Visual Learning of Patterns and Objects

Walter F. Bischof and Terry Caelli

***Abstract*—We discuss automatic rule generation techniques for learning relational properties of two-dimensional (2-D) visual patterns and three-dimensional (3-D) objects from training samples where the observed feature values are continuous. In particular, we explore a conditional rule generation method that defines patterns (or objects) in terms of ordered lists of bounds on unary (pattern part) and binary (part relation) features. The technique, termed conditional rule generation (CRG), was developed to integrate relational structure representations of patterns and the generalization characteristics of evidenced-based systems (EBS). We show how this technique can be used for recognition of complex patterns and of objects in scenes. Further, we show the extent to which the learned rules can identify patterns and objects that have undergone nonrigid distortions.**

I. Introduction

ONE major problem in the development of systems for visual pattern and object recognition is the development of representations and search procedures that allow efficient instantiation of known models in new image data. Many three-dimensional (3-D) object recognition systems use database techniques that are designed to index efficiently model features in structures such as hash tables, trees, or constraint satisfaction networks (see [1] for a review of such systems). Although such approaches have proven somewhat successful in the recognition of models, patterns, or shapes, they lack the generalization capabilities that typically distinguish pattern recognition from database systems. In two-dimensional (2-D) pattern recognition, on the other hand, classical methods have been used to address the issue of generalization from samples. These include parametric and nonparametric classifiers [2] and, more recently, neural networks.

The current paper is concerned with the application of recent machine learning techniques to the solution of the generalization problem and related issues. More specifically, we study the application of machine learning techniques to the learning of relational structures as required for visual pattern and object recognition. These techniques can provide solutions to three problems: feature selection, generalization, and efficiency. The first problem, feature selection, refers to the problem of how to select and/or order pattern features in order to optimize the recognition process. The second problem, generalization, refers to the problem of generating "structural descriptions" that cover a set of training examples, as well as distorted and similar, unseen examples. The third problem, efficiency, refers to the problem of optimizing search and matching procedures for both, pattern learning and pattern recognition.

The type of pattern representation most frequently used in vision has been the *relational structure* (RS) [3]. In this approach, complex patterns are described as being composed of constituent parts. Pattern descriptions involve enumeration of (unary) features of pattern parts and (binary) features of relations between parts. In the case of an image, for example, the pattern parts might correspond to segmented image regions, the unary features might include area, average brightness or orientation of the regions, and the binary features might include distance, relative orientation or length of common boundaries between pairs of image regions. These part and part relation features can be linked together into a relational structure with parts corresponding to graph vertices and part relations corresponding to graph edges, both being described by a set of features [3], [4]. RS representations are limited in several respects. First, generalizations in terms of new views or nonrigid transformations of old views are difficult to represent. Second, pattern recognition involves typically graph matching with a computational complexity that is exponential in the number of parts [3], [5]. Prior knowledge can be used to prune the search space [6], [7] but the basic problem remains. For the same reason, RS representations are difficult to apply to the recognition of (possibly occluded) objects embedded in complex scenes. Finally, little attention has been paid to the design of optimal search procedures that use conjunctions of particular sets of feature values to define important characterizations of patterns.

Evidence-based systems (EBS's) have been introduced [8] to overcome some problems of the RS approach. Both approaches share common characteristics: they work within a supervised learning (learning by example) paradigm and they require subprocesses for encoding, segmentation and feature extraction. In EBS, patterns and objects are encoded by rules of the form

```
if ⟨conditions on feature values⟩
then ⟨evidence weights for each class⟩
else ⟨no inference at all⟩.
```

Rule conditions are usually defined in terms of bounds on feature values, and rules instantiated by data provide weighted evidence for different pattern classes. The main task of EBS has been to determine feature bounds and evidence weights. EBS typically involve partitioning the feature spaces into regions associated with different pattern classes, and the problem has been to find a partitioning that minimizes misclassifications while, at the same time, maximizing rule

Manuscript received February 26, 1995; revised December 22, 1995 and September 19, 1996. This project was funded by a grant from the Australian Research Committee. The first author was also supported by the Canadian Natural Sciences and Engineering Council under Grant OGP38251.

W. F. Bischof is with the Department of Psychology, University of Alberta, Edmonton, Alta., Canada T6G 2E9.

T. Caelli is with the Department of Computer Science, Curtin University of Technology, Perth WA 6001, Australia.

Publisher Item Identifier S 1083-4419(97)07130-6.

1083–4419/97$10.00 © 1997 IEEE

generalization. For a given feature (attribute) space partitioning, rule or pattern generality is defined in terms of the range of feature values associated with each partition, or, in other words, the volume of the feature space hyper-rectangle defined by the feature bounds of a rule. Consequently, pattern generality also determines the degree of feature variability or pattern distortion tolerated by each classification rule.

Regions in feature space are not necessarily class disjoint, and so evidence weights are usually used to index the degree to which samples within a region correspond to different classes. Evidence weights are typically derived from the relative frequencies of different classes per region [8] or, more recently, by minimum entropy and associative neural network techniques [9]. Some systems actually combine clustering (or feature space partitioning) with weight estimation into a single neural network architecture [10].

Although these types of evidence-based systems allow generalizations from samples, they only attain implicit learning of the relational structures. Pattern encoding is achieved using unary rules (rules related to part features) and binary rules (rules related to part relational features) that are both activated to evidence patterns or objects. However, structural pattern encoding is typically incomplete in EBS. This is so because EBS-generated rules are not necessarily label-compatible, i.e., the feature conditions in the EBS rules do not contain labels to index specific pattern parts and their relations in order to guarantee compatibility of unary and binary feature states. This leads to problems when patterns are defined uniquely by the enumeration of specific *labeled* unary and binary feature states of the form $U_i - B_{ij} - U_j$. This is illustrated in Fig. 1 where two patterns are shown that have isomorphic unary and binary feature states (i.e., color and distance values) but are not identical. This shows that the existence of such correspondences does not guarantee identity in structure unless the unary and binary feature labels are compatible. Rules satisfying this "label compatibility" property of rules must evidence objects or patterns uniquely, i.e., lists of unary and binary feature states must evidence *specific joint occurrences of parts and relations*. The problem then is how to generate rules having this property.

As already stated, a labeled and attributed graph is the simplest representation for visual patterns that takes into account the label-compatibility of unary and binary features. Graph matching techniques are used to solve the recognition problem where a sample pattern structure (for example, new data for classification) is matched to a model structure by searching for a label assignment that maximizes some objective similarity function [3]. Pattern classes are represented by sets of instances and classification is thus achieved by searching through all model graphs to determine the one producing the best match. This representation and graph matching approach, in the form of interpretation trees and feature indexing, has been the preferred architecture for object recognition [7], [1].

Different approaches to improving the efficiency of the matching processes have been proposed, such as constraint-based decision trees [6], "precompiled" tree generation [11], heuristic search techniques [12], dynamic programming [13], relaxation labeling [14], or hierarchical model fitting [15].

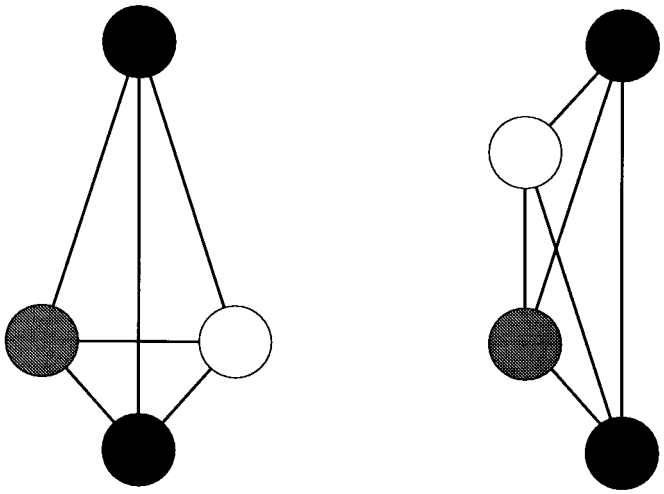

Fig. 1. Two patterns that have isomorphic unary (U = vertex color) and binary (B = distance) feature states but differ with respect to their label-compatibilities. That is, the sequences of $U_i - B_{ij} - U_j - \cdots$ differ between the two patterns (from [21]).

However, the problem of learning and constructing union and discrimination trees for structural descriptions has been addressed only sporadically in the literature, such as in [16] within the framework of inductive learning of symbolic structural descriptions or in [17] within the framework of probabilistic inductive prediction of sequential patterns.

In summary, graph matching methods solve the label-compatibility problem but do not address adequately the fundamental issue of generalization, i.e., the ability to recognize equivalences between patterns that are not identical. Also, they do not fully exploit learning to determine the optimal search path amongst unary and binary feature states to evaluate the existence of specific patterns. In 3-D object recognition, in particular, it is often necessary to classify objects as belonging to a specific object type even though individual samples of the class may be nonrigid transformations of other members of the same class. Evidence-based systems, on the other hand, provide a means for pattern generalization, but do not adequately address the label-compatibility problem.

In the following sections, we focus on the analysis of a technique for the learning of structural relations, *conditional rule generation* (CRG). The CRG method searches for the occurrence of unary and/or binary feature states between connected components of the training patterns and generates trees of hierarchically organized rules for classifying new patterns. It so enables induction (generalization) on labeled and attributed graphs as well as generating optimal decision trees for the identification of specific RS. The aim of this paper is to analyze how the CRG method can be applied to problems involving the recognition of 2-D patterns and 3-D objects in complex visual scenes.

II. Conditional Rule Generation

CRG is designed for the encoding and learning of complex patterns or objects that are assumed to consist of multiple parts. In object recognition, image regions are assumed to be segmented consistently into multiple regions or parts. The

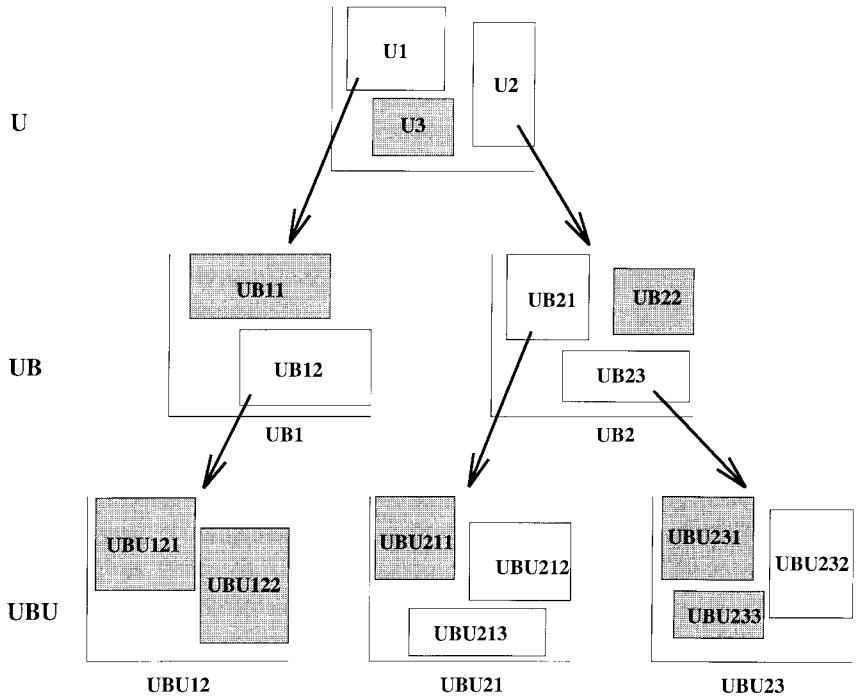

Fig. 2. Cluster tree generated by the conditional rule generation (CRG) procedure. Grey squares denote cluster that are unique with respect to class membership (i.e., clusters with $H_i = 0$); white squares denote nonunique clusters. The unresolved unary clusters (U_1 and U_2)—with element from more than one class—are expanded to the binary feature spaces UB_1 and UB_2. Expansion and clustering continues until either all clusters are unique with respect to class membership or the (*a priori* chosen) maximum tree depth is reached. In the latter case, the cluster tree is refined through reclustering or cluster splitting.

resulting parts are described by (unary) features such as area, average brightness, or eccentricity. Relations between parts are described by (binary) features, such as distance between centers, relative orientation, or length of common boundaries. As described in more detail below, rules in CRG are defined as clusters in conditional feature spaces which correspond to either unary or binary features of the training data. The clusters are generated to satisfy two conditions: they should maximize the covering of samples from one class, and they should minimize the inclusion of samples from other classes (see also [18]). In our approach, such rules are generated through decision tree expansion and cluster refinement as described below.

A. Cluster Tree Generation

In the following, we present the technique for generating cluster trees, first in an informal way and then more formally. Cluster tree generation begins by collecting the unary feature vectors of all parts, of all views, and of all objects into a unary feature space. This feature space is partitioned into a number of clusters. Some clusters may contain elements of a single pattern class and may thus provide classification rules for some pattern parts. The other clusters have to be analyzed further. For each part p of a nonunique cluster we collect all binary feature vectors of the relation between p and other pattern parts into a binary feature space. This feature space can be analyzed analogous to the unary feature space. The analysis continues by analyzing unary and binary features of longer and longer sequences (chains) of pattern parts until all pattern parts can be classified uniquely.

More formally, each training pattern is assumed to be composed of a number of parts (pattern components). Each part $p_r, r = 1, \cdots, N$ is described by a set of unary features $\vec{u}(p_r)$, and pairs of parts (p_r, p_s) are described by a set of binary features $\vec{b}(p_r, p_s)$. Below, $S(p_r)$ denotes the pattern to which a part p_r belongs, and H_i refers to the information or cluster entropy statistic:

$$H_i = -\sum_k q_{ik} \ln q_{ik} \tag{1}$$

where q_{ik} defines the probability that an element of cluster i belongs to class k.

First, the unary features of all parts of all patterns are collected into a unary feature space: $U = \{\vec{u}(p_r), r = 1, \cdots, N\}$. This feature space is partitioned into a number of clusters U_i. Some clusters (e.g., U_3 in Fig. 2) are unique with respect to class membership (with entropy $H_i = 0$) and provide a simple classification rule for some patterns: if a pattern contains a part p_r whose unary features satisfy the bounds of a unique cluster U_i then the pattern $S(p_r)$ can be assigned a unique classification. The nonunique clusters contain parts from multiple pattern classes and have to be analyzed further. For every part of a nonunique cluster U_i (e.g., U_2 in Fig. 2) we collect the binary features of this part with all other parts to form a (conditional) binary feature space: $UB_i = \{\vec{b}(p_r, p_s) \mid \vec{u}(p_r) \in U_i$ and $S(p_r) = S(p_s)\}$. This binary feature space is clustered into a number of clusters UB_{ij}. Again, some clusters may be unique (e.g., UB_{11} in Fig. 2) and provide classification rules for some patterns: if a pattern contains a part p_r whose unary features satisfy the bounds of cluster U_i, and there is

an other part p_s, such that the binary features of the pair $\langle p_r, p_s \rangle$ satisfy the bounds of a unique cluster UB_{ij} then the pattern $S(p_r)$ can be assigned a unique classification. For each nonunique cluster UB_{ij}, the unary features of the second part p_s are used to construct another unary feature space: $UBU_{ij} = \{\vec{u}(p_s) \mid \vec{b}(p_r, p_s) \in UB_{ij}\}$, which is clustered into clusters UBU_{ijk}. Again, unique clusters provide classification rules for some patterns (e.g., UBU_{121} in Fig. 2), the other clusters require further analysis, either by repeated conditional clustering involving additional parts at levels $UBUB, UBUBU$, etc. or through cluster refinement, as described below.

Every element of a cluster in the cluster tree corresponds to a sequence $U_i - B_{ij} - U_j - B_{jk} \cdots$ of unary and binary features associated with a noncyclic chain (path) of pattern parts. CRG thus produces classification rules for (small) pattern fragments and their associated unary and binary features whereas EBS and rulegraphs produce classification rules for sets of unary and binary features. In the current implementation, we analyze all chain permutations, i.e., all permutations of sequences $p_i - p_j - p_k, p_i - p_k - p_j, p_j - p_i - p_k, \cdots$. This is required in order to guarantee classification of arbitrary partial patterns.

Feature space clustering can be obtained using parametric or nonparametric clustering [2]. Alternatively, one can omit clustering altogether and rely completely on the cluster refinement methods described in the following section. In the current implementation of CRG, cluster trees are generated in a depth-first manner up to a (*a priori* chosen) maximum level of expansion (see Table I). Clusters that remain unresolved at that level are split in a way described in the following section.

TABLE I
CLUSTER TREE GENERATION

```
level(root) :=0
push(root,queue)
while (queue not empty)
    c := pop(queue)
    if level(c) = maxlevel then
        split(c)
    else
        f := ConditionalFeatureSpace(c)
        clist := cluster(f)
        foreach cluster c' ∈ clist
            if unresolved(c') then
                level(c') := level(c)+1
                push(c',queue)
            endif
        endforeach
    endif
endwhile


ConditionalFeatureSpace(c)
if c = root then
    type(f) := U
    elements(f) := {u(p_r), r = 1,..N}
else
    if type(c)=U then
        type(f) := B
        elements(f) := {b(p,q) | u(p) ∈ c and (p,q) in a chain}
    else
        type(f) := U
        elements(f) := {u(q) | b(p,q) ∈ c}
    endif
endif
return(f)
```

B. Cluster Refinement

All nonunique (unresolved) clusters at a given level of the cluster-tree (e.g., clusters UBU_{212}, UBU_{213}, and UBU_{232} in Fig. 2) have to be analyzed further to construct unique decision rules. One way of doing this is to simply expand the cluster tree, analyzing unary and binary attributes of additional parts to generate rules of the $\{UBUBUB\ldots\}$ form. However, if the features used are insufficient, it may be impossible to obtain completely "resolved" branches in the cluster tree. Alternatively, the derived clusters in the tree can be refined or broken into smaller clusters, using more discriminating feature bounds, as described below. Both approaches have their respective disadvantages. Cluster refinement leads to an increasingly complex feature-space partitioning and thus may reduce the generality of classification rules. Cluster-tree expansion, on the other hand, successively reduces the possibility of classifying pattern fragments, or, in the case of 3-D object recognition, classifying objects from partial views. In the end, a compromise has to be established between both approaches.

In cluster refinement, two issues must be addressed, the refinement method and the level at which cluster refinement should be performed. Consider the cluster tree shown in Fig. 2 with nonunique clusters UBU_{212}, UBU_{213}, and UBU_{232}. One way to refine clusters (for example, cluster UBU_{232}) is to recluster the associated feature space (UBU_{23}) into a larger number of clusters. However, classification rules associated with other clusters (UBU_{231} and UBU_{233}) are lost and have to be recomputed. Alternatively, given that each cluster is bounded by a hyper-rectangle in feature space, refinement of a cluster can be achieved by splitting this rectangle along some boundary. This ensures that other sibling clusters remain unaffected. With respect to the level at which cluster refinement is performed, instead of splitting an unresolved leaf cluster (UBU_{232}) one could split any cluster in the chain of parent clusters (UB_{23} or U_2).

Consider splitting the elements of an unresolved cluster C along a (unary or binary) feature dimension F. The elements of C are first sorted by their feature (attribute) value $f(c)$, and then all possible cut points T midway between successive feature values in the sorted sequence are evaluated. For each cut point T, the elements of C are partitioned into two sets, $P_1 = \{c | f(c) \leq T\}$ with n_1 elements and $P_2 = \{c | f(c) > T\}$ with n_2 elements. We define the partition entropy $H_P(T)$ as

$$H_P(T) = n_1 H(P_1) + n_2 H(P_2). \tag{2}$$

The cut point T_F that minimizes $H_P(T_F)$ is considered the best point for splitting cluster C along feature dimension F

(see also [19]). The best split of cluster C is considered the one along the feature dimension F that minimizes $H_P(T_F)$. As noted above, rather than splitting an unresolved leaf cluster C_L, one can split any cluster C_i in the parent chain of C_L. For each cluster C_i, the optimal split T_F is computed, and the cluster C_i that minimizes T_F is considered the optimal level for refining the cluster tree. Clusters above C_L may contain elements of classes other than those that are unresolved in C_L. Hence, in computing H_P for those clusters, we consider only elements of classes that are unresolved in C_L.

Two further properties of the splitting procedure are important, since they affect the type of rules generated by CRG. First, if a nonterminal cluster of the cluster tree is split, the feature spaces conditional upon that cluster are recomputed since the elements of the feature space have changed. Second, in the case of a tie, i.e., if two or more clusters have the same minimal partition entropy $H_P(T)$, the cluster higher in the cluster tree is split. Together, this leads to CRG having a clear preference for shallow cluster trees and for short rules, which, in turn, leads to efficient rule evaluation.

In the generation of a cluster tree, every new feature space (the initial unary feature space U, or any of the conditional feature spaces UB_i, UBU_{ij}, etc.) can be partitioned using standard cluster methods [9] to obtain an initial rule set. Alternatively, one can refrain from feature space clustering and rely completely on the splitting procedure introduced above. The latter approach was used in all applications reported below.

The rules generated by CRG are sufficient for classifying new pattern or pattern fragments, provided that they are sufficiently similar to patterns presented during training and provided that the patterns contain enough parts to instantiate rules. However, cluster trees and associated classification rules can also be used for partial rule instantiation. A rule of length m (for example, a $UBUBU$-rule) is said to be partially instantiated by any shorter $(l<m)$ sequence of unary and binary features (for example, a UBU-sequence). From the cluster tree shown in Fig. 2, it is clear that a partial instantiation of rules (for example, to the UB-level) can lead to unique classification of certain pattern fragments (for example, those matched by the U_3 or UB_{11} rules, but it may also *reduce* classification uncertainty associated with other nodes in the cluster tree (for example, UB_{23}). From the empirical class frequencies of all training patterns associated with a node of the cluster tree (for example, UB_{23}), one can derive an expected classification vector, or *evidence* vector. For example, if cluster UBU_233 (in Fig. 2) contains five elements of class 1, three elements of class 2, and no other elements, the associated evidence vector would be $E(UBU_233) = [0.625\ 0.375 \cdots]$. The evidence vector is used to predict the classification vector of any part, or sequence of parts, that instantiates the associated rule.

C. Evidence Combination

CRG generates rules for the classification of chains of pattern parts. In the application of these rules to a pattern, one obtains therefore multiple evidence vectors, typically one for each chain and each instantiation of a rule. These evidence vectors have to be *combined* into a single overall classification of the pattern. The problem of evidence combination is very closely related to the concept of "stacked generalization" [20] with some added difficulties that are discussed below. Dependent on a number of factors there are several possible approaches to evidence combination.

First, different rules can be devised dependent on whether information about training patterns is completely preserved or not. In the rulegraph approach [21], for example, information about training patterns is completely preserved and the classification of a sample pattern is based, in the end, on finding the best match of the sample graph to the stored training or model graphs. Here, evidence rules are simply used to prune the search tree. In EBS [22], on the other hand, pattern information that is not preserved in the classification rules is lost. Second, evidence combination schemes become typically more elaborate as one progresses from the recognition of single, complete patterns to single, incomplete (partially occluded) patterns to complex scenes containing multiple, incomplete patterns. Third, evidence combination rules can either be given *a priori* as in [20] or they can be learned as in [22]. In the present paper, we discuss the use of CRG for the case of complex scenes with multiple patterns, where information about training patterns is not completely preserved and where evidence combination rules are given *a priori*.

III. Detecting 2-D Patterns in Scenes

In this section, we illustrate learning of 2-D patterns using the CRG method and the recognition of these patterns embedded in more complex scenes using the generated rules. The first example, line triples, consists of four classes of patterns with four training examples each [see Fig. 3(a)]. Each pattern is described by the unary features "length" and "orientation," and the binary features "distance of line centers" and "intersection angle" between adjacent parts, i.e., part pairs whose center-to-center distances do not exceed a limit $(d_{\max})$. The line patterns are simplified versions of patterns found in geomagnetic data that are used to infer the presence of certain metals or minerals.

CRG was run with maximum rule length set to $maxlevel = 5$ (i.e., rules up to the form of $UBUBU$ are being generated), and it produced 35 rules, three U-rules, 18 UB-rules, two UBU-rules, and 12 $UBUB$-rules.

At recognition time, a montage of patterns was presented [see Fig. 3(b)], and the patterns were identified and classified as described below, producing the classification result shown in Fig. 3(d). Pattern identification and classification is achieved using the following steps.

1) Unary features are extracted for all scene parts (lines), and binary features are extracted for all adjacent scene parts, i.e., pairs whose center distance does not exceed the distance used in training $(d_{\max})$. The adjacency graph is shown in Fig. 3(c), where dots indicate the position of the line centers, and adjacent pattern parts (lines) with a center-center distance $d<d_{\max}$ are connected.
2) Given the adjacency graph, all noncyclic paths up to a certain length l are extracted, where $l \leq maxlevel$.

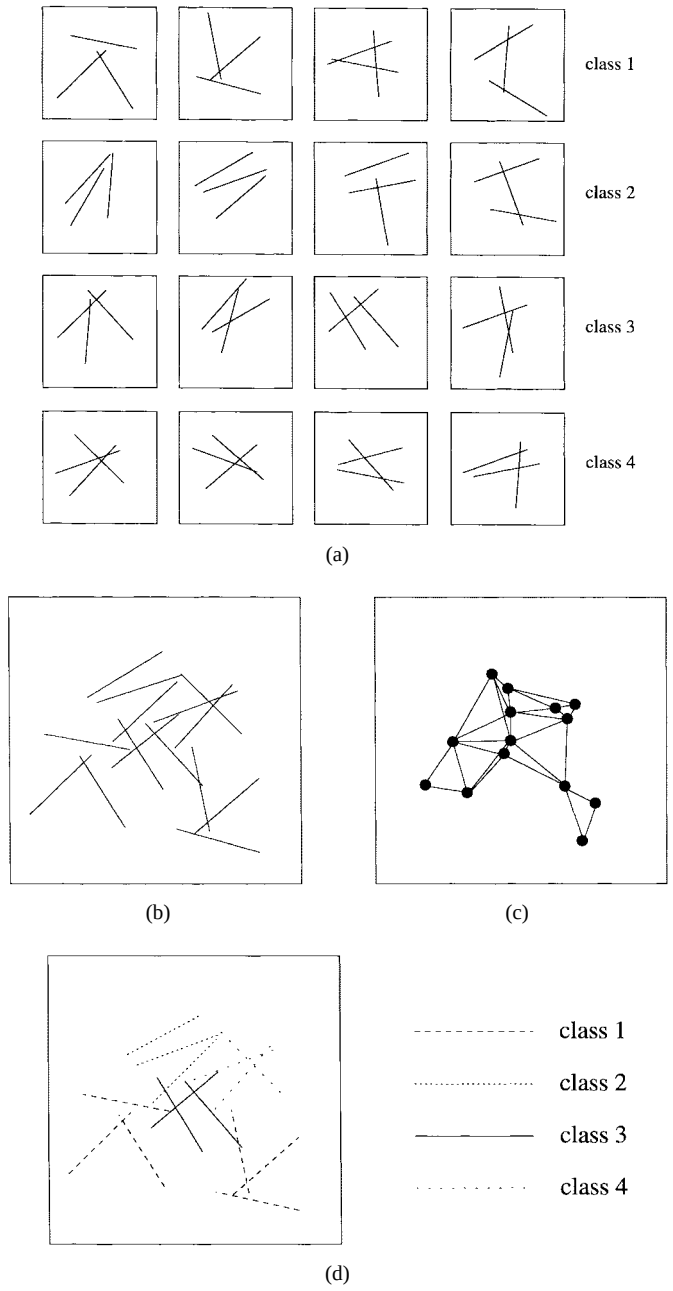

Fig. 3. (a) Four classes of patterns with four training patterns (views) each. Each pattern is composed of three lines. Lines are described by the unary features "line length" and "orientation," and pairs of lines are described by the binary features "distance of line centers" and "intersection angle." (b) Montage of (slightly distorted) line triples. (c) In the adjacency graph for the montage, dots indicate the position of the line center and adjacent lines (with a center distance below a given limit) are connected. (d) Result of the pattern classification using the rules generated by CRG. Class labels for each line are shown on the right.

These paths, termed *chains*, constitute the basic units for pattern classification. A chain is denoted by $S = \langle p_i, p_j, \cdots, p_n \rangle$ where each p_i denotes a pattern part. For some chains, all parts belong to a single learned pattern, but other chains are likely to cross the "boundary" between different patterns.

3) Each such chain $S = \langle p_i, p_j, \cdots, p_n \rangle$ is classified through parallel instantiation of the rules generated during training. Depending on the unary and binary feature states, a chain may or may not instantiate one (or more) classification rule. In the former case, rule instantiation may be partial (with a nonunique evidence vector $\vec{E}(S)$), or complete (with $H[\vec{E}(S)] = 0$). As discussed above, the evidence vector for each rule instantiation is derived from the empirical class frequencies of the training examples.
4) The evidence vectors of all chains $\langle p_i, p_{j_1}, \cdots, p_{n_1} \rangle, \langle p_i, p_{j_2}, \cdots, p_{n_2} \rangle, \cdots$ starting at a common part p_i must be combined to obtain a classification for part p_i.

We have studied two ways of combining the evidence vectors, a winner-take-all (WTA) solution and a relaxation labeling solution. Implementation of the WTA solution is straightforward. The evidence vectors of all chains starting at p_i are averaged to give $\vec{E}_{\text{av}}(p_i)$, and the most likely class label is used to classify part p_i.

The WTA solution does not take into account that, for a chain $S = \langle p_i, p_j, \cdots, p_n \rangle$, the average evidence vectors $\vec{E}_{\text{av}}(p_i), \vec{E}_{\text{av}}(p_j), \cdots, \vec{E}_{\text{av}}(p_n)$ may be very different and possibly incompatible. If they are very different, it is plausible to assume that the chain S is "crossing" boundaries between different patterns/objects. In this case, the chain and its evidence vectors should be disregarded for the identification and classification of scene parts. Accordingly, compatibilities between evidence vectors are taken into account using a relaxation labeling (RL) procedure. Here, the weight of an evidence vector $\vec{E}(p_i)$ of a part p_i depends on the similarity (compatibility) of $\vec{E}(p_i)$ to the evidence vectors of neighboring parts. Further, constraints on evidence vectors are propagated throughout the pattern using a standard relaxation labeling technique [23]. More precisely, the RL solution is given by

$$\vec{E}^{t+1}(p_i) = \Phi\left[\sum_{S=\langle p_i \cdots p_n \rangle} \vec{E}^t(p_i) C(p_i, p_n)\right] \tag{3}$$

where $\vec{E}^t(p_i)$ corresponds to the evidence vector of p_i at iteration t, with $\vec{E}^0(p_i) = \vec{E}_{\text{av}}(p_i)$. $C(p_i, p_n)$ corresponds to the compatibility between parts p_i and p_n, and Φ is the logistic function

$$\Phi(z) = (1 + \exp[-20(z - 0.5)])^{-1}. \tag{4}$$

The compatibility function is defined in terms of the scalar product between the evidence vectors of parts p_i and p_n,

$$z = C(p_i, p_n) = \vec{E}(p_i) \cdot \vec{E}(p_n). \tag{5}$$

For identical evidence vectors $\vec{E}(p_i)$ and $\vec{E}(p_n), C(p_i, p_n) = 1$, and for incompatible evidence vectors, for example $\vec{E}(p_i) = [1, 0, 0]$ and $\vec{E}(p_n) = [0, 1, 0], C(p_i, p_n) = 0$.

Compatibility of evidence vectors is a weak constraint for updating the evidence vectors of each part and it may even have an adverse effect if the adjacency graph is complete. This is due to the following. As mentioned before, the evidence vectors of all chains $\langle p_i, p_{j_1}, \cdots, p_{n_1} \rangle, \langle p_i, p_{j_2}, \cdots, p_{n_2} \rangle, \cdots$ starting at part p_i are combined to obtain a classification for part p_i. Some of these chains may involve completely "unrelated" parts, i.e., parts belonging to different patterns,

yet they may instantiate some classification rule. To the extent that this happens, classification of part p_i may be adversely affected and even be incorrect. For these reasons, good classification performance for complex pattern montages (as shown in Fig. 3) are not a trivial result. For the simple patterns shown in Fig. 3, and the moderate connectivity of the adjacency graphs of the montages, the relaxation method outlined here proved to be sufficient to obtain perfect part labeling. For the montage of undistorted, noise-free training patterns in Fig. 3(b), recognition performance was perfect. In order to test recognition performance for noisy patterns, we added Gaussian noise with a standard deviation of 1%, 2%, and 5% of the feature ranges to each feature. For these noise levels, performance dropped only moderately to an average of 97%, 89%, and 80%, respectively. When the patterns in Fig. 3(b) were separated from each other, thus eliminating the possibility that chains of "unrelated" parts were classified, pattern classification performance remained high at 99.3% correct for Gaussian noise with a standard deviation of 5% (of the feature ranges) and dropped to 87% correct for Gaussian noise with a standard deviation of 10%. These results clearly show that CRG is capable of finding adequate generalizations of the training patterns, and that it is capable of generating recognition rules that show a relatively high resilience to pattern distortions.

Much stronger constraints than compatibility of evidence vectors can be derived from more specific structural information of the training patterns, such as the label-compatibilities between pattern parts, or from pose information in the case of 3-D object recognition. The usefulness of such information is, however, pattern dependent and considered beyond the scope of the present paper.

IV. Object Recognition Using Intensity Data

The *blocks* example presented in this section consists of various configurations of colored blocks. The configurations are learned in isolation (see Fig. 4) and have to be identified in more complex arrangements (see Fig. 5). The training set consisted of five classes of block configurations, each with three training examples, and the test arrangements consisted of up to 20 blocks.

Images of the training and test scenes were captured with a color camera. Preprocessing was fairly simple, consisting of a *segmentation* stage and a *feature extraction* stage. Segmentation was achieved using a form of K-means clustering (minimizing within-cluster variance in feature space) on position (x, y) and color (r, g, b) attributes [24]. For the resulting clusters, small clusters were merged with larger neighbor clusters in order to eliminate spurious image regions. Given the rich image information, it is not surprising that the resulting image regions correspond fairly well to the individual block faces.

In the feature extraction stage, the following unary features were extracted for each image region: size (in pixels), compactness $(perimeter^2/area)$, and the normalized color signals $R/(R+G+B)$, $G/(R+G+B)$, and $B/(R+G+B)$. For pairs of image regions the following binary features were computed: absolute distance of region centers, minimum distance between the regions, distance of region centers normalized by the sum of the region areas, and length of shared boundaries normalized by total boundary length.

For the training data, CRG analyzed 276 different chains of pattern parts and produced 32 rules: nine U-rules, four UB-rules, 12 UBU-rules, three $UBUB$-rules, and four $UBUBU$-rules. From the distribution of rule types, it is evident that CRG used predominantly unary features for classification. Given the fact that CRG has a strong tendency to produce shallow cluster trees and short rules (see Section II-A), and given the fact that the unary features are quite diagnostic (see Fig. 4), this result is not surprising. However, each unary and binary feature was used in at least some of the classification rules.

Classification performance was tested with several complex configurations of block patterns, two of which are shown in Fig. 5, together with the classification results. Classification proceeded as described in Section III, using the chain analysis and relaxation labeling solution. For both scenes, all parts [11 in Fig. 5(a), 17 in Fig. 5(b)] were classified correctly with the exception of a single part from the class-4 configuration [see Fig. 5(c) and (d)]. The reasons for the occurrence of misclassifications were discussed in the previous section.

For comparison purposes, we have analyzed the block example using classical decision trees [25]. For decision trees, an implicit relational structure has to be imposed on the selection process. In the first analysis, a UBB-triple analysis, each image part P of the training and test images was described by 13 features. These features consisted of the five unary features of P (see above), the four binary features (see above) of the relation between P and its closest neighbor, and another four binary features of the relation between P and its second-closest neighbor. For the class 1 cases which consisted of two parts only, the feature values for the second binary relation were set to "unknown." A decision tree was generated using C4.5 with default parameters [25], and the resulting tree was used to classify all parts of the test scenes in Fig. 5. In each of the two scenes, three parts were misclassified. The good performance obtained with C4.5 is consistent with the observation that the use of higher-order relational information does not seem to be crucial for successful classification of this data set.

The first comparison using C4.5 employed features of all UBB-triples (unary features and binary features of relations with two other parts) for classification. A second analysis, using UBU-triples (with 14 features: the same five unary features of all pairs of parts, as well as the same four binary features of their relation) was performed, but the results are much worse. For the scene in Fig. 5(a), 33 out of 110 UBU-triples or 30% were misclassified, and for the scene in Fig. 5(b) 103 out of 272 UBU-triples or 37.8% were misclassified. Given these high error rates, it is not surprising that part classification for the scenes in Fig. 5 was rather poor. For the scene in Fig. 5(a), 11 out of 17 parts were classified correctly, and for the scene in Fig. 5(b), five out of 11 parts were classified correctly. This performance could not be improved using the relaxation scheme described in Section III.

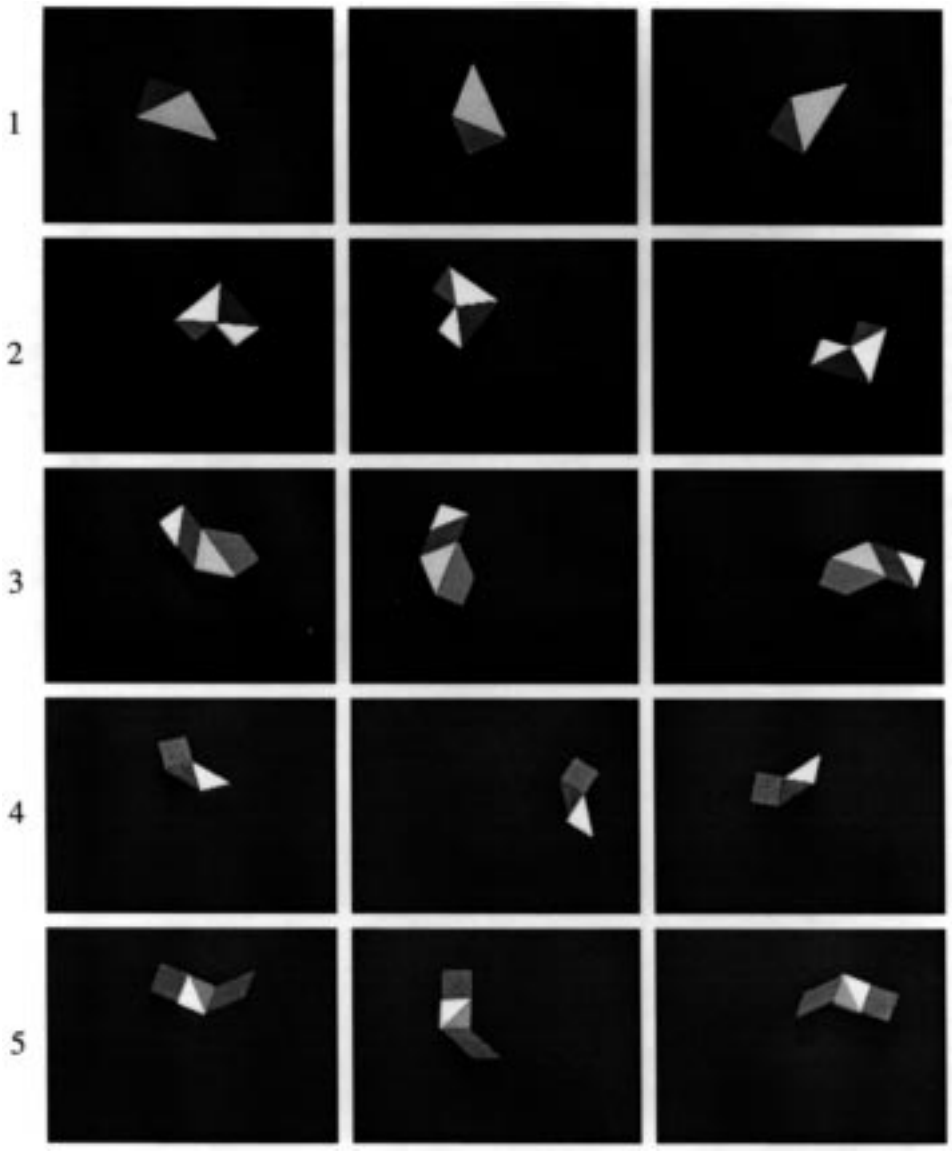

Fig. 4. Images of five classes of toy block configurations with three views each. The image parts are described by the unary features size, eccentricity and the three normalized color coordinates. Pairs of image parts are described by the binary features of midpoint distance, area-normalized midpoint distance, minimum distance, and normalized shared boundary length.

A general point is, however, more important. The CRG method generates rules of (minimal) variable length *optimized for a given training set*, whereas the decision tree (C4.5) *fixes* the dimensionality of the feature space and rule length. Indeed, C4.5 does not use part-indexing and so the relational structure has to be encoded implicitly in the attributes extracted from different parts and part relations. The choice of UBB-triples for the block example lead to a C4.5 performance that was essentially the same as that of CRG, but for the UBU-triples C4.5 performance was much worse. This choice has to be done *a priori* whereas it is adjusted dynamically in the CRG method. Further, CRG is designed to exploit structural information of patterns and dependencies between feature states, whereas C4.5 analyzes a fixed set of features that are assumed to be independent. In this sense, the application of C4.5 to the blocks data was somewhat misleading in the sense that the necessary and relevant structural information was generated manually. This example of forcing C4.5 to function with RS and complex scene data emphasizes the very need for systems like CRG, for relational learning algorithms in general.

Recently, Quinlan [27] and Muggleton and Buntine [28] have investigated general methods for learning *symbolic* relational structures in the form of Horn clauses in the following sense. In FOIL, [27] considers the problem of learning, from positive examples (closed world) or positive and negative examples, conjunctions of literals that satisfy

$$C \leftarrow L_1, \cdots, L_m$$

where C would correspond, in our case, to a class label. FOIL solves such problems by expanding the literals—adding predicates and their variables—to the right-hand-side to maximize the covering of positive instances and to minimize inclusion of negative ones. In this framework, then, CRG is also concerned

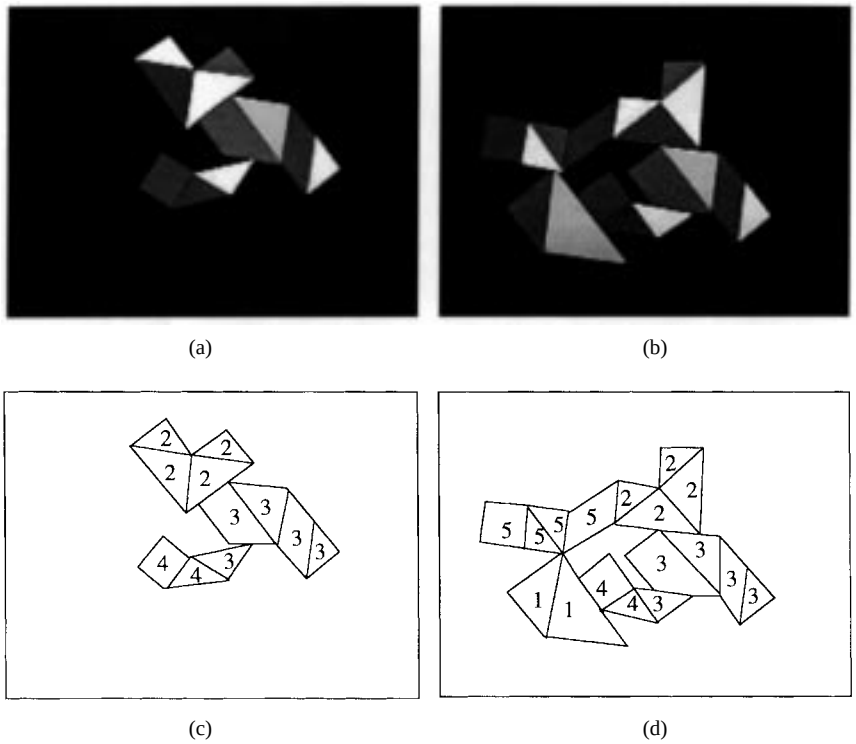

Fig. 5. Two block scenes and their classifications. (a) Block scene consisting of 11 blocks corresponding to examples of classes 2, 3, and 4. (b) Block scene consisting of 17 blocks corresponding to examples of all classes. (c) Classification result for block scene in (a) with region labels corresponding to classes. (d) Classification result for block scene in (b) with region labels corresponding to classes.

with generating similar class descriptions of the specific forms

$$C_1^1 \leftarrow U^1(X), B^1(X,Y), U^2(Y), B^2(Y,Z), U^3(Z), \cdots$$
$$\cdots$$
$$C_1^{m_1} \leftarrow U^1(X), B^1(X,Y), U^2(Y), B^2(Y,Z), U^3(Z), \cdots$$
$$\cdots$$
$$C_m^1 \leftarrow U^1(X), B^1(X,Y), U^2(Y), B^2(Y,Z), U^3(Z), \cdots$$
$$\cdots$$
$$C_m^{m_m} \leftarrow U^1(X), B^1(X,Y), U^2(Y), B^2(Y,Z), U^3(Z), \cdots.$$

However, CRG differs significantly from FOIL in the following ways.

1) Choice of unary U-rules and binary B-rules as bounded attribute (feature) states, is determined within continuous unary and binary feature spaces.
2) Ordering of literals must be *satisfied* in the rule generation.
3) Search technique uses backtracking and recursive splitting.
4) Resultant rules are not only Horn clauses but each literal *indexes* bounded regions in the associated feature space (as shown in Fig. 2).

V. Discussion

CRG develops structural descriptions of patterns in the form of decision trees on attribute bounds of ordered predicates with labeled parts and part relations (see Fig. 2). It is thus useful to compare it with other techniques from machine learning which attain similar ends symbolically.

CRG shares with ID3/C4.5 [25], [26] and related techniques, similar methods for the search and expansion of decision trees. However, these latter techniques were not designed to generate rules satisfying label compatibility between unary and binary predicates. CRG, on the other hand, is explicitly designed to develop rules for unique identification of classes with respect to their "structural" (i.e., linked unary and binary feature) representation.

In decision trees, features or attributes are analyzed within a single feature space, independent of their relationships or arities, and no preferential order is imposed on the features. In contrast, the CRG method generates conditional feature spaces and defines a preferential ordering on attributes in the sense that, for example, a split of a U-feature is preferred over a split of UBU-features. This preferential order leads to the generation of shallow cluster trees and short rules, as discussed in the previous sections.

Decision trees operate on a fixed chain length (for example, the UBB- or UBU-triples in the block example) and thus *force*, *a priori*, the choice of implicit relational structures to be analyzed. CRG, on the other hand, has variable length chain expansion determined by the number of parts and their relations that are required to uniquely define patterns. Consequently, CRG is superior to classic decision trees when classification relies on relational information and does so to different degrees for different patterns or classes. Under these circumstances one would be forced to use high-dimensional features spaces with classical decision trees, whereas CRG would generate minimal depth trees. Generating minimum depth trees is, however, of crucial importance given

that the number of chains grows exponentially with chain length.

In summary, one can say the classical decision trees are *attribute-indexed* in the sense that various levels in the tree define different attributes and the nodes define different attribute states. CRG adds another structural layer to the decision tree structure, a *part-indexed* tree of features spaces, each with its own attribute-indexed decision tree. With this tree of decision trees, CRG imposes both a limit on the number of attributes that are being considered, and an ordering on the evaluation of attributes.

In the CRG method, pattern recognition is achieved by combining evidence about the classification of small pattern fragments (chains). This approach allows the classification rules to be applied efficiently to arbitrary partial patterns. On the other hand, given that the number of pattern chains grows exponentially with chain length, it is not feasible to apply CRG in situations where pattern recognition has to rely on very long chains of pattern parts. At the level of attribute testing, the complexity of CRG is identical to classical decision trees. However, the unique *relational aspects* of CRG may result in more efficient learning, depending on the type of learning context.

It should be noted that pattern recognition using CRG does not require perfect identification or classification of all component parts. As illustrated earlier, the CRG method identifies "focal" features which best discriminate between classes and, as such, provide critical and sufficient signatures for recognition. However, one difference with respect to the issue of performance evaluation lies in whether CRG is treated as an automated relational hashing procedure or a technique for pattern definition which discards the training data. If the latter is true then CRG cannot directly recover pose or exact model projections onto data. This, in some instances, may not be required. However, for a more exact performance of the procedure, including training data within the data structures allows us to illustrate the strength of recognition and model projection.

The CRG method is an example of the general solution to complex pattern recognition problems involving the generation of rules which are linked together in ways that determine "structure" uniquely enough to identify classes but enable generalization to tolerate distortions. Both aims, uniqueness and generalization, are not explicitly guaranteed in other methods, such as neural networks or decision trees. Further, uniqueness and generalization constitute the equivalent of a "cost" function in CRG, and the search technique has been developed to satisfy these constraints.

Finally, CRG raises the question as to what really is a "structural description" of a pattern. CRG simply generates conditional rules that combine an attempt to generalize the pattern definitions in terms of feature bounds and to restrict the description lengths as much as possible. For complex and highly variable training patterns, CRG can generate a large number of rules which can be thought of as a set of *equivalent descriptions* of the pattern structure. It is possible to determine the more frequently occurring chains and associated feature bounds from the cluster tree, if the notion of "commonness" is deemed necessary for a structural description. However, this may not really be a meaningful definition of structure. Rather than producing a singular rule structure, a "structural description" is defined by a *set of rules* that CRG generates from a set of training patterns.

REFERENCES

[1] P. Flynn and A. K. Jain, "Three-dimensional object recognition," in *Handbook of Pattern Recognition and Image Processing, Vol. 2: Computer Vision*, T. Y. Young, Ed. New York: Academic, 1993.

[2] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[3] D. Ballard and C. Brown, *Computer Vision*. Englewood Cliffs, NJ: Prentice-Hall, 1982.

[4] L. Shapiro and R. Haralick, "Structural descriptions and inexact matching," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-3, pp. 504–519, 1981.

[5] R. E. Tarjan and A. E. Trojanowski, "Finding a maximum independent set," *SIAM J. Comput.*, vol. 6, pp. 537–546, 1977.

[6] W. E. L. Grimson, *Object Recognition by Computer*. Cambridge, MA: MIT Press, 1990.

[7] P. Flynn and A. K. Jain, "3-D object recognition using invariant feature indexing of interpretation tables," *Comput. Vision, Graph., Image Processing*, vol. 55, pp. 119–129, 1992.

[8] A. K. Jain and D. Hoffman, "Evidence-based recognition of objects," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, pp. 783–802, 1988.

[9] T. Caelli and A. Pennington, "An improved rule generation method for evidence-based classification systems," *Pattern Recognit.*, vol. 26, pp. 733–740, 1993.

[10] T. Poggio, "Networks for approximation and learning," *Proc. IEEE*, vol. 78, pp. 1481–1497, 1990.

[11] K. Ikeuchi and T. Kanade, "Automatic generation of object recognition programs," *Proc. IEEE*, vol. 76, pp. 1016–1035, 1988.

[12] R. C. Bolles and P. Horaud, "A three-dimensional part orientation system," *Int. J. Robot. Res.*, vol. 5, pp. 3–26, 1986.

[13] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *IEEE Trans. Comput.*, vol. C-22, pp. 67–92, 1973.

[14] R. Mohan and R. Nevatia, "Using perceptual organization to extract 3-D structures," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, pp. 1121–1139, 1989.

[15] D. G. Lowe, "Three-dimensional object recognition from single two-dimensional images," *Artif. Intell.*, vol. 31, pp. 355–395, 1987.

[16] R. Michalski and R. E. Stepp, "Automated construction of classifications: Conceptual clustering versus numerical taxonomy," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-5, pp. 396–409, 1983.

[17] K. C. C. Chan, A. K. Wong, and D. K. Y. Chiu, "Learning sequential patterns for probabilistic inductive prediction," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, pp. 1532–1547, 1994.

[18] R. S. Michalski, "A theory and methodology of inductive learning," in *Readings in Machine Learning*, J. W. Shavlik and T. G. Dietterich, Eds. San Mateo, CA: Morgan Kaufmann, 1990.

[19] U. Fayyad and K. Irani, "On the handling of continuous-valued attributes in decision tree generation," *Mach. Learn.*, vol. 8, pp. 87–102, 1992.

[20] D. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, pp. 241–259, 1992.

[21] A. Pearce, T. Caelli, and W. F. Bischof, "Rulegraphs for graph matching in pattern recognition," *Pattern Recognit.*, vol. 27, pp. 1231–1248, 1994.

[22] T. Caelli and A. Dreier, "Variations on the evidenced-based object recognition theme," *Pattern Recognit.*, vol. 27, pp. 185–204, 1994.

[23] A. Rosenfeld and A. Kak, *Digital Picture Processing*. New York: Academic, 1982.

[24] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[25] J. R. Quinlan, *C4.5 Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.

[26] ———, "Induction of decision trees," *Mach. Learn.*, vol. 1, pp. 81–106, 1986.

[27] ———, "Learning logical definitions from relations," *Mach. Learn.*, vol. 5, pp. 239–266, 1990.

[28] S. Muggleton and W. Buntine, "Machine invention of first-order predicates by inverting resolution," in *Proc. 5th Int. Conf. Machine Learning*, 1988, pp. 339–352.

Walter F. Bischof received the Ph.D. degree in psychology from the University of Bern, Bern, Switzerland, in 1982.

He is currently Associate Professor of Psychology and Adjunct Professor of Computer Science at the University of Alberta, Edmonton, Alta., Canada, and Adjunct Associate Professor of Computer Science at Curtin University of Technology, Perth, Australia. His research interests are in the areas of human and machine vision. In human vision, he is interested in spatio-temporal characteristics of early visual analysis and motion perception. In machine vision, he is interested primarily in pattern and object recognition, with a particular emphasis on learning of recognition methods.

Terry Caelli received the Ph.D. degree in visual pattern recognition from the University of Newcastle, U.K., in 1975.

He is Professor of Computer Science and Head of the Department at Curtin University of Technology, Perth, Australia. His interests are in human and machine vision, pattern recognition, image interpretation, machine learning, and applying these technologies to industrial, agricultural, mining, and environmental problems. He has published widely in vision and pattern recognition and serves on the editorial board of five international journals in the area.

Dr. Caelli has won a number of awards including outstanding paper awards, distinguished visiting DFG Professor (Munich, 1986) and the Convocation Medal for Research from the University of Newcastle, 1993.