of the game matrix, namely, the entry corresponding to the play at the nth iteration, $\hat{D}_{i_1,\cdots,i_N}(n+1)$. Mathematically,

$$\begin{aligned} R_{i_1,\cdots,i_N}(n+1) &= R_{i_1,\cdots,i_N}(n) + \beta(n) \\ Z_{i_1,\cdots,i_N}(n+1) &= Z_{i_1,\cdots,i_N}(n) + 1 \\ \hat{D}_{j_1,\cdots,j_N}(n+1) &= \frac{R_{j_1,\cdots,j_N}(n+1)}{Z_{j_1,\cdots,j_N}(n+1)} \quad j_k \in \{1,\cdots,r\}, 1 \le k \le N \\ \hat{E}^k_{i_k}(n+1) &= \max\{\hat{E}^k_{i_k}(n), \hat{D}_{i_1,\cdots,i_N}(n+1). \end{aligned} \tag{13}$$

At start, $R_{i_1,\cdots,i_N}(0) = 0$, $Z_{i_1,\cdots,i_N}(0) = 0$, and $\hat{E}^k_{i_k}(0) = 0$ for all $\{i_1,\cdots,i_N\}$ and k.

4.3 Choice of the Learn Rate

The outlined learning algorithm is optimal and can be theoretically shown to converge to the global optimum point with the right choice of learning parameter μ (see [45], [9] for details). The rate of convergence is inversely related to μ. If one chooses a very small μ, then the learning algorithm is very slow, but the probability of finding the global optimum is high. A large μ implies faster convergence, but does not guarantee a global optimal point. In our experiments, we start with $\mu = 0$ for the first 100 iterations to let the algorithm form a starting estimate of the game matrix and then, for later iterations, μ is set to 0.1. Observe that the effect of this learning rate on the learning process is somewhat different from that in other types of learning strategies. Fixing μ to a constant does not imply a fixed *effective* learning rate. Recall from (12), the amount of change at each iteration is dependent on two other factors: differences in feedback for different actions and the action probabilities at that iteration. In fact, the amount of updating is small toward the beginning iterations and gradually increases, even for constant μ.

4.4 Stopping Conditions

Traditionally, the stopping conditions are cast in terms of the maximum action probability over all the players. Ideally, the final action probability vector a of player should have a probability of one corresponding to the optimal action and zero for the others. For the parameter selection case, we are not interested in *the* optimal parameter combination but in a set of good parameter combinations. So, *we stop when the automata team does not find any new parameter combination that is better than the ones already found for a number (typically 200) of consecutive iterations.* This condition is easy to detect: If no $\hat{E}^k_{i_k}(n+1)$, for $1 \le k \le N$, is updated at an iteration (13), then it implies that no new parameter combination that is better than the previous ones has been found at that iteration. We keep track of the number of consecutive iterations for which this is true.

4.5 The Learned Parameter Combinations

The set of good parameter combinations is selected from the sequence of actions, which we term the *trace* or the *run* chosen by the team of learning automata. From this trace (or run), we choose the k-best parameter combinations for each image. Remember that, at each iteration, we have the individual performance of the grouping algorithm with the chosen parameter combination on all training images. The k-best parameter combinations constitute the learned set of parameters. Note, this strategy for learning good parameter sets is faster than training on each image and then choosing the k-best parameter combinations. We explore the parameter space guided by the average performance performance of the grouping algorithm over the input images, but the final selection of parameters is done based on individual images.

4.6 How Is the Performance Feedback, β, Computed?

The learning automata updates its action probability vector based on feedback from the environment. The environment in the present case consists of an edge detector, a contour segmentor, and a grouping algorithm, along with the training image set. At each iteration, the environment applies the grouping algorithms on the training image set, with parameters determined as per the LA actions. The average performance forms the feedback to the LA team.

The feedback measure, which captures the performance of the grouping algorithm on an image, is a combination of three terms. The first term represents the expected speed of object recognition from the groups generated. The second term represents the confidence in the recognition results. The third term is dependent on the false alarm rate. These measures have been proposed in [46] as a part of a set of five performance measures for grouping modules. The rationale behind these measures is reproduced here for completeness.

We motivate the estimation of the speed of recognition from a constrained search point of view, based on Grimson's [2] complexity analysis of object recognition using imperfect groups in the presence of clutter. Let N_G denote the number of features in a detected group, N_O denote the number of model features, and $N_{G\cap O}$ denote the number of group features that lie on the model. Assuming that all features are equally important, Grimson showed that the expected search, W_{term}, is essentially polynomial if we terminate when the number of matched features equal some predetermined threshold, t. The exact expression is given by:

$$\begin{aligned} N_O N_G \frac{N_G}{N_{G\cap O}} &\le W_{term} \\ &\le t N_O N_G \frac{N_G}{N_{G\cap O}} \left(1 + \frac{\kappa^2}{N_O}\right)^2 \left(\kappa^2 \frac{N_G}{N_O}\right)^{\lfloor (N_G/N_O)\kappa^2 \ \ 1 \rfloor}. \end{aligned} \tag{14}$$

The constant κ is small and is typically equal to $0.2\frac{P}{D}$, where P is the total perimeter of the object and D is the image dimension. If $\frac{N_G}{N_O} \le 50\frac{D^2}{P^2}$, then the search is essentially quartic. In the worst case, $P \approx D$ and the requirement is $N_G \le 50 N_O$, a very liberal requirement. The term in (14), which depends on the quality of the group, is the ratio $\frac{N_G}{N_{G\cap O}}$. This constitutes the first part of performance measure.

$$P_{time}(G,O) = \frac{N_{G\cap O}}{N_G}. \tag{15}$$

This measure ranges from zero to one and should be as large as possible to minimize the amount of search.

The quality of the terminated constrained search will be proportional to the threshold, t, which is the number of model features explained by the group. Thus, $\frac{t}{N_O}$ captures

the model to group match quality. Using this expression, coupled with the fact that the termination threshold t is less the number of common features, $N_{G\cap O}$, between the model and the group, we suggest the second part of the performance measure to be:

$$P_{qual}(G,O) = \frac{N_{G\cap O}}{N_O}. \tag{16}$$

This measure ranges from zero to one and should be large to ensure high confidence recognition. Large values of this measure will help discriminate between models, and thus, boost the accuracy of recognition.

The performance measures in (15) and (16) need the availability of object models or, at least, estimates of the numbers of features (N_O) in the models. Since we are concerned with an edge-based recognition strategy, the features of interest are edge pixels. Manual construction of 3D models is cumbersome and renders the performance analysis almost intractable for real domains. We circumvent this problem using manual outlines of object boundaries in each image. Given an edge image, the collection of edge pixels close to the manual outline represents the perfect grouping of features in an image. For 2D model-based recognition scenarios, such as those that are view-based, the number of edge points will provide a good estimate of the number of model features. For 3D model-based recognition scenarios, we expect the number of edge features in an image to be proportional to the actual number of 3D model features (on average).

Let the grouping algorithm generate N groups, $G_1, \cdots, G_N$ for an image with M objects, $O_1, \cdots, O_M$. The false alarm groups are defined to be groups that do not overlap with any object of interest. For each pair of group, G_i, and image object, O_j, that overlap, we compute $P_{time}(G_i, O_j)$ and $P_{qual}(G_i, O_j)$. Let the total number of overlaps be $N_{overlaps}$, which can be anywhere between 0 and NM. We then combine these measures as follows to generate the performance measure, β:

$$\beta = \sqrt{\left(\frac{\sum_{ij} P_{time}(G_i,O_j)}{N_{overlaps}}\right)\left(\frac{\sum_{ij} P_{qual}(G_i,O_j)}{N_{overlaps}}\right)\left(1 \quad \frac{N_{false}}{N}\right)}, \tag{17}$$

where N_{false} is the number of groups that do not overlap with any object. Notice that the measure β is inversely related to the number of false alarms and that it ranges from zero to one, with one being the desired value. Other combinations of the measures might be desirable based on the task at hand; however, this combination suffices for the illustration of the essential ideas.

5 Results and Analyses

We present thorough analyses and evaluation of the performance of both the spectral grouping and the learning strategies. First, we investigate the performance of the spectral grouping algorithm on a variety of real images. Second, we compare the performance of the particular spectral partitioning technique that we use for grouping with normalized cut-based spectral graph partitioning suggested elsewhere [22]. Third, we demonstrate the ability to learn to group features of a *single* object *type*, e.g., airplanes, in the presence of different types of background clutter. Fourth, we demonstrate the ability to learn to form groups that correspond to *several* object types in a particular domain, e.g., aerial.

5.1 General Performance of Spectral Grouping

Fig. 6 shows some sample results of the spectral grouping algorithm on a variety of real images, namely, oblique aerial views, top aerial views, and outdoor images of manmade and natural objects. The left column shows the gray-level images with the ground truth objects (manually) outlined in different colors. The middle column shows the input edge features that are to be grouped. The rightmost column shows the different detected groups. Each group is colored differently. Note how the algorithm is able to pick out salient features in the scene even in the presence of significant image clutter.

In the multistory building of Fig. 6a, the grouping algorithm picked out the parallel structures corresponding to the windows. The image in Fig. 6d has significant clutter, but the algorithm is able to pick out the salient groups corresponding to the major objects in the scene. The algorithm is also able to resolve the buildings (circular and rectangular) shown in Fig. 6g. Figs. 6j-6o demonstrate the applicability of the grouping algorithm to different domains and to images that are close views of man-made and natural objects. In spite of the large image clutter in the mailbox image of Fig. 6j, the groups corresponding to the major structures in the scene are found. In Fig. 6m, the tiger is segmented out from the scene (red group). The algorithm is able to handle curved edge segments.

Parameter values for good grouping. The results attest to the flexible nature of the grouping algorithm. It is capable of producing good results with complex images even in the presence of significant image clutter. The various input parameters make the grouping algorithm very flexible. We can tune the relative importance of different relations to suit a particular domain or object. To study the effect of the choice of the grouping parameters on performance, we employed the team of learning automata to learn a set of 100 parameter sets with the best performance, as measured by β (17), on images such as those shown in Fig. 6. We make the following observations based on the distribution of these good parameter sets, which are plotted as normalized histograms in Fig. 7.

1. The priors that result in good performance for each of the salient 2-ary relationships differ from image to image. Both high and low prior choices result in good performance, depending on the image.
2. For most of the images, a prior of 0.5 or more for region similarity results in superior performance. This suggests that photometric attributes play a significant role. This is in agreement with the Gestalt psychologists recent suggestion of the importance of common region as a grouping factor [3]. However, so far, photometric attributes have not played a significant part in extended feature grouping algorithms in computer vision.
3. Continuity between edge segments is not always important for figure ground segmentation. For some of the images (e.g., Fig. 6b), low priors for this

Fig. 6. The performance of the spectral grouping algorithm. The images in the middle column show the image edge features that are grouped. The right column shows the feature groupings found using the spectral method. Each cluster is shown in a single color.

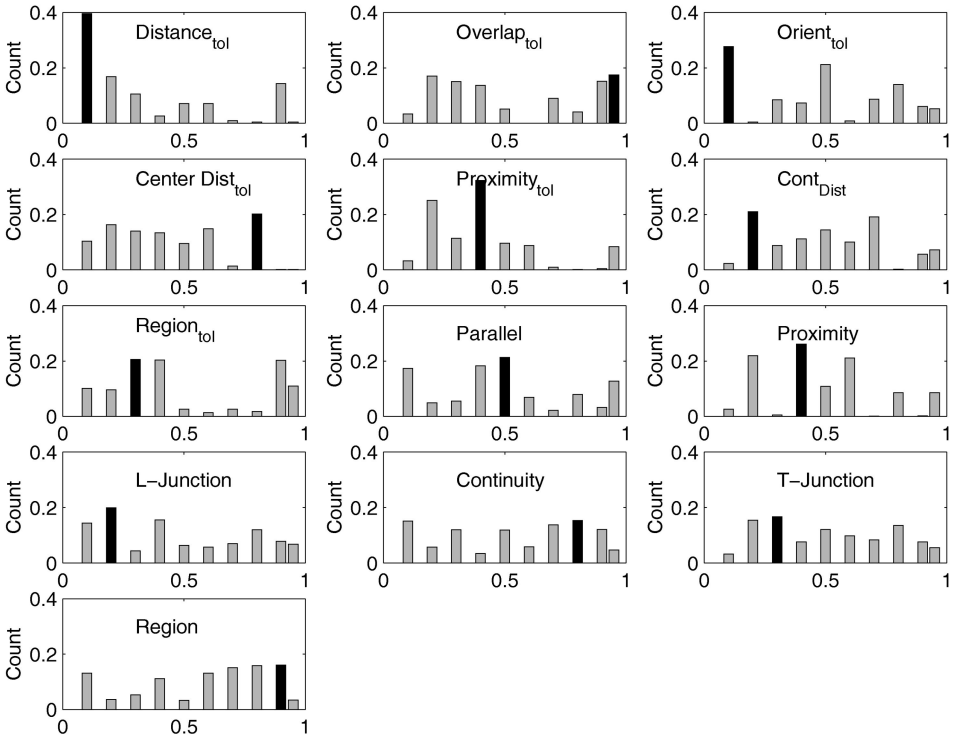


Fig. 7. Normalized histogram of the parameter values that result in good performance over images like that shown in Fig. 6. Each bar plot corresponds to a parameter, as labeled. The horizontal axis of each plot corresponds to the 10 different values for each parameter. The highest bar in each plot is shown darker than the rest.

relation result in good performance. We should point out that we are referring to continuity at an extended edge segment level and not at a pixel level. The learning process might choose a low continuity prior even for images with seemingly long continuous edge features. This can happen if long continuous chains of edge pixels do not get fragmented as a result of the edge detection and the contour segmentation processes and, thus, the extracted edge segments are long and exhibit low continuity between them.

4. For most of the images, low priors for proximity and T-junctions result in good performance, which suggests that these relations might not be important for every image.

5.2 Comparison with Normalized Cut Partitioning

The graph bipartitioning algorithm that is at the heart of the grouping algorithm can be shown to minimize the weight between the partitions normalized by the size of the partitions, thus the name average cut. One can also define bipartitions based on minimizing the weight between the

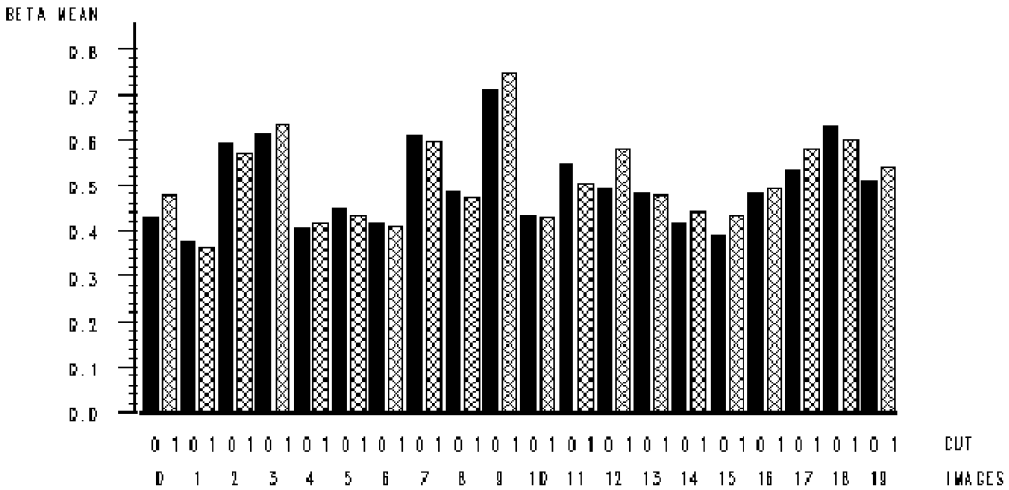


Fig. 8. Variation of performance of normalized cut and average cut-based grouping on a set of 20 images. The solid bars correspond to the best performance achieved on an image for average cut-based grouping. The shaded bars correspond to the performance of the normalized cut-based grouping on the corresponding image.

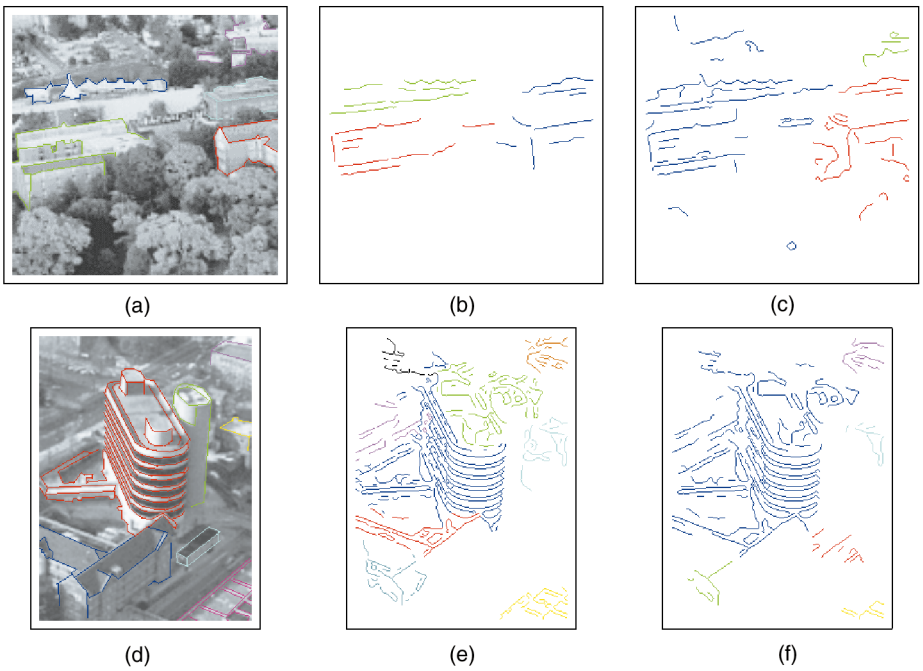
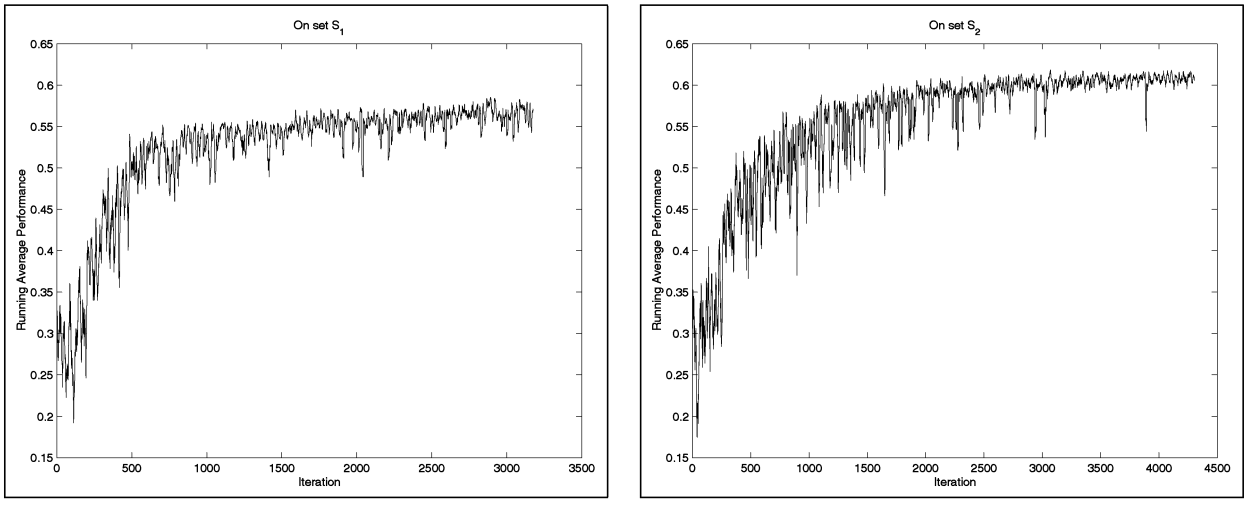


Fig. 9. (a) The image on which the performance of average cut-based grouping, shown in (b), is most superior to normalized cut-based grouping, shown in (c). (d) The image on which the performance of average cut-based grouping, shown in (e), is most inferior to normalized cut-based grouping, shown in (f).

partitions, normalized by the total connection within the partitions—normalized cut [22]. Here, we investigate the need for normalized cut, with its associated larger computational burden, as opposed to average cut.

We compared average and normalized cut-based grouping on a set of 20 aerial images. On each of these images, we used the team of learning automata to sample the parameter space to obtain the *best* performance for normalized and average cut-based groupings. We considered two different learning runs or traces of the automata team. Fig. 8 shows the variation of *best* performance of the normalized cut and average cut. The solid bars correspond to the best performance achieved on an image for average cut-based grouping. The shaded bars correspond to the performance of the normalized cut-based grouping on the corresponding image. It is evident from the plot that the performance of the average cut and the normalized cut-based groupings are similar. For some images, average cut is better and, for some, normalized cut is better. Fig. 9a shows the image on which the performance of average cut-based grouping, which is shown in Fig. 9b, is most superior to normalized cut-based grouping, which is shown in Fig. 9c. Conversely, Fig. 9d is the image on which the performance of average cut-based grouping (Fig. 9e) is most inferior to normalized



(a) (b)

Fig. 10. Typical iteration traces of the learning automata team. The plot in (a) corresponds to learning on set S_1 of airplane images and that in (b) corresponds to the set S_2. The vertical axis corresponds to the running average feedback (β) over last 10 iterations.

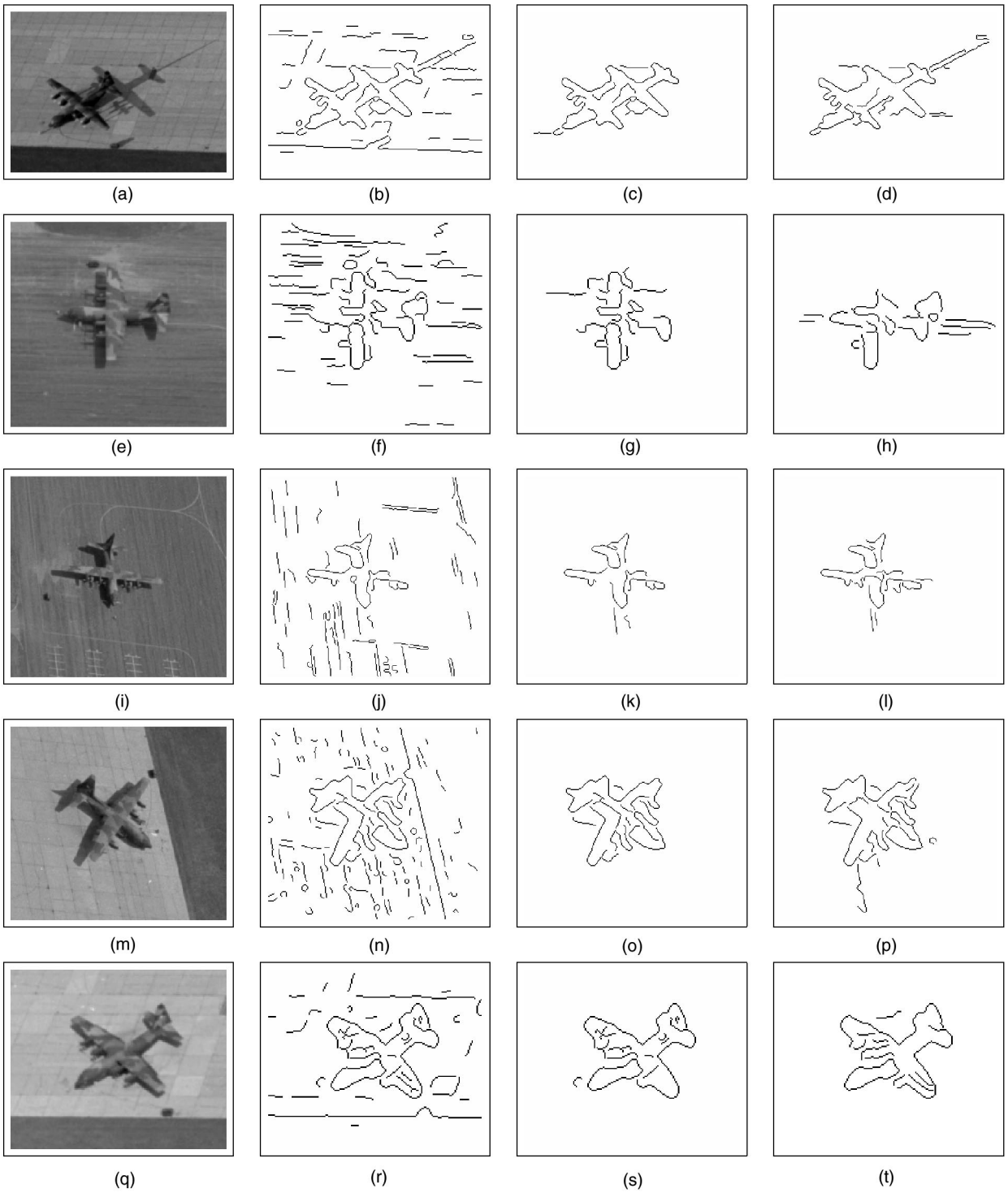


Fig. 11. Representative images from the set of 40 airplane images. The images in the second column show the edge features that are the input of the grouping algorithm. Each image in the third column shows the output groups with the best parameters combination obtained by training on the set of 20 images that *includes* the corresponding image (on the left). Each image in the fourth column image shows the groups with the best parameter combination obtained by training on the set that *does not include* the corresponding gray-level image.

cut (Fig. 9f) based grouping. While interpreting the images, we should remember that the performance measure decreases with presence of false alarm groups, i.e., groups that do not overlap with any objects.

Statistically, the variation of performance between the different cut-based groupings is *not* significant. The non-parametric Wilcoxon test, which is based on ranks, shows that the distributions of the performance of the average and the normalized cut are statistically from the same population

$$(\text{Rank Sum} = 395.0, Z = \quad .392281, \text{Prob} \geq |Z| = 0.6949).$$

Recall that nonparametric tests do not make assumptions about the forms of the distributions and are also good for comparing distributions with small number of samples.

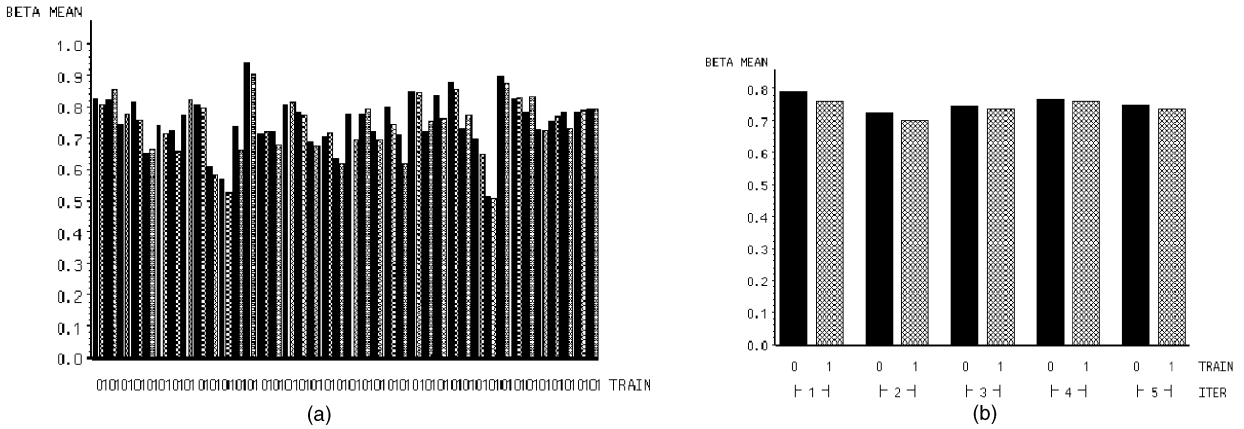


Fig. 12. (a) Variation of train and test performance for different airplane images. The solid bars correspond to the best performance achieved on an image when it was included in the training set. The shaded bars represent the best performance achieved on an image when it was *not* included in the training set. (b) Variation of average training and testing performance with different runs. The solid bars correspond to the best train performance as averaged over the set of images for one learning run. The shaded bars correspond to the best test performance as averaged over the set of images for one learning run.

Thus, our results indicate that average cut bipartitioning is sufficient for grouping extended edge features.

5.3 Adaption of the Grouping Algorithm to Object Types

Can the team of learning automata adapt the grouping process to segregate a particular object *type* from its natural background contexts? To study this, we selected the class of airplanes as the object type. The different types of airplanes could be in different background contexts, such as a tarmac or grass fields, and also at different orientations. We selected 40 such aerial images, with different lighting conditions, viewpoints, and scales. The left column in Fig. 11 show samples of these images. The images, which are of different sizes, are printed to occupy the same size on paper.

We separated the 40 images into two sets, S_1 and S_2, so that we could train on one set and test with the other. In the training phase, the team of learning automata sampled the parameter space, first using the image set S_1 and then S_2, such that the average performance was maximized. Fig. 10 shows two typical traces, one for image set S_1 and the other for set S_2. Note how the average feedback quickly converges in about 3,000 iterations. Compare this with the size of the search space, which is 10^{21}; there are 21 parameters, including the edge detector and contour segmentation parameters and each parameter can take 10 possible values. From the sampling trace, we composed a set of 100 good parameter sets using five best parameter combinations for each image in S_1 (or S_2). The learned 100 parameter combinations for S_1 was applied on S_2, and vice versa, to obtain the test performances. Thus, for each image, we have the best performance that can be achieved by training on the set containing it—the train performance—and the best performance using the parameters learned on the set *not* including the image—the test performance.

The images in the third column of Fig. 11 show the best train performance on the images in the first column. The images in the fourth column show, the best test performance. The second column of images show the input edge features. Note the similarity of the train and test groups. This attests to the good performance of the learning algorithm. It is also interesting to note how the group corresponding to the airplane is separated from other features in an image. The background feature statistics vary from image to image. In some images, the background is more organized than in others. There are also strong and long edge features in the background that cannot be eliminated by simple edge thresholding. In fact, the edge images shown in the second column are the best possible edges that can be obtained by changing the edge scale and

TABLE 3
ANOVA of the Learning Performance on the Aerial Images of Planes

Analysis of learning performance on the airplane images					
Source	DF	SS	F-Value	P-value	
Run	4	0.1798	14.43	0.0001	Significant
Train/Test	1	0.0226	7.26	0.0078	Significant
Image	39	2.8792	23.70	0.0001	Significant
Run × Train/Test	4	0.0078	0.63	0.6430	Not Significant
Run × Image	156	0.4780	0.98	0.5404	Not Significant
Train/Test× Image	39	0.1314	1.08	0.3586	Not Significant

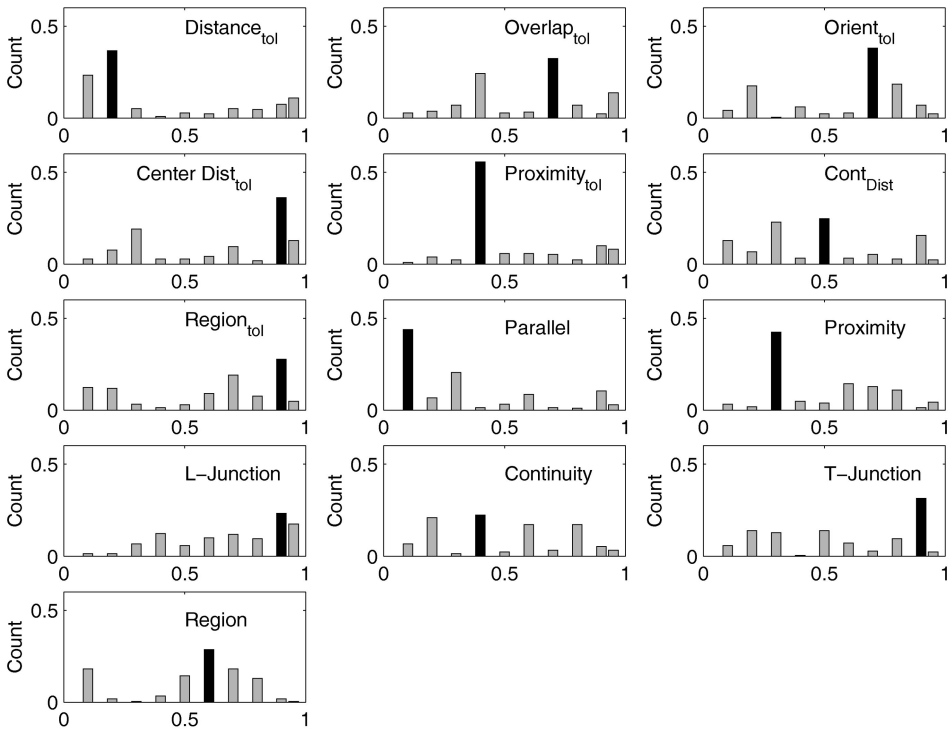


Fig. 13. Normalized histogram of the parameter values that result in good performance for segmenting planes from aerial views. Each bar plot corresponds to a parameter, as labeled. The horizontal axis of each plot corresponds to the 10 different values for each parameter. The highest bar in each plot is shown darker than the rest.

thresholds and, even, the contour segmentation parameters. Recall, that parameters of the edge detection and contour segmentation are also part of the learning process.

Statistical Analysis. With regards to the performance of the learning algorithm, we consider the following questions: 1) Does the observed train and test performance depend on the particular image? 2) Does the observed train and test performance depend on different runs of the learning automata team? Fig. 12a plots the *best* train and *best* test performance for each of the 40 images. The black bars correspond to the best train performances and the best test performances are denoted by shaded bars. For each image, the train and test performance are close to each other with a mean difference of 4 percent, however, the maximum achievable performance does vary from image to image. Fig. 12b plots the train and test performance, as averaged over the 40 images, for five different runs of the learning automata team. The solid bars correspond to average train performance and the shaded bars correspond to average test performance, over the image set. As expected, due to the stochastic nature of the learning algorithm, there is variation between different runs; the mean difference between average training performance is 8 percent. However, the relative difference between overall test and train performance from run to run is small, at around 2 percent.

Although Fig. 12 gives us a visual feel for the robustness of the learning algorithm, it does not quantify the statistical significances of the observations. For statistical analysis, we employed the Analysis of Variance (ANOVA) technique, which can assess the statistical significance of the effect of different factors, and their interactions on the overall performance variation. The main factors that can effect the grouping performance in our case are three: 1) the different learning runs, 2) whether it is train or test performance, and 3) the images. ANOVA can compute the significance of the performance variations not only due to individual factors, but also due to their interactions. Thus, we can answer questions such as does the train and test performance interact with images or is the variation of train and test performance dependent on the images (Train/Test $\times$ Image)? Is the interaction of train and test performance and different learning runs significant (Run $\times$ Train/Test)? Is performance on an image dependent on the particular learning run (Run $\times$ Image)?

Table 3 lists the ANOVA results. From the results, we can see that the variations due to the three main factors are significant; however, their interactions are not significant. Thus, the train and test performance differences that we see in Fig. 12 are statistically significant. This is not unusual. It is indeed rare that the train and test performance is the same for a learning algorithm. Typically, the test performance is expected to be lower than train performance. What is of interest is the extent of the difference which, in the present case, is small—about 4 percent difference. Similarly, although the variation in performance with respect to the particular stochastic run is significant, it is small—the mean difference is about 8 percent. On the contrary, the performance difference between image is not only significant, but is also not small—the mean difference is about 30 percent. This attests to the variety of the image set—it is not homogeneous.

From Table 3, we also see that the interactions are not significant, which implies that we can claim that 1) the observed train and test performance is *not* dependent on the images (Train/Test $\times$ Image), 2) the observed train and test performance is *not* dependent on the stochastic sampling

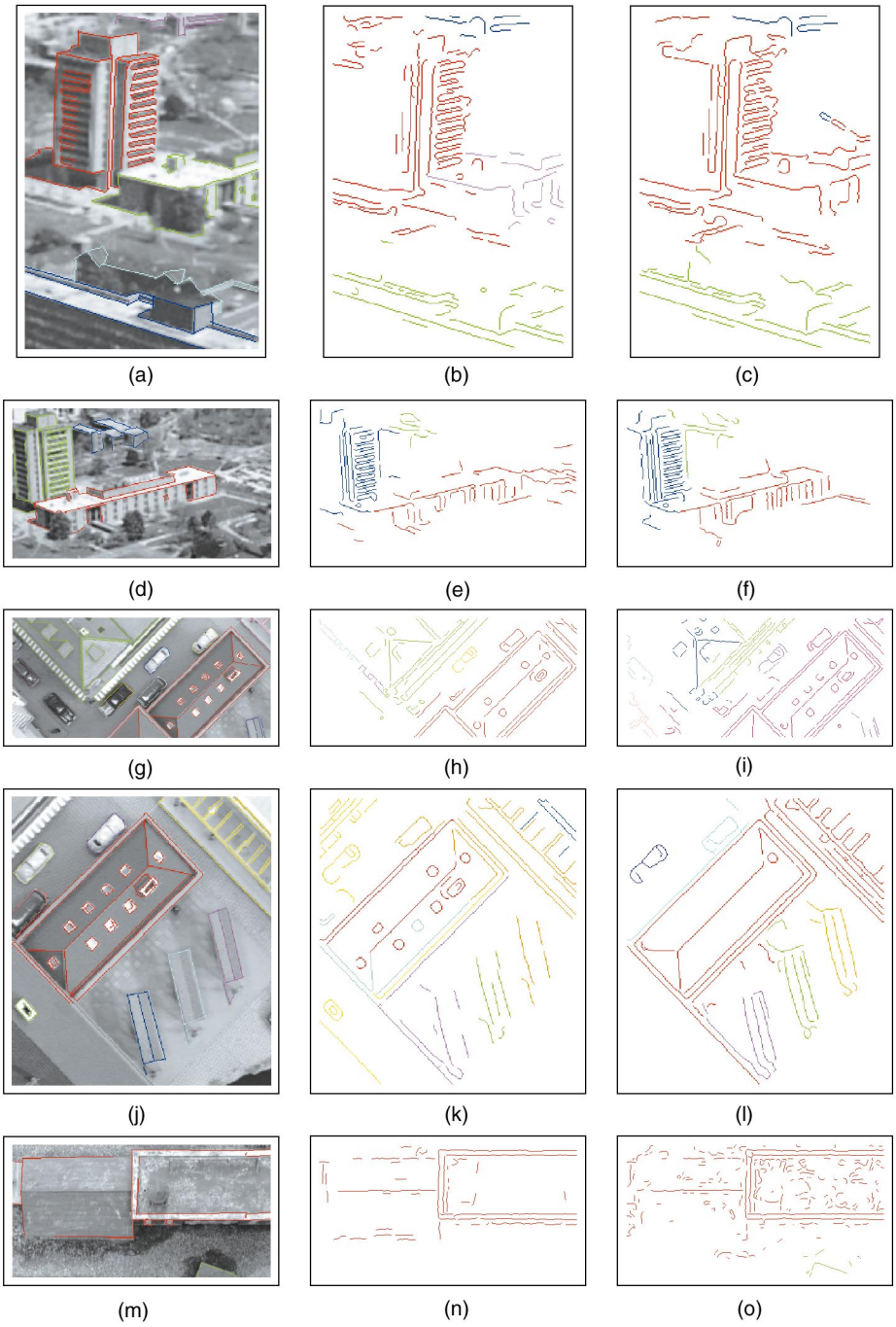


Fig. 14. Representative images from the set of 20 aerial images. Each image in the second column shows the output groups with the parameters obtained by training on the set of 20 images that *includes* the corresponding image (on the left). Each image in the third column image shows the groups with the parameter obtained by training on the set that *does not include* the corresponding gray-level image.

runs (Run $\times$ Train/Test), and 3) the observed performance on an image is *not* dependent on the particular stochastic run (Run $\times$ Image).

The parameter choices. Fig. 13 shows the normalized histograms of the parameter choices that compose the set of 100 learned parameter sets. The plots with subscripted titles correspond to the grouping parameter tolerances and the other six plots correspond to the priors for the salient relationships, i.e., parallel, proximity, L-junction, Continuity, T-junction, and Region. The modes of the distributions are marked with dark bars. These distributions are less dispersed than the one for the set of general images in Fig. 7, which can be attributed to the similar nature of the airplane images. We can also see that L-junction, T-junction, and region similarity play a greater role in segmenting out the plane from the background than do parallelism, proximity, or even continuity. This is due to the fact that the latter three relationships are sometimes present in the background to a larger extent than in the object itself, hence, they are bad indicators for figure-ground segmentation in this context.

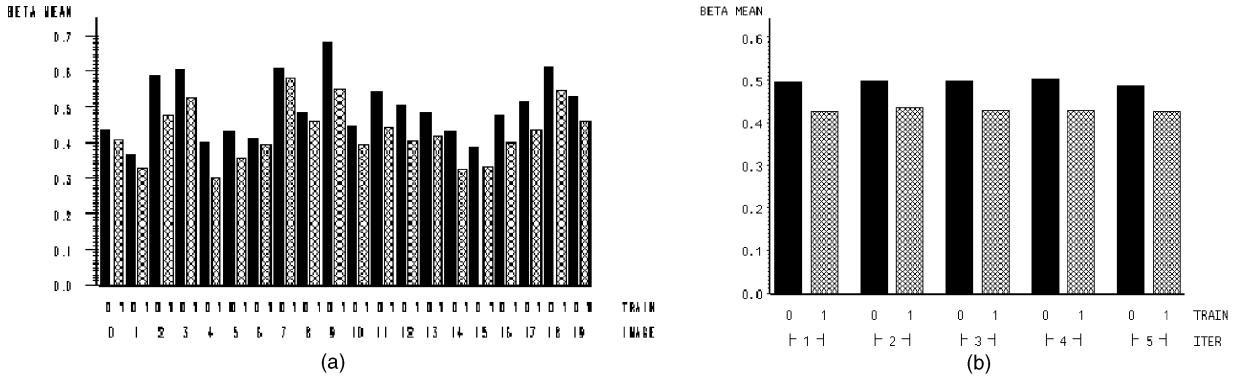


Fig. 15. (a) Variation of train and test performance on the aerial images. The solid bars correspond to the best performance achieved on an image when it was included in the training set. The shaded bars correspond to the best performance achieved on an image when it was *not* included in the training set. (b) Variation of training and testing performance with different runs. The solid bars correspond to the best train performance as averaged over the set of images for one learning run. The shaded bars correspond to the best test performance as averaged over the set of images for one learning run.

5.4 Adaptation of the Grouping Algorithm to a Domain

Is it possible to adapt the grouping algorithm to a domain and not just to a particular object type, as we have seen in Section 5.3? Specifically, we investigate if it is possible to achieve good learning performance on a general class of images such as aerial images. We concentrate on a set of 20 images, some samples of which are shown in the left column of Fig. 14. The ground truth object (manual) outlines are shown overlaid on the gray-level image. As before, we separated these 20 images into two sets, S_1 and S_2. We trained on one set and tested with the other. In the training phase, the team of learning automata sampled the parameter space using S_1 (and S_2) such that the average performance was maximized. From the learning trace, we composed a set of 100 good parameter sets using the 10 best parameter combinations for each of the 10 images in S_1 (or S_2). The learned 100 parameter combinations for S_1 were applied on S_2, and vice versa, to obtain the best test performances on each image. Thus, for each image, we have the best performance that can be achieved by training on the set containing it—the train performance—and the best performance using the parameters learned on the set *not* including the image—the test performance. The images in the second column of Fig. 14 show train performance. The images in the third column show the test performance. Note the reasonable similarity of the train and test groups. This attests to the good performance of the learning algorithm.

Statistical Analysis. Fig. 15a plots the *best* train and *best* test performance for each of the 20 images. The black bars correspond to train performances. Test performances are shown using shaded bars. For each image, the test performance is lower than the train performance. The mean difference between train and test performance is 14 percent. This is larger than the differences observed for learning a single object type, which is to be expected since we are trying to generalize across a domain rather than across just a single object type. As before, the overall group quality differs from image to image; there is 44 percent variation across images. Fig. 12b plots the train and test performance as averaged over the 20 images for five different runs of the learning automata team sampling. As before, the solid bars correspond to train performance and shaded bars correspond to test performance. The mean difference in train performance from run to run is about 3 percent. However, the difference between train and test performance, which is about 14 percent, does not seem to vary with runs.

To quantify the statistical significance of the observed differences, we employed the Analysis of Variance (ANOVA) technique. The factors that can give rise to overall variations are the same as before, namely, 1) learning runs, 2) train or test case, 3) images, and their interactions, (Train/Test $\times$ Image), (Run $\times$ Train/Test), and (Run $\times$ Image). Table 4 lists the ANOVA results. From the results, we can see that:

TABLE 4
ANOVA of the Learning Performance on Aerial Images

Analysis of learning performance on the aerial images					
Source	DF	SS	F-Value	P-value	
Run	4	0.0024	1.19	0.3220	Not Significant
Train/Test	1	0.2395	479.10	0.0001	Significant
Image	19	1.2790	134.66	0.0001	Significant
Run $\times$ Train/Test	4	0.0015	0.77	0.5492	Not Significant
Run $\times$ Image	76	0.0396	1.04	0.4299	Not Significant
Train/Test$\times$ Image	19	0.0490	5.16	0.0001	Significant

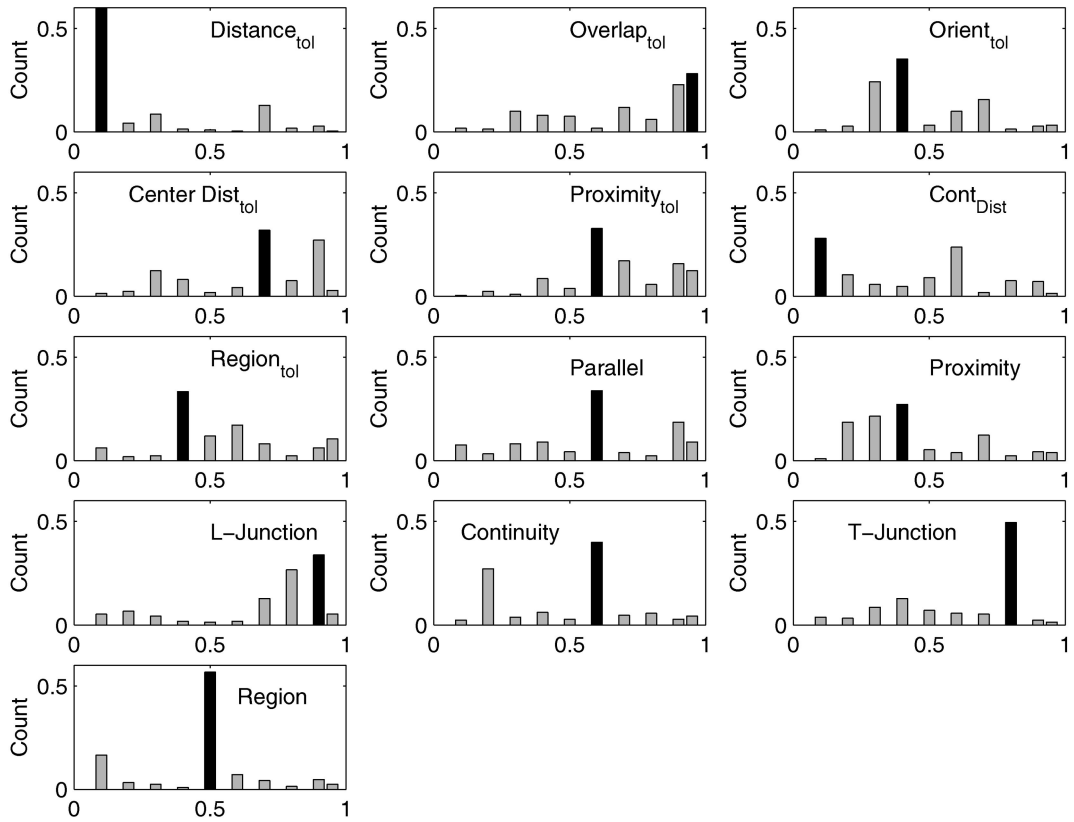


Fig. 16. Normalized histogram of the parameter values that result in good grouping performance for aerial images. Each bar plot corresponds to a parameter, as labeled. The horizontal axis of each plot corresponds to the 10 different values for each parameter. The highest bar in each plot is shown darker than the rest.

1. The train and test performance difference (of about 14 percent mean) that we see in Fig. 12 is statistically significant.
2. Images are a significant source of variation, which attests to the variety of the image set.
3. The performance does not vary significantly between different learning runs. This is desirable, but definitely not typical, for learning based on stochastic samplings. We believe this might be due to the underlying nature of the parameter space, which might have low variations.
4. The observed train and test performance is dependent on the images (Train/Test $\times$ Image). Thus, the relative train and test performance vary for image to image. For some images, the difference between train and test is lower than others. This is due to the larger variety of the images being considered.
5. The observed relative train and test performance is *not* dependent on the learning runs (Run $\times$ Train/Test).
6. The observed performance on an image is *not* dependent on the particular learning run (Run $\times$ Image).

The parameter choices. Fig. 16 shows the normalized histograms of the parameter choices that compose the set of 100 learned parameter sets. The plots with subscripted labels correspond to the grouping parameter tolerances and the other six correspond to the priors for the salient relationships, i.e., parallel, proximity, L-junction, Continuity, T-junction, and Region. The modes of the distributions are marked with dark bars. For aerial images, we see that, except for proximity, all other relationships play an important role in segmenting objects from background. Unlike for the plane images, where parallelism and continuity did not consistently play an important role, here they do play a significant role. This dependence of grouping performance on the relative importance of the relationships is precisely the reason why we need a framework that can adapt the grouping process.

6 Conclusion

We presented a flexible, learnable, perceptual organization framework based on graph partitioning. The graph spectral techniques facilitate the easy consideration of global context in the grouping process. An N-player automata framework learns the grouping algorithm parameters. We demonstrated the performance of the grouping algorithm on a variety of images. Among the interesting conclusions are: 1) It is possible to perform figure-ground segmentation from a set of local salient relations, such as parallelism, continuity, perpendicularity, proximity, and region similarity, each defined over a small number of primitives. 2) The relative importance of the salient relations is dependent on the object or domain of interest. 3) Just geometric relationships are not sufficient for groupings. Photometric attributes, such as region similarity, play a significant role in grouping extended low-level features (see discussion associated with Figs. 7, 13, and 16). We also showed that grouping performance with a different underlying graph

spectral formulation, namely normalized cut, did not result in statistically significant differences.

Extensive statistical analysis of the learning algorithm shows that it is possible to adapt grouping process to single object types (e.g., airplanes) with performances within 4 percent of the best possible performance. We found that the observed learning performance on an image is not dependent on the learning run (or trace). Also, the observed train and test performance differences are independent of the particular image. Furthermore, we demonstrated that it is also possible to learn grouping parameters for a specific image domain (e.g., aerial), with a mean train and test difference of 14 percent. In this case, too, we found that the performance of the learning algorithm is independent of the learning run.

Although we motivated the grouping problem from an object recognition point of view, the grouping output can also be used for other vision tasks, such as to focus attention in a scene. Similarly, the learning algorithm can be used for other vision tasks such as performance characterization. To compare two vision algorithms, we need to first decide on the best parameters on a per image basis or for a group of images. As the number of parameters increases, exhaustive search becomes computationally very expensive. The learning framework in this paper offers an efficient alternative strategy. It can be used to find the best parameter on a per image basis or for a group of images, just by controlling the images that are considered to be part of the environment. The parameter learning framework can also be used to tune parameters of a *network* of vision modules, where the number of parameters is usually large and there are strong interactions between different parameters.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers and Mike Heath for their excellent critique of the earlier draft of the paper, which greatly helped in enhancing the quality of the paper. This work was supported by U.S. National Science Foundation CAREER grant nos. IRI-9501932, IIS-9907141, and EIA-9729904. A preliminary version of this paper was presented at the IEEE Conference on Computer Vision and Pattern Recognition, 1998 [47].

REFERENCES

[1] D.T. Clemens and D.W. Jacobs, "Space and Time Bounds on Indexing 3D Models from 2D Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 10, pp. 1,007-1,017, Oct. 1991.

[2] W.E.L. Grimson, "The Combinatorics of Heuristic Search Termination for Object Recognition in Cluttered Environments," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 9, pp. 920-935, Sept. 1991.

[3] I. Rock and S. Palmer, "The Legacy of Gesalt Psychology," *Scientific Am.,* pp. 84-90, Dec. 1990.

[4] D.G. Lowe, "Three-Dimensional Object Recognition from Single Two-Dimensional Images," *Artificial Intelligence,* vol. 31, pp. 355-395, 1987.

[5] A. Etemadi, J.-P. Schmidt, G. Matas, J. Illingworth, and J. Kittler, "Low-Level Grouping of Straight Line Segments," *Proc. British Machine Vision Conf.,* pp. 119-126, 1991.

[6] D.W. Jacobs, "Robust and Efficient Detection of Salient Convex Groups," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 1, pp. 23-37, Jan. 1996.

[7] G. Roth and M.D. Levine, "Geometric Primitive Extraction Using a Genetic Algorithm," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 1, no. 9, pp. 901-905, Sept. 1994.

[8] R. Mohan and R. Nevatia, "Using Perceptual Organization to Extract 3-D Structures," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 11, no. 11, pp. 1,121-1,139, Nov. 1989.

[9] M.L. Thathachar and P.S. Sastry, "Learning Optimal Discriminant Functions Through a Cooperative Game of Automata," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 17, pp. 73-85, Jan. 1987.

[10] M.S. Lew, T.S. Huang, and K. Wong, "Learning and Feature Selection in Stereo Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 16, no. 9, pp. 869-881, Sept. 1994.

[11] H. Greenspan, R. Goodman, R. Chellappa, and C.H. Anderson, "Learning Texture Discrimination Rules in a Multiresolution System," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 16, no. 9, pp. 894-900, Sept. 1994.

[12] K. Cho and S.M. Dunn, "Learning Shape Classes," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 16, no. 9, pp. 882-887, Sept. 1994.

[13] B. Bhanu, S. Lee, and J. Ming, "Adaptive Image Segmentation Using a Genetic Algorithm," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 25, pp. 1,543-1,567, Dec. 1995.

[14] A.R. Pope and D.G. Lowe, "Learning Probabilistic Appearance Models for Object Recognition," *Early Visual Learning,* S.K. Nayar and T. Poggio, eds., pp. 67-98, Oxford Univ. Press, 1996.

[15] B. Draper, "Modelling Object Recognition as a Markov Decision Process," *Proc. Int'l Conf. Pattern Recognition,* vol. D, pp. 95-99, 1996.

[16] S. Houzelle, T.M. Strat, P. Fua, and M.A. Fischler, "Using Contextual Information to Set Control Parameter of a Vision Process," *Proc. Int'l Conf. Pattern Recognition—Part A,* pp. 830-832, 1994.

[17] V. Murino, G.L. Foresti, and C.S. Regazzonni, "A Distributed Probabilistic System for Adaptive Regulation of Image Processing Parameters," *IEEE Trans. Systems, Man, and Cybernetics—Part B: Cybernetics,* vol. 26, pp. 1-20, Feb. 1996.

[18] V. Murino, G.L. Foresti, and C.S. Regazzoni, "A Belief Based Approach for Adaptive Image Processing," *Int'l J. Pattern Recognition and Artificial Intelligence,* vol. 11, pp. 359-392, May 1997.

[19] J. Peng and B. Bhanu, "Closed Loop Object Recognition Using Reinforcement Learning," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 2, pp. 139-154, Feb. 1998.

[20] Z. Wu and R. Leahy, "An Optimal Graph Theoretic Approach to Data Clustering: Theory and Its Application to Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 15, no. 11, pp. 1,101-1,113, Nov. 1993.

[21] G. Guy and G. Medioni, "Inferring Global Perceptual Contours from Local Features," *Int'l J. Computer Vision,* vol. 20, pp. 113-133, 1996.

[22] J. Shi and J. Malik, "Normalized Cuts and Image Segmenation," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* pp. 731-737, June 1997.

[23] A. Amir and M. Lindenbaum, "A Generic Grouping Algorithm and Its Quantitative Analysis," *IEEE Tran. Pattern Analysis and Machine Intelligence,* vol. 20, no. 2, pp. 168-185, Feb. 1998.

[24] A. Sha'ashua and S. Ullman, "Structural Saliency: The Detection of Globally Salient Structures Using a Locally Connected Network," *Proc. Int'l Conf. Computer Vision,* pp. 321-327, 1988.

[25] J. Elder and S. Zucker, "Computing Contour Closure," *Proc. European Conf. Computer Vision,* pp. 399-412, 1996.

[26] L. Williams and A.R. Hanson, "Perceptual Completion of Occluded Surfaces," *Computer Vision and Image Understanding,* vol. 64, pp. 1-20, July 1996.

[27] S. Casadei and S.K. Mitter, "Hierarchical Image Segmentation-Detection of Regular Curves in a Vector Graph," *Int'l J. Computer Vision,* vol. 27, no. 3, 1998.

[28] S. Sarkar and K.L. Boyer, "Integration, Inference, and Management of Spatial Information Using Bayesian Networks: Perceptual Organization," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 15, no. 3, pp. 256-274, Mar. 1993.

[29] S. Sarkar and K.L. Boyer, "A Computational Structure for Preattentive Perceptual Organization: Graphical Enumeration and Voting Methods," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 24, pp. 246-267, Feb. 1994.

[30] L. Herault and R. Horaud, "Figure Ground Discrimination: A Combinatorial Optimization Method," *IEEE Tran. Pattern Analysis and Machine Intelligence,* vol. 15, no. 9, pp. 899-914, Sept. 1993.
[31] J.D. McCafferty, *Human and Machine Vision: Computing Perceptual Organization.* West Sussex, England: Ellis Horwood, 1990.
[32] S. Sarkar and K.L. Boyer, "Quantitative Measures of Change Based on Feature Organization: Eigenvalues and Eigenvectors," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* pp. 478-483, June 1996.
[33] S. Sarkar and K.L. Boyer, "Quantitative Measure of Change Based on Feature Organization: Eigenvalues and Eigenvectors," *Computer Vision and Image Understanding,* vol. 71, pp. 110-136, July 1998.
[34] L.S. Shapiro and J.M. Brady, "Feature-Based Correspondence: An Eigenvector Approach," *Image and Vision Computing,* vol. 10, pp. 283-288, 1992.
[35] K. Sengupta and K.L. Boyer, "Organizing Large Structural Modelbases," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 17, no. 4, pp. 321-332, Apr. 1995.
[36] E. Ihler, D. Wagner, and F. Wagner, "Modeling Hypergraphs by Graphs with the Same Mincut Properties," *Information Processing Letters,* vol. 45, pp. 171-175, Mar. 1993.
[37] K.L. Boyer, D.M. Wuescher, and S. Sarkar, "Dynamic Edge Warping: An Experimental System for Recovering Disparity Maps in Weakly Constrained Systems," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 21, pp. 143-158, Jan. 1991.
[38] J. Pearl, *Probabilistic Reasoning in Intelligent Systems.* San Mateo, Calif.: Morgan Kaufman, 1988.
[39] R.A. Horn and C.R. Johnson, *Matrix Analysis.* Cambridge Univ. Press, 1990.
[40] M. Fiedler, "Algebraic Connectivity of Graphs," *Czechoslovakian Math. J.,* vol. 23, pp. 298-305, 1973.
[41] A. Pothen, H. Simon, and K.P. Liou, "Patitioning of Sparse Matrices with Eigenvectors of Graphs," *SIAM J. Matrix Analysis Applications,* vol. 11, pp. 430-452, 1990.
[42] M. Fiedler, "A Property of Eigenvectors of Nonnegative Symmetric Matrices and Its Application to Graph Theory," *Czechoslovakian Math. J.* vol. 25, pp. 619-637, 1975.
[43] S.Z. Li, "Parameter Estimation for Optimal Object Recognition: Theory and Application," *Int'l J. Computer Vision,* vol. 21, pp. 207-222, Feb. 1997.
[44] M. Pelillo and M. Refice, "Learning Compatibility Coefficients for Relaxation Labelling Processes," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 16, no. 9, pp. 933-945, Sept. 1994.
[45] K.S. Narendra and M.L. Thathachar, *Learning Automata: An Introduction.* Prentice Hall, 1989.
[46] S. Borra and S. Sarkar, "A Framework for Performance Characterization of Intermediate-Level Grouping Modules," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 11, pp. 1,306-1,312, Nov. 1997.
[47] S. Sarkar, "Learning to Form Large Groups of Salient Image Features," *IEEE CS Conf. Computer Vision and Pattern Recognition,* pp. 780-786, June 1998.

Sudeep Sarkar received the BTech degree in electrical engineering from the Indian Institute of Technology, Kanpur, in 1988, where he was judged the best graduating electrical engineer. He received the MS and PhD degrees in electrical engineering, on a University Presidential Fellowship, from The Ohio State University, Columbus, in 1990 and 1993, respectively. Since 1993, he has been with the Computer Science and Engineering Department at the University of South Florida (USF), Tampa, where he is currently an associate professor. He is a recipient of the U.S. National Science Foundation CAREER Award in 1994, the USF Teaching Incentive Program Award for undergraduate teaching excellence in 1997, and the Outstanding Undergraduate Teaching Award in 1998. He is the coauthor of the book *Computing Perceptual Organization in Computer Vision* (World Scientific). He is also the guest coeditor of the *Computer Vision and Image Understanding Journal* (*CVIU*), special issue on perceptual organization in computer vision, October 1999. He is presently serving on the editorial boards for the *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *Journal of Pattern Recognition*, and *Pattern Analysis and Applications Journal*. His research interests include perceptual organization in single images and multiple image sequences, probabilistic reasoning, Bayesian Networks, low-level image segmentation, color-texture analysis of burn scars, nonrigid modeling of impact of burn scars, and performance evaluation of vision systems. His recent research projects are listed at http://marathon.csee.usf.edu/~sarkar/sarkar.html. Dr. Sarkar is a member of the IEEE.

Padmanabhan Soundararajan received the BE degree in electronics and communication from Mysore University, India, in 1995. From 1995 to 1998, he was a project assistant at the Indian Institute of Science, Bangalore, India. He is currently a PhD student in the Computer Science and Engineering Department at the University of South Florida, Tampa.