Pergamon

Pattern Recognition, Vol. 27, No. 9, pp. 1231–1247, 1994
Elsevier Science Ltd

0031-3203/94 $7.00 + .00

0031-3203(94)E0038-M

RULEGRAPHS FOR GRAPH MATCHING IN PATTERN RECOGNITION

ADRIAN PEARCE,† TERRY CAELLI‡ and WALTER F. BISCHOF‡§

† Department of Computer Science, The University of Melbourne, Parkville, Victoria 3052, Australia
‡ Department of Computer Science, Curtin University of Technology, Perth, Western Australia 6001, Australia
§ Department of Psychology BSP-577, The University of Alberta, Edmonton, Alberta T6G 2E9, Canada

(*Received* 22 *April* 1993; *in revised form* 24 *March* 1994; *received for publication* 12 *April* 1994)

Abstract—In Pattern Recognition, the Graph Matching problem involves the matching of a sample data graph with the subgraph of a larger model graph where vertices and edges correspond to pattern parts and their relations. In this paper, we present *Rulegraphs*, a new method that combines the Graph Matching approach with Rule-based approaches from Machine Learning. This new method reduces the cardinality of the (NP-Complete) Graph Matching problem by replacing model part, and their relational, attribute states by rules which depict *attribute bounds* and evidence for different classes. We show how rulegraphs, when combined with techniques for checking feature label-compatibilities, not only reduce the search space but also improve the uniqueness of the matching process.

A* search Classification Evidence-Based Systems Feature indexing Graph matching Machine learning Pattern recognition Relational structures Structural descriptions Subgraph isomorphism

1. INTRODUCTION

This paper is concerned with integrating two different approaches to Pattern Recognition, Graph Matching and Evidenced-Based Systems, into a new method, Rulegraph Matching. Graph Matching refers to the classical approach which views Pattern Recognition as a problem of matching data samples to models, where both are represented as graphs of pattern parts (vertices) and relationships between parts (edges). These kinds of graphs are also known as Structural Descriptions[1] or Relational Structures. For n vertices, the worst-case computational complexity of the matching process is known to be of the order $O(2^{n/3})$ and cannot be improved even when valences (connectivities) of each vertex are taken into account.[2,3]

Evidence-Based Systems (EBSs) are typically used in Expert Systems.[4,5] They represent models, or class data, via the enumeration of attribute (feature) states which *evidence* pattern classes and assign evidence *weights* according to the occurrence of feature states in observed data. Rather than use a graph representation, EBSs typically use *rules* of the if-then-else form and the evidence is accumulated over the weights for rules which have been activated. Both unary (vertices in the graph representation) and binary (edge) features are typically included in the feature lists.[6,7]

EBSs do not typically consider the *label-compatibilities* between model feature states in so far as evidence rules are activated simply when feature states are present—whether they are unary or binary. For example, the existence of two parts of given sizes, and an observed distance between two parts may trigger three evidence rules. However, EBSs do not usually consider the relationships between the *specific parts* which trigger the unary rules and the pairs of parts which trigger the binary rules. This does not imply that EBSs cannot have local compatibility checks in their rule generation. For example, it is possible to generate rules in joint unary (U) and binary (B) feature space. But the dimensionality of such a space must correspond to a Cartesian product space of U × U × B in order to guarantee that each sample point in the space has a label consistent with unary and binary features. Since EBSs typically avoid such high-dimensional spaces, uniqueness and such conjoint rules are usually sacrificed for rules based on simple unary or binary feature states. Furthermore, local compatibility checks of this type are often insufficient to differentiate between samples.

What EBSs loose in uniqueness, they gain in speed. In essence, the EBS is a technique for learning those feature states which capture important properties of patterns and it produces *rules* which can optimally discriminate between classes without considering the specific joint occurrences of parts and their relations in the data set.

On the other hand, Graph Matching, by definition, takes into account parts and their relations in the matching process. However, issues of generalization from model data and optimal matching strategies are typically not developed in graph matching processes. In the present paper, we develop a new pattern matching technique which exploits the benefits of both methods by combining them in a single framework.

2. GRAPH MATCHING

The Graph Matching formulation for Pattern Recognition reduces to essentially that of finding the *best* match between representative model (class) graphs and given data graphs. All vertices and edges of the graphs are labeled using unique identifiers and vertices or edges may share common feature states, as shown in Fig. 1(a). In this example, the unary feature attribute is colour and the binary features are distance and angle between parts.

In general, graphs may be matched by comparing vertices and edges according to their contribution to a relational distance metric.[1] There are three main approaches to the graph matching problem: Indexed Search, Constraint Propagation and Relaxation Labeling.

Indexed Search

In Indexed Search, parts and their relationships are ordered in terms of their ability to identify and discriminate pattern classes on feature states using Feature Indexing.[8] Perhaps the clearest example of this approach is the, now classical, Decision Tree[9] where models or patterns are identified from data by descending a decision tree and each level defines different sets of attribute (feature) bounds for different classes. Such trees have been used in Pattern Recognition[10] over the past two decades. Indexed Search, however, does not usually consider the *conjoint occurrence of unary and binary features* and consequently does not differentiate between graphs with different adjacencies (connectivities). This situation gets worse as the valency of the graphs is reduced, i.e. when there are fewer binary features.[11] More recently, Interpretation Tables[12] have used features of higher arity, triples, and a matching technique closely related to geometric hashing. Classification performance of Indexed Search can be adversely effected when parts of the input data are either missing or distorted—for example, due to occlusion and sampling errors—leading to the absence of feature states necessary for reliable recognition.

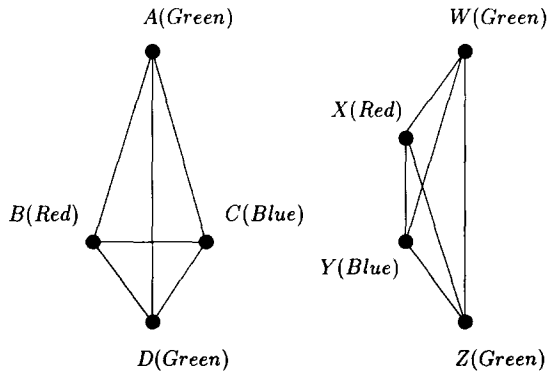


Fig. 1. (a) A labeled and attributed graph. Global label-compatibility checks (Subgraph Isomorphism) are required to differentiate between this graph and the graph in (b) as their unary and binary feature states, in isolation, are identical (same set of colours—unary, same set of distances and angles—binary).

Constraint Propagation

Constraint Propagation is similar to Indexed Search in so far as feature states constrain the search of the database but, in addition, it takes into account the joint occurrence of unary and binary features. In face recognition, for example, it is not only the existence of a nose, mouth and a given distance between two unspecified parts which is important, but also the joint occurrence of the nose and mouth in a specific relationship (part i–relation ij–part j: U_i–B_{ij}–U_j). In Interpretation Trees[13] specific relationships are checked by allowing unary and binary constraints to propagate through an associated search tree in accordance with the *in-place* occurrences of individual parts and their relationships. *Interpretations* of the sample in terms of model graphs are generated at each terminal node of the search tree. It should be noted that this type of *local* unary–binary–unary (U_i–B_{ij}–U_j) label-compatibility does not guarantee uniqueness of structural match, particularly if the unary and binary features are limited or if higher-order constraints are necessary to define patterns.[1] In Fig. 1 a configuration of parts is shown in (a) which is isomorphic to (b) using local compatibility only. However, the configurations are different due to a simple reallocation of unary features.

The problem of representing higher-order adjacency can be addressed using local binary constraints together with global labeling and Subgraph Isomorphism. For example, subgraph matching systems based on Subgraph Isomorphism have been used in Computer Vision.[14] In this scheme *global* label-compatibility checks are made with *previously* matched parts based on a set of label mapping states. When a *new* part is matched, the new label mapping must be consistent with previous mappings states for this label and this enables differentiation between the two graphs shown in Fig. 1. The complexity of Constraint Propagation is higher than that of Traditional Bipartite Matching due to the backtracking required. Even though such problems can be solved by branch and bound search,[15,16] it is not necessarily efficient since a depth-first backtracking scheme can still be expensive.

Relaxation Labeling

Relaxation Labeling solutions to the Graph Matching problem employ a parallel iterative scheme which updates the mapping between model and data parts as a function of the compatibilities between the part relations in each structure. Since the connection of a node with its neighboring nodes is fundamental to a graph, relaxation-based processing extends naturally to computations over graphs using labels as constraints.[17] Kitchen and Rosenfeld[18] describe a Discrete Relaxation Labeling scheme for matching relational structures

and Kim and Kak[19] have proposed a scheme using bipartite matching embedded in discrete relaxation for matching relational structures for 3D object recognition. In all cases, the solution found is not guaranteed to be optimal, and Simulated Annealing[20] may be used to increase the likelihood of a globally optimal match.

Again, in this paper we connect the generalization capabilities (rule bound selection) of Indexed Search methods like Decision Trees with the label-compatibility checking process implicit in the latter two methods, all within the context of EBSs.

3. EVIDENCE-BASED SYSTEMS

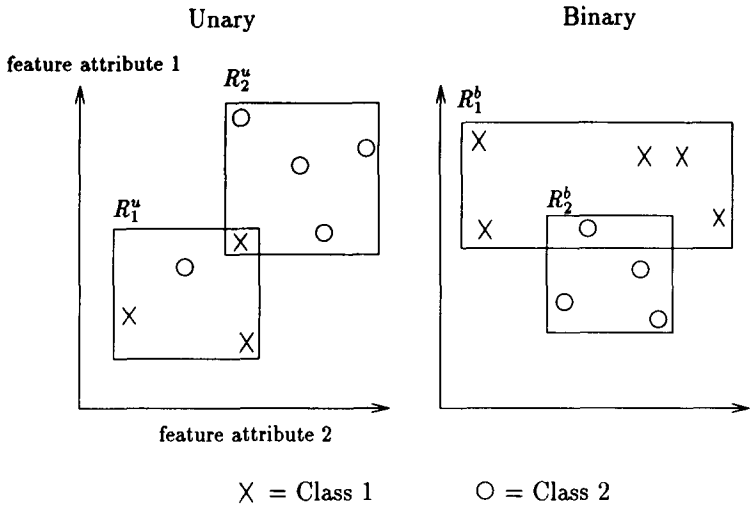
Evidence Based Systems (EBSs)[21] summarize known class or model data by sets of rules each of which defines feature (attribute) bounds which, in various degrees (weights), "evidence" different classes or models. The learning component of EBSs is that of determining the bounds and weights (generalization) from the training data. Such automatic rule generation procedures use well known clustering techniques (for example, Leader and K-means methods, see[22,23]), the ID3 System[9,24] and various probability paradigms including Bayesian and Dempster-Shafer Theory.[25] Caelli and Dreier[6,7] describe such a scheme for classifying data based on unary and binary features in a 3D Object Recognition System (ORS) and we will examine the automatic rule generation procedure of that system.

Feature values of parts and relationships are mapped to points in unary and binary *feature spaces* where each axis corresponds to different attributes from the feature sets. Points in feature space from a union of all training patterns are first clustered using the Leader clustering algorithm in which points are grouped according to a neighborhood distance threshold. The heads of such rules are then formed by fitting a bounding hyper-rectangle to each cluster which minimize a class-based entropy function. The rules are of the form:

IF $\text{Bounds}_{lower,upper}(\text{feature}_1, \ldots, \text{feature}_n)$
THEN Evidence Weights $(w_1, \ldots, w_m)$
ELSE no evidence.

Figure 2(a) shows a situation where points are not segregated in the feature space and the corresponding

(a)



(b)

Rules	Class Weights	
	C_1	C_2
R_1^u	0.75	0.25
R_2^u	0.20	0.80
R_1^b	0.83	0.17
R_2^b	0.00	1.00

Fig. 2. An Evidenced-Based System (EBS) rulebase: (a) EBS rules (R) are shown as rectangles for unary (u) and binary (b) feature spaces of two attribute dimensions; and (b) evidence weights for each class and each rule are derived from the class relative frequencies.

rules are overlapping, a situation which does not occur in Traditional Feature Indexing. Indeed, the clustering scheme is essentially a non-parametric method of deriving a probability distribution for all parts in the training data.

For each of these rules, evidence weights may be derived using the training patterns from which they were generated. In the simplest case, these weights may be the frequencies of points for each class (see Fig. 2(b)). At run time, rules are activated by the collective unary and binary features present in a sample pattern and a total class weight is returned to allow identification of the class membership for the sample. The rule activation is parallel and so evidence is evaluated in linear time with respect to the number of features.

As stated above, Caelli and Dreier[6,7] use a minimum entropy clustering procedure for generating rules which optimize classification when class samples are not contiguous or segregated in feature space. The method combines clustering with simulated annealing[26] to minimize a combined cluster entropy function, and rule boundaries are shifted to optimize the class membership of each rule. The evidence weights are then derived by training a Neural Network[27] thus allowing a nonlinear combination of evidence weights for different patterns of rule activations (see Fig. 3).

Evidence-Based Systems (EBSs) do not explicitly label data (assign labels to each part and relation) in the definition of models or patterns. Rather, the *existence* of a model is simply *evidenced* by the activation of sets of unary and binary rules.

Perhaps the main limitation of the EBS-Neural Network approach is that the representation, from an analytic viewpoint, is not unique in so far as rules are generated without explicit consideration of the relationships between specific unary and binary feature states that define specific objects. This is attained *implicitly* via the hidden units in the Neural Network (Fig. 3) where unary and binary feature states occurring in the same view would simultaneously activate one or more hidden units. This process does not *guarantee* a unique representation of structural relations in the data.

However, EBSs rule generation schemes can be used to reduce the search space of models and parts. In the reduced search space, traditional subgraph matching can be used to check predicate correspondence and to guarantee an optimal match between data and model.

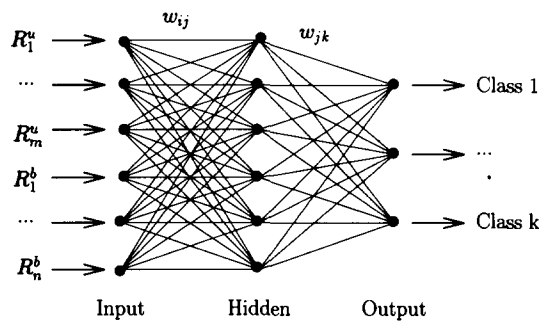


Fig. 3. A single hidden layer Neural Network is shown for use in matching the Sample Pattern shown in Fig. 4(c) with the Class 1 Training Pattern. All unary and binary rules are used for input and the output corresponds to the class label. A non-linear activation function is used to calculate the state of each node depending on the state of each connecting node and the weights w_{ij}.

4. RULEGRAPHS

In this section, we introduce the notion of rulegraphs. They combine three essential components:

- A set of rules produced by an EBS.
- A method for maintaining label information.
- An Bayesian framework for relational evidence.

The labeling of rules is demonstrated by the simple example in Fig. 4(a). Here, two classes of polyhedra are shown which are described by the unary features "perimeter" and "colour" and the binary features "distance between centers" and "sum of corner distances". Two unary rules and two binary rules represent the different feature states in each feature space. For the two classes, feature states are shared by the same set of rules and, as a result, the EBSs representation, alone, is unable to differentiate between these two classes.

If unique label identifiers are assigned to each of the parts from the training patterns then the compatibility between these EBS-rules can be maintained. For example, the binary feature states of relation AC for class 1 and EF for class 2 are both represented by R_1^b. However, the parts giving rise to these relations are represented by different rules—R_1^u for A and R_2^u for C as opposed to R_2^u for E and F. *Labeling* of these rules is used to represent the compatibility between them. Unary rules are labeled with single label identifiers, and binary rules are labeled with pairs of label identifiers, for example, $R_1^u(A, D)$ and $R_1^b(AC, EF)$ (see Fig. 4(a)).

The *compatibilities between rules* can be represented by considering the unary rules as vertices and the binary rules as edges in a graph using the labels to determine the connections. A *rulegraph is a graph of rules* in which vertices correspond to unary rules and edges correspond to binary rules according to the following connection criterion:

- Two unary rules R_i^u and R_j^u are connected by a binary rule R_k^b if there exists labels X, Y such that $X \in R_i^u$ and $Y \in R_j^u$ and $XY \in R_k^b$.

A *rulegraph model* for a training pattern corresponds to a graph where unary and binary rules replace model parts and their relationships. In Fig. 4(b) two different rulegraph models are shown which represent the training patterns for class 1 and class 2. Rulegraphs, therefore, explicitly represent the rules produced by EBSs and their interrelations via shared label instances and they capture compatibility information with respect to the structural aspects of the pattern description.

In Fig. 5(b) a rulegraph model is shown for the

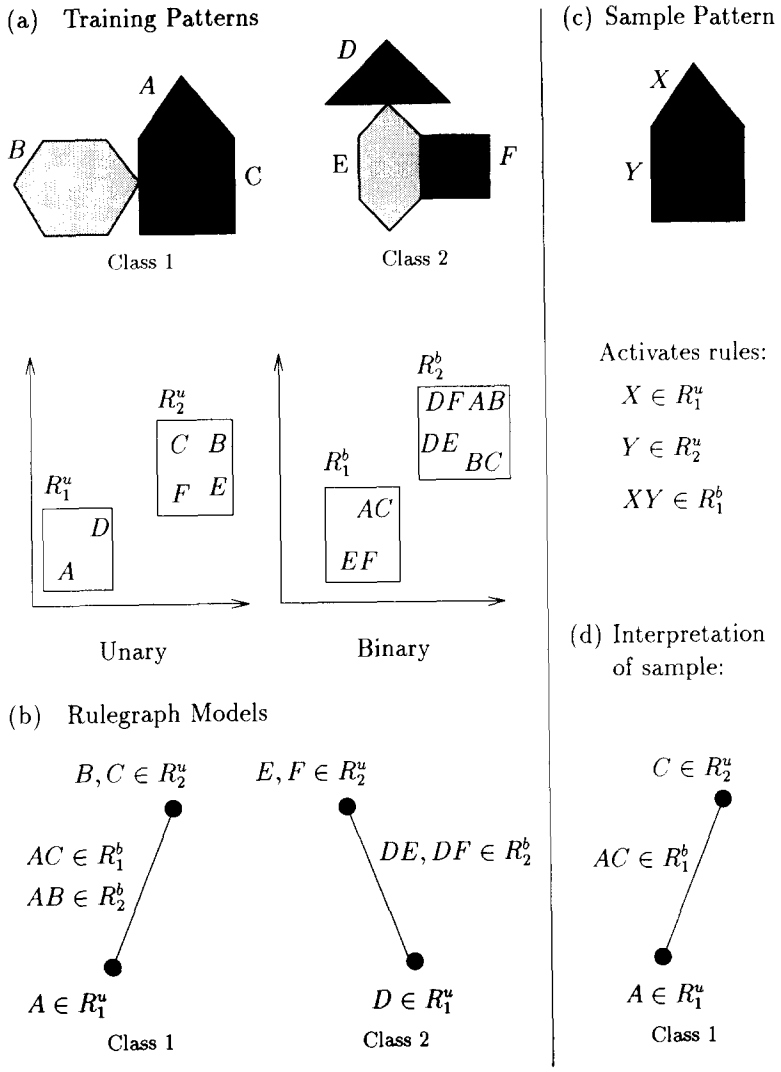


Fig. 4. Training patterns are used in (a) to label the unary and binary rules according to the mapping of the parts and their relationships into each feature spaces. Unary rules are labeled with single labels and binary rules are labeled with label pairs. Rulegraph models may then be formed, according to the connection criterion, and these are shown in (b). At run time, parts in the sample pattern activate unary and binary rules based on their feature states as shown in (c). The search for label-compatible rules between the sample and the model results in a rulegraph interpretation (best match) as is seen in (d).

simple graph using colours and distance. The rulegraph has three unary rules—R^u_{red}, R^u_{green} and R^u_{blue}—and two binary rules corresponding to two ranges of distance—R^b_{short} and R^b_{long}. The rulegraph is a convenient representation since it has lower cardinality than the original graph as each unary rule may contain multiple labels. Further, multiple binary rules may connect two unary rules consistent with this connection criterion. The adjacency information is reflected by the connectivity in the rulegraph according to the labeling of each rule vertex and edge.

The *likelihood* of a rulegraph corresponding to each class in the training set may be determined by the evidence weights for each rule vertex and edge. For each unary and binary EBSs-rule we can determine probabilities from the class frequencies within a given rule's bounds in feature space. For example, for R^u_1 from Fig. 2(b), $p(\text{class}_1|R^u_1) = 0.75$ and $p(\text{class}_2|R^u_1) = 0.25$.

The relational property of rulegraphs is to be contrasted with normal rule-based systems which assume—justified or not—independence of evidence.[28,29] This assumption leads to problems when evidence weights are dependent on the adjacency of parts as well as the parts themselves (as shown in Figs 1 and 4(a)). In this case *labeling* is the key to effective *combination of evidence* as it defines just what the dependencies between unary and binary feature states should be to instantiate a given model or pattern class. Indeed, both Compatibility Relations and Inference Networks have previously been used in the Expert Systems domain.[28] Rulegraphs carry these techniques over into Visual Pattern Recognition.

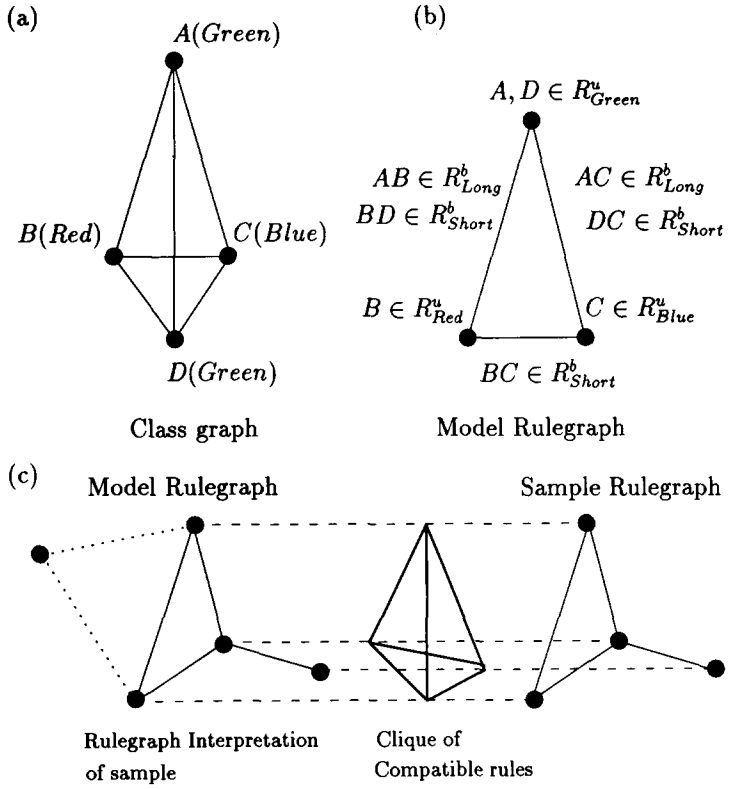


Fig. 5. The cardinality of the rulegraph model shown in (b) is reduced with respect to the class graph it represents (a) as parts have been replaced by rules. Multiple binary rules may connect two unary rules consistent with the connection criterion. A clique of compatible rules corresponds to a set of label-compatible rules between the model and the sample shown in (c). A Rulegraph Interpretation of the sample in terms of the model corresponds to rules from this clique.

5. RULEGRAPH MATCHING

In this section we show how rulegraph models may be used to classify labeled sample data by checking the compatibility between rules. A modified version of Subgraph Isomorphism is used to check label compatibility between *rules* rather than *parts*. This reduces the cardinality of the search space normally associated with Traditional Subgraph Isomorphism.

A Bayesian probability framework is used to determine evidence weights for rules, and matching is carried out using a simple metric for relational structures. The Relational Evidence Metric is presented in detail in Appendix B. The evidence weights are then used to probabilistically reduce the search space further, using A* search. As far as the authors know, this has not been used before for Subgraph Isomorphism.

Initially, parts in the sample pattern activate unary and binary rules based on the feature states of the parts (see Fig. 4(c)). The task is now to determine rules which are label-compatible between model and sample, by pairing vertices and edges in much the same way as in traditional Graph Matching.

5.1. *Label compatibility checking method*

The aim of the label compatibility checking method is to check the compatibility between the labels in the model rules and the labels in the sample data. In rulegraphs, multiple labels may be present in the vertices and edges, and this necessitates a new method for checking label-compatibility.

Among the methods for checking compatibility between *individual parts*, Subgraph Isomorphism is the most effective in differentiating between samples. In **Traditional Subgraph Isomorphism**, labels in the sample are *mapped* to labels in the model, provided that their mapping states correspond, and this gives rise to sets of *mapping states*. Compatibility between the sample and model relies on a consistency of the mapping states and can be checked using Constraint Propagation Methods. For example, the steps used to determine compatibility between the two labeled sample parts X, Y in Fig. 4(c) and A, C from class 1 in Fig. 4(a) are as follows.

Traditional subgraph isomorphism for ordinary graphs.

1. From primitive parts X, Y in the sample and A, C in the model, test the possible Mapping States: $A \rightarrow X$ and $C \rightarrow Y$.
2. Existence Check: given Mapping States $A \rightarrow X$ and $C \rightarrow Y$, if edge AC exists in the model graph then edge XY must exist in the sample graph.
3. For successful Existence Checks, update the list of acceptable Mapping States with the Mapping States $A \rightarrow X$ and $C \rightarrow Y$.

In step 1, possible mapping states are created for the labels between the sample and model. Each label in the sample must be exactly mapped to only one label in the model. In step 2, the existence criterion for edges of the sample graphs is checked. In step 3, the mapping states for each label are updated with the new states.

In rulegraphs several labels may exist in each rule vertex and this gives rise to *multiple* mapping states involving the same labels. A new technique which determines label compatibility between *rules* instead of *parts*—**the Label Compatibility Checking Method**—is now outlined for rulegraphs. This method uses a modified existence criterion capable of handling multiple labels and the binary evidence weights are used for updating the Mapping States. The steps used to determine compatibility between the two active rules R_1^u and R_2^u for the sample in Fig. 4(c) and the model rulegraph for class 1 in Fig. 4(b) as follows.

Label compatibility checking method for rulegraphs.

1. From unary rules R_1^u and R_2^u we have the possible Mapping States:
$[R_1^u: A \rightarrow X$ and $R_2^u: B \rightarrow Y$ and $R_2^u: C \rightarrow Y]$.
2. Existence Check: given Mapping States $R_1^u: A \rightarrow X$ and $R_2^u: C \rightarrow Y$, if $AC \in R_j^b$ exists in the model rulegraph then $XY \in R_j^b$ must exist in the sample data.
3. For valid Existence Checks, update the Mapping States with $A \rightarrow X$ and $C \rightarrow Y$ by instantiation (if the label is *not* yet mapped) or elimination (if the label *is* mapped). The new Mapping States are:
$[R_1^u: A \rightarrow X$ and $R_2^u: C \rightarrow Y]$.
4. Several R_j^b can exist between R_1^u and R_2^u. In this case $R_1^b - R_n^b$ are updated in order of decreasing evidence weights.
5. There must exist at least one $R_j^b \in R_n^1 \ldots R_n^b$ for which $AC \in R_j^b XY \in R_j^b$.

In step 1, possible mapping states are created for *all* the labels in the two rules. In this scheme single labels may be represented in several mapping states. In step 2, a modified existence check is carried out for each pair of mapping states for R_1^u and R_2^u. In step 3, the (multiple) mapping states are updated by instantiation (if the label is *not* yet mapped) or elimination (if the label *is* mapped) using the mappings generated from the new existence checks and the old mapping states. In step 4, the mapping states are updated in order of decreasing evidence weights of rules into which they map (since several binary rules can exist between two unary rules). This ensures that labels which have strongest evidence for a particular class will be mapped first. Finally, in step 5, we check that at least one binary rule is satisfied. A description of the complete algorithm is given in Appendix A.

Steps 1–5 of the Label Compatibility Checking Method describe a technique for checking compatibility between two rules. The matching process may now proceed by finding sets of rules which are all pairwise compatible—*cliques*[30] (see Fig. 5(c)):

- A *clique* of compatible rules corresponds to a set of rules where each rule is label-compatible with every other rule.

The problem of finding the *best* match now reduces to that of finding the clique which has the largest total evidence weight. The process of compatibility testing updates that label mapping states and—as a result of backtracking—search is required in order to find the best match (as in Constraint Propagation, see Grimson[13]). Terminal nodes of the resulting search tree correspond to *interpretations* of the sample with respect to model rulegraphs (as shown in Fig. 4(d)).

5.2. *A* search for best match*

The cardinality of the search problem (excluding label-compatibility checks) has already been reduced to the number of unary rules instead of the number of primitive parts. The evidence weights can be used in rulegraphs to direct the search toward rules and models for which strong evidence for isomorphism exists using dynamic programming principles. To achieve this we use A* search combined with the Relational Evidence Metric described in Appendix B to allow probabilistic pruning of the search tree.

The aim of the A* search is to find the best interpretation of the sample rulegraph in terms of the model rules, i.e. the clique of compatible rules with the largest evidence weight. A* is essentially a branch and bound search utilizing current cost with a heuristic estimate of the remaining cost.[31] The heuristic is based on the evidence of the current match plus an upper bound estimate of the potential match remaining and this results in optimal pruning of the search. The current match is obtained by evaluating the evidence weight for the rules in the current clique of compatible rules and the upper bound of potential match possible can be calculated based on the (optimistic) assumption that all presently compatible rules turn out to be compatible with one another. For example, the sample in Fig. 4(c) activates rules R_1^u, R_2^u and R_1^b. Initially, evidence for the potential match for each class is calculated simply using the active rules—since the cliques are all empty. For class 1 $R_1^u(A)$, $R_2^u(B, C)$ and $R_1^b(AC)$ and for class 2 $R_1^u(D)$ and $R_2^u(E, F)$. In addition, duplicate search states are removed, further pruning the search space—as follows:

A search for best match using rulegraphs.*

1. Form a queue of all cliques of compatible (initially empty) rules for all model classes.
2. Maintain queue in decreasing sorted order of: sum of clique evidence weights plus an upper bound estimate of the complete sum of evidence weights.
3. Initially, all cliques are empty. A clique may be extended with active rules that are compatible with every rule in the clique.
4. Repeat until best match is found:

(a) Extend the first clique on the queue by one rule.

(b) For each extension, update the label mapping states and re-insert into queue while removing duplicates.

(c) A best match has been found when clique can no longer be extended.

An initial priority queue of active rules from all classes is constructed based on estimates of the potential match weight for each rulegraph model, and the queue is sorted in decreasing order of evidence potential. The queue contains rulegraph interpretations from *all* classes of data and is searched simultaneously. This maximizes the pruning effect of A* search by only extending those cliques which have the highest potential for being the best match. This results in alternate classes being examined during the course of the search. A clique is extended with rules in decreasing order of their evidence weights, thus ensuring that the sample parts are first assigned to the model parts to which they most likely correspond.

The best match has been found when the clique at the head of the queue cannot be further extended. The queue order guarantees that extensions of cliques further down the queue cannot possibly yield a better match. The result of such a search for the sample in Fig. 4(c) is shown in Fig. 4(d). Here the best *rulegraph interpretation* for the sample is shown in terms of the rulegraph model for class 1. The system produces an overall evidence weight for the interpretation corresponding to the likelihood of the sample coming from the class.

6. CLASSIFICATION PERFORMANCE AND COMPLEXITY: 2D COMPLEX PATTERN RECOGNITION

The Rulegraph Matching technique is designed to improve on the uniqueness in pattern classification

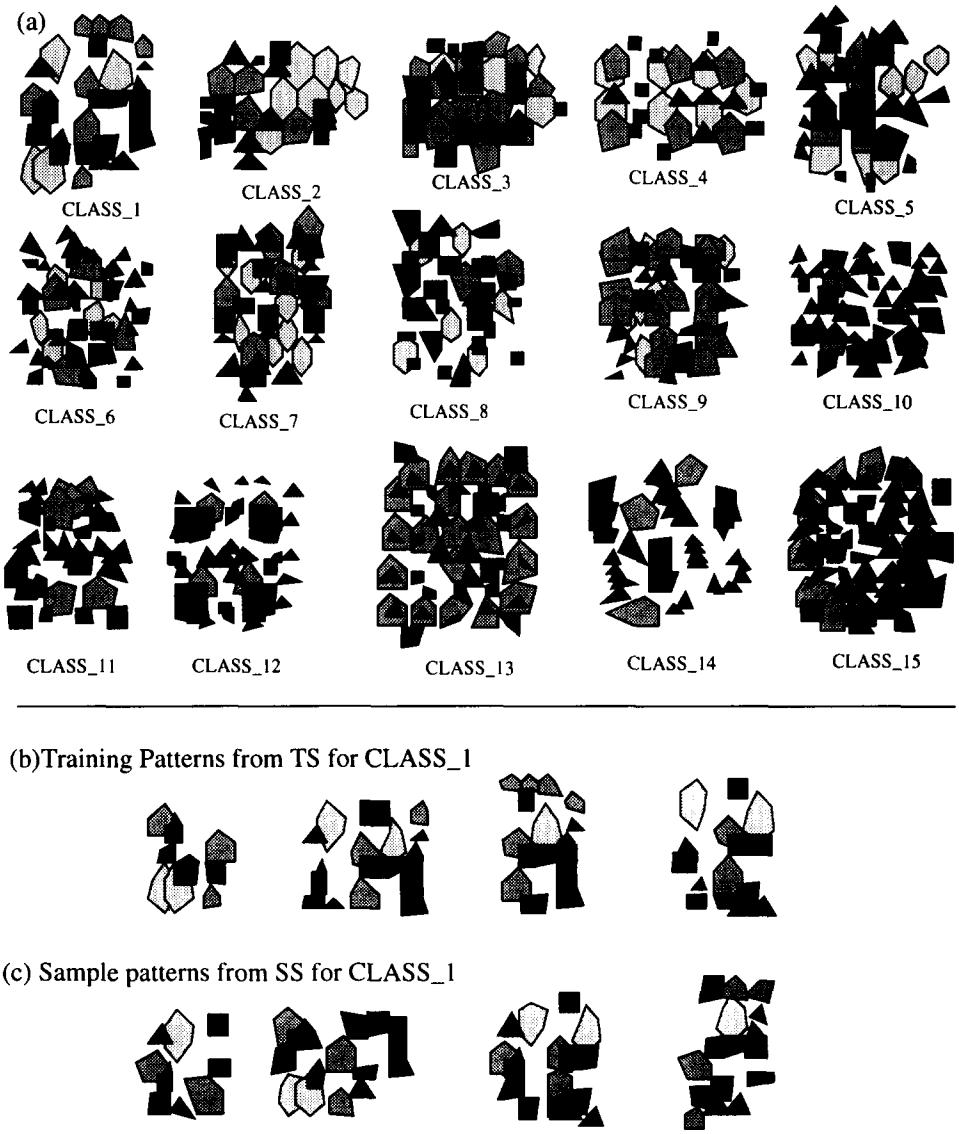


Fig. 6. In (a) all 15 classes for the Blocks Data are shown. Four training patterns (or views) for class 1 are shown in (b). (c) Four different test patterns with distorted features and missing (occluded) parts.

while, at the same time, reducing the computational complexity of the graph matching stage. Rulegraph Matching has to be evaluated both with respect to classification performance and with respect to complexity of matching. First, classification performance of Rulegraph Matching is compared to that of EBS using a Neural Network and that of Traditional Subgraph Isomorphism. Second, a comparison of the complexity of these different approaches is made and in order to do so systematically, synthetic data were used.

A complete set of patterns is shown in Fig. 6(a). For the Training Set (TS), four fragments were extracted from each of the 15 patterns (see Fig. 6(b)). Similarly, four different (not necessarily part disjoint) fragments were extracted from each of the 15 patterns for the test Sample Set (SS). This scheme of pattern sampling simulates occlusion and data loss and is consistent with sampling regimes found in 3D Object Recognition and other complex Pattern Recognition problems. In addition, both unary and binary feature attributes were distorted using additive Gaussian noise with a variance corresponding to five percent of the original feature variance. This moved the corners, colour and spatial positioning of the polyhedra relative to the class from which they were sampled (see Fig. 6(c)).

The extraction of relations between all part pairs creates significant redundancies and adds a squared factor to the storage requirement. As a result, only adjacent and non-overlapping binary features were used. Data graphs with different numbers of vertices were generated by varying the numbers of parts, but in all cases the SS consisted of 60 different samples over which performance was averaged. The data is not guaranteed to be perfectly classifiable and exhibits many characteristics fundamental to problems encountered in Pattern Recognition.

The rule generation scheme used Leader clustering (in feature spaces) based on the nearest neighbor method and required only a single parameter, a distance threshold. Smaller thresholds generate more specific—and more numerous—rules with lower class entropy values with respect to the TS and higher thresholds generate more general—less numerous and possibly overlapping—rules that are resilient to variation and distortion of the data. As a result there is an optimum number of rules associated with any particular Pattern Recognition problem though, in this example, we have run tests with different numbers of rules.

For comparison purposes, the EBS (see Caelli and Dreier[6,7]) was trained using a Neural Network (EBS-NNet) with one hidden layer in which the number of nodes was equal to the maximum of the input (number of unary and binary rules) or output (number of classes)—whichever was larger. Backpropagation was used to minimize the error in the network and 1000 sets of training epochs were used over several different learning rates and the best performing trained network were used for classification.

Tests were also performed by matching using Traditional Subgraph Isomorphism matching and the same relational evidence metric that was used in Rulegraph Matching. This was done in order to find the best possible classification for each data set. For Subgraph Isomorphism, a depth-first search strategy was used utilizing Branch-and-Bound (SI-BB) which constrains the search when it is not possible to reach a better match result via extension of the current interpretation. Depth-first search of this type is typically preferred for problems of high cardinality[16] since breadth-first search can lead to exponential space requirements. Indeed, without the use of a depth-first strategy we would not have been able to obtain exhaustive search results—the Branch and Bound scheme used is as follows:

Branch and bound depth first search for best match using subgraph isomorphism.

1. Form a queue of all cliques of compatible (initially empty) parts for all model classes.
2. Initially, all cliques are empty. A clique may be extended with active parts that are compatible with every part in the clique.
3. Set the best match (so far) to the first clique.
4. Repeat until best match is found:

(a) Extend the first clique on the queue by one part.

(b) For each extended clique:

(i) Update the label mapping states.

(ii) If the Sum of clique evidence weights plus an upper bound estimate of the complete sum of evidence for the clique is greater than for the best match then re-insert into the *front* of the queue else discard.

(iii) If the clique has a higher sum of clique evidence weights than the best match set the best match to the clique.

(c) A best match has been found when clique can no longer be extended.

Classification performance for matching the Training Set (TS) to itself is shown in Fig. 7(a). It can be seen that both EBS-NNet and Rulegraph Matching achieve perfect classification when the number of rules is sufficiently large to enable each rule to differentiate between patterns. Since there is no occlusion or distortion this is similar to a simple string matching procedure, and Rulegraph Matching achieved perfect classification with very small sets of rules. Using the occluded and distorted SS best classification performance was achieved with between 5 and 15 rules (see Fig. 7(b)). It can be seen, here, that the best classification performance for the Rulegraph Matching (88%) is considerably better than for the EBS-NNet (55%) and it is almost as high as is possible using Traditional Subgraph Isomorphism (SI-BB) (90%).

The high classification performance of Rulegraph Matching can be attributed to its ability to encode more class information through the use of labels, while, at the same time, allowing for rules which are general enough to allow for variation and distortion of data.

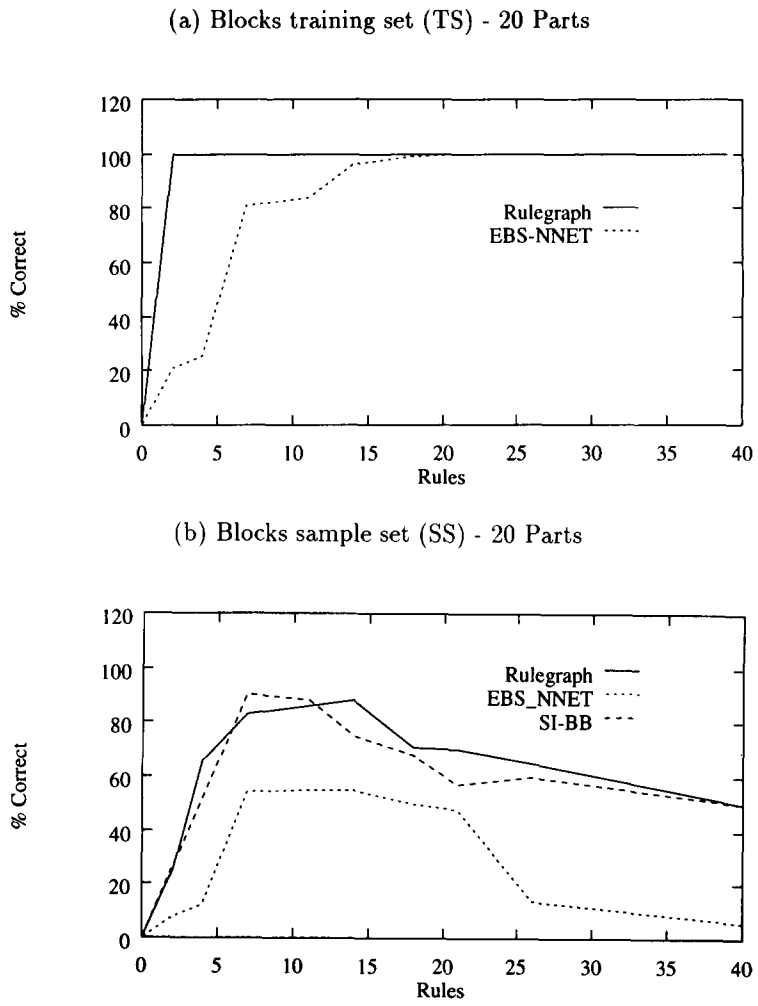


Fig. 7. (a) Classification performance for different numbers of rules for the Training Set (TS). (b) Classification performance is shown for the distorted and occluded Sample Set (SS). Here Rulegraph, EBS-NNet and SI-BB correspond to the Rulegraph Matching system, the Evidence Based-Neural Network system and Traditional Subgraph Isomorphism with Branch-and-Bound, respectively.

In this implementation, as the number of rules increase there is less generalization.

Using the same data sets, we can also compare the computational complexity of the different methods. For the case of the EBS-NNet the complexity is determined by the number of weight additions at each node of the network during the feedforward operation. For a fully connected network the complexity is $O(r^2)$ with r being the number of nodes in the hidden layer, provided, of course, that this number is larger than the sum of unary and binary features activating the rules. For both SI-BB and Rulegraph Matching, the complexity is determined by the total number of operations which compare a single edge in the model with respect to the existence of a single label-compatible edge in the sample. As a result, we have expressed the computational cost of SI-BB and Rulegraphs in terms of existence checks. For comparison purposes, we equate an existence check of Rulegraph Matching and SI-BB with one weight addition in EBS-NNet.

Results for the average computational cost for the same Blocks Sample Set (SS) are shown in Fig. 8(a). It is apparent that Label Compatibility method used by the Rulegraph Matching system requires only a fraction of the operations required by Traditional Subgraph Isomorphism (SI-BB). In fact, it is close to the number of operations required by the EBS-NNet. The numbers of existence checks was consistent with the observed run times. EBS-NNet and Rulegraph Matching system matched nearly instantaneously while SI-BB consumed large amounts of computation time. Further, it should be noted that rulegraphs are superior to Neural Nets at *learning time*: frequencies and labels of training data are merely recorded, while Neural Nets require substantial training time for Backpropagation.

In summary, the results indicate that the rulegraphs offer a classification performance close to the obtainable optimum and a significant improvement over EBSs; in particular for occluded and distorted data. The computational complexity of the rulegraph method

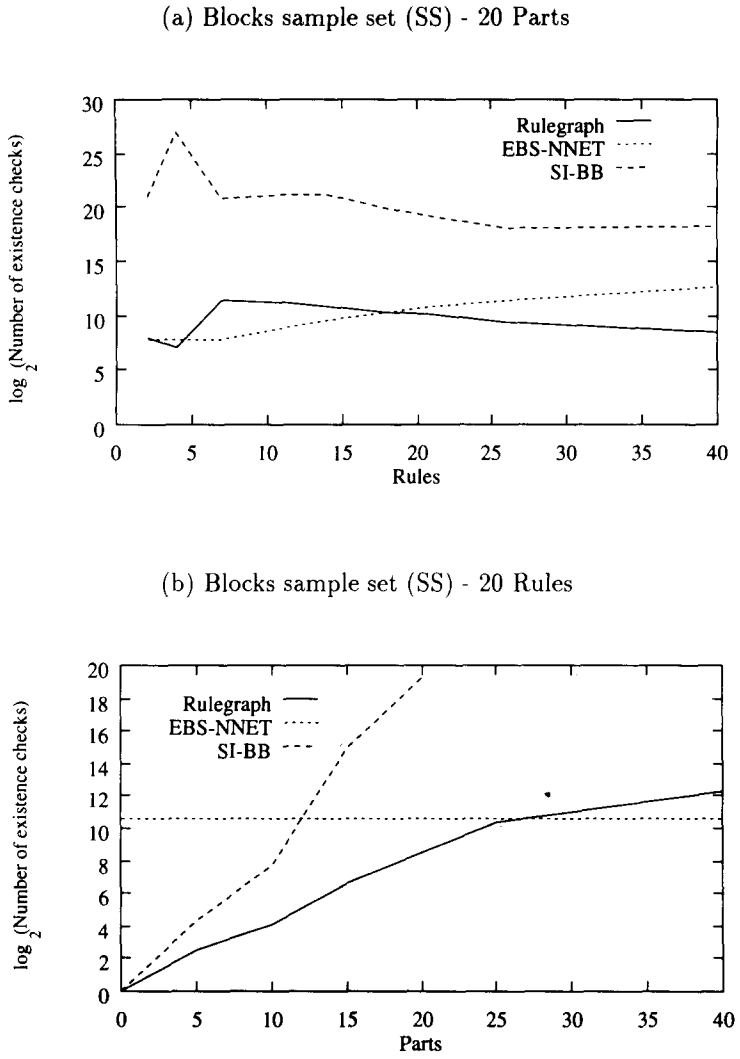


Fig. 8. A comparison of the average case computational complexity expressed in $\log_2$ of the total number of existence checks required to find best match, for the different systems. (a) The results for the Sample Set (SS) and different numbers of rules and (b) for different number of parts.

is much lower than that of Subgraph Isomorphism (using BB) and similar to that of the Neural Network.

7. WORST CASE COMPLEXITY

To study the worst case computational complexity of Rulegraph Matching, we give a theoretical analysis, based on the assumption that feature ordering and pruning methods have no effect. We give empirical results for tests using different numbers of parts to confirm the validity of our theoretical analysis.

The process of subgraph matching using Subgraph Isomorphism is equivalent to the maximum weighted clique problem[30] for which Tarjan and Trojanowski[2] have shown complexity to be $O(2^{v/3})$ where v, in our case, corresponds to the number of vertices or parts. This is consistent with the empirical values for the SI-BB system for different numbers of parts (Fig. 8(b)).

In Rulegraph Matching, primitive parts are replaced by rules and search for a maximal clique of rules is carried out in a space which is reduced in cardinality compared to that of Traditional Subgraph Isomorphism. Further, each rule in the model may only be paired with one and the same rule in the sample so that the worst case number of operations involved in this search will be $2^{r/3}$ for r being the number of unary rules, assuming that the A* search has no pruning effect. This determines the maximum possible size of the search tree and since each problem has an optimum number of rules this is constant for a given recognition problem.

The worst complexity of the lable compatibility checking method occurs when the feature values from every vertex map into both unary rules and the feature values from every edge map into every binary rule between these two unary rules. Under these conditions the existence of every edge in the model graph will have to be checked with the existence of every edge in the sample graph, resulting in γ^2 operations for γ being the number of edges. The overall worst case complexity of the Rulegraph Matching algorithm method thus is

$O(\gamma^2 2^{r/3})$. This is consistent with the results for Rulegraph Matching shown in Fig. 8(b) where the number of existence checks is seen to grow as a quadratic function of the number of edges (the number of vertices and edges were approximately equal).

It should be pointed out that our analysis is conservative in so far as it assumes that the label compatibility checking method will behave in the worst case at each node of the search. Further work is required to establish the complexity of the Rulegraph Matching method when used with the automatically learned rules and this may require a probabilistic analysis similar to that of Grimson[13] for the case of Interpretation Trees.

Based on these above analysis alone, Rulegraph Matching has a worst case complexity of $O(\gamma^2 2^{r/3})$ compared to that of Traditional Subgraph Isomorphism with $O(2^{v/3})$ and $O(r^2)$ for that of the EBS. This indicates that the Rulegraph Matching is capable of matching very large graphs if constant numbers of rules are used.

8. A 3D OBJECT RECOGNITION EXAMPLE

We have also compared the Rulegraph system with the EBS-NNet system for recognizing synthetic (CAD-generated) 3D objects where the learning of their relational structures is attained through a finite number of views. The segmentation and feature extraction stages are identical to those used in the Evidence-Based Object Recognition System used by Caelli and Dreier[6,7] and are summarized as follows. The input data for the model construction stage consisted of view-dependent depth maps. View-dependent input samples are chosen in order to restrict the computations of surface curvatures, or pixels, to what is visible (see Fig. 9). The segmentation procedure uses the zero-crossings of the determinant of the Hessian

$$f_{xx}f_{yy} - f_{xy}^2 \tag{1}$$

and determines convex, concave and planar regions in a way which minimizes noise amplification which typically occurs when full H and/or K zero-crossings are evaluated (Yokoya and Levine, 1989).[32] Such a segmentation procedure applies equally to training and sample data and is invariant to rigid motions. Once these parts are extracted then part attributes (unary features) and relations (binary features) can be computed.

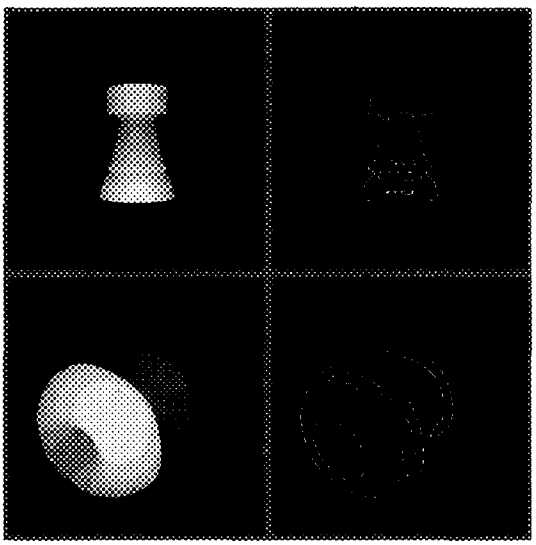


Fig. 9. Left: range data is shown for two 3D objects. Here, intensity corresponds, inversely, to depth. Right: segmentation is shown in terms of convex, concave and planar region types (from zero-crossings of Gaussian curvature). Surfaces were smoothed by an isotropic Gaussian filter with $\sigma = 6$ pixels for 256×256 pixel images before determining the derivatives (and the determinant of the Hessian, see equation (1); from Caelli and Dreier, 1994).

The Rulegraph and EBS-NNet Systems were tested using a database of six objects (see Fig. 10): the 3D Objects database, each with 24 views defined over equal angular steps in azimuth and elevation of a view-sphere. In addition to these 144 views, an extra 84 new views were generated (14 for each object) where each new view was oriented half-way between training set views.

Segmentation, feature extraction and rule generation was performed on each view of the training data. In this case the unary features were areas and 3D spanning distances and the binary features were centroid and maximum distances. Results obtained for the TS and with new SS views are shown in Fig. 11 (For a more detailed analysis of features see Caelli and Dreier, 1994[6,7]).

The Rulegraph system achieves perfect classification on the TS with less rules than the EBS-NNet system. Again, label-compatibility checking allows for the use of more general rules (larger feature value ranges) leading to improved classification for the new views in the SS. Here, the number of features and training views were reduced with respect to what was necessary for perfect classification of the new sample views. Only two unary features (area and maximum span) and two binary features (centroid distance and maximum distance) were extracted and a reduced number of views was used. Increasing the number of training views and/or the kinds of features extracted also increases classification performance for the new sample views. Indeed, Caelli and Dreier[6,7] achieved near perfect classification performance when sufficient views and features are used, particularly if the features are invariant to pose.

Label-compatibility checks allow for the use of more general rules which allows for larger numbers of object classes to be learned from fewer training views. In addition, the representation offers improved uniqueness when objects share similar parts and relations.

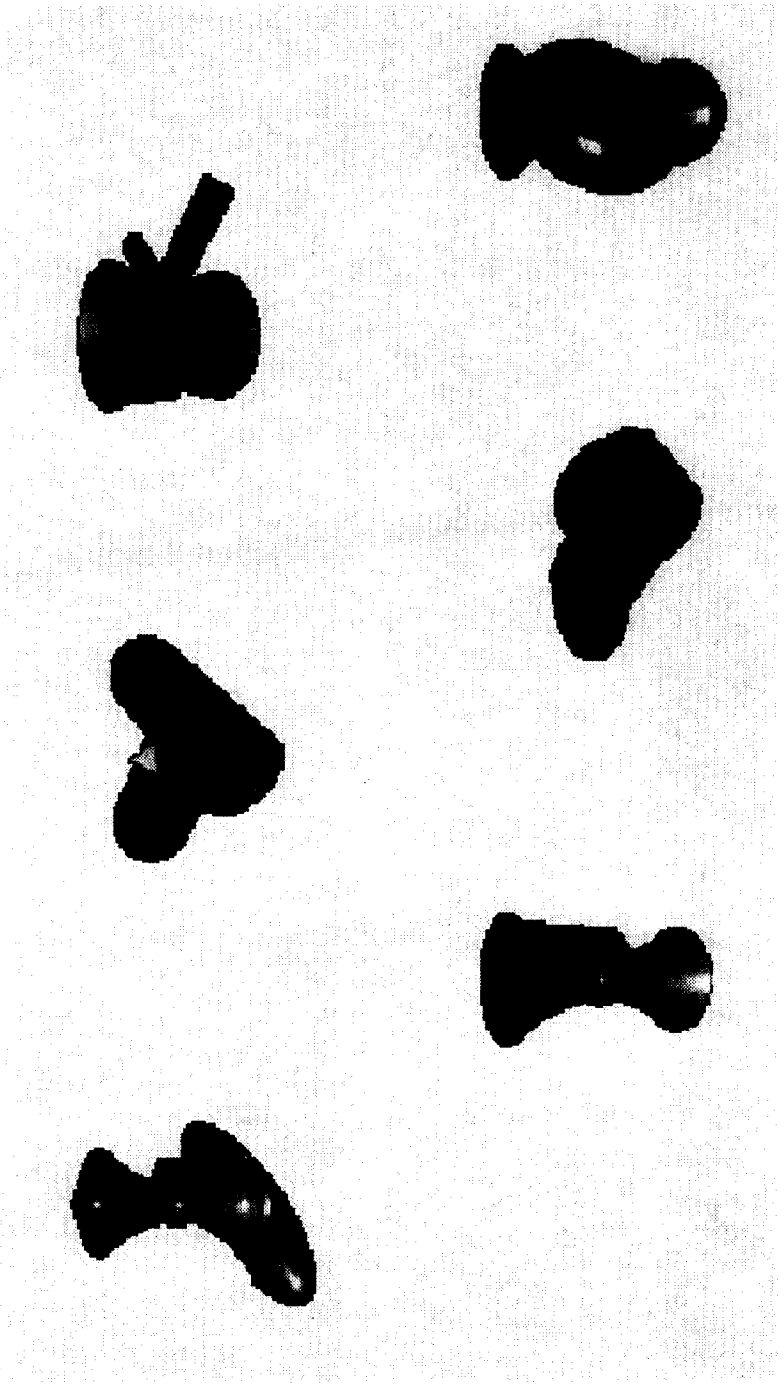


Fig. 10. Rendered example views for each of the objects in the 3D objects database. There were 24 views per object in the Training Set (TS) of images and 14 (new) views in the Sample Set (SS) of images (from Caelli and Dreier, 1994).

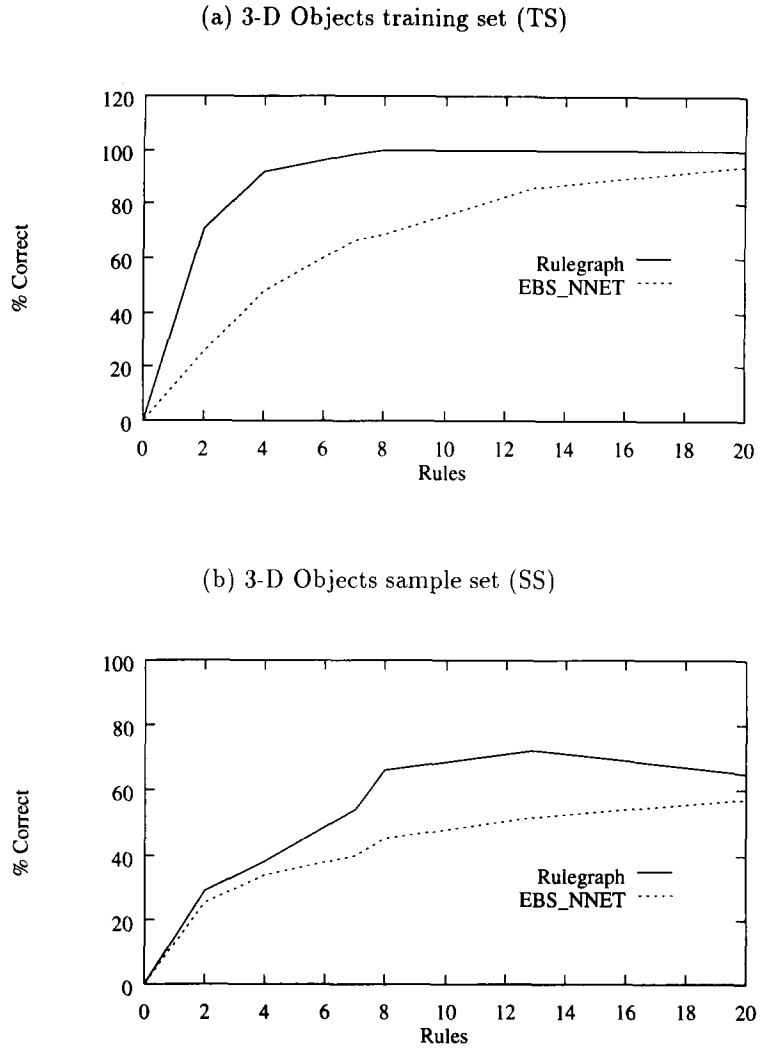


Fig. 11. (a) Classification performance is shown for the 3D Object database for different numbers of rules in the Training Set (TS), and (b) (new) views in the Sample Set (SS).

9. DISCUSSION

One problem with evaluating such 2D and 3D learning/recognition procedures is that of developing an objective definition of problem difficulty. In this work we have adapted a slightly different approach—we have analyzed the complexity of the algorithms, particularly as a function of the number of rules. The reason for this is relatively clear. Learning/recognition can be attained if we have enough features, rules and computational power. Consequently, it can be agreed that what makes one approach an improvement on others is whether it can achieve the same (or better) performance—in this case generalization and classification accuracy—with less operations. The rulegraph system described here uses properties of EBSs and Graph Matching/Search techniques to enable efficient matching search processes within a representation which retains pattern (relational) uniqueness and generalization from training data.

Rulegraph matching derives its power from a combination of several components. First, the computational complexity is reduced compared to traditional graph matching techniques through the replacement of pattern parts by rules. Second, the use of label-compatibility checking between model and sample rules leads to an improved representational uniqueness. Third, the search space of the matching process can be pruned significantly by embedding rulegraph matching into an evidential framework. That is, the EBS weights provide a *weighted* graph which enables us to use A* search to find the best match. The superior performance of rulegraph matching has several important implications for the pattern matching process, apart from reduced complexity and improved classification performance.

Rulegraphs offer an improved way of learning from training patterns. No model-based prior knowledge is required about pattern classes and no strict control is required over the presentation of patterns at training

time. Hence rulegraph matching may prove useful in situations where training data is limited or incomplete. The classification performance of rulegraphs is high even when small numbers of rules are generated. Rulegraph matching thus allows pattern recognition in situations where only partial or distorted data is available at classification time, as is the case in 3D object recognition. The reason for this is clear—the *conjunction* of broad attribute bands (evidence) reduces possible candidates and the label-compatibility checking further enacts this conjunctive process.

For Object and Pattern Recognition new benefits of Rulegraphs have become apparent. First, Rulegraphs—like Graph Matching—are capable of returning information in addition to class membership. The orientation of the input data with respect to the sample domain and number and identity of parts are also returned. Indeed, the Rulegraph system has been shown to provide information necessary for systems performing analyses at a symbolic level for the case of scene interpretation.[33,34] Second, a trade-off is possible between the uniqueness of the matching results and computational cost through choosing different numbers of rules. For any particular problem an optimum number of rules will produce best classification performance and hence, as the number of parts in a graph increase, the complexity grows only quadratically. This opens the possibility for matching graphs with larger numbers of parts than has been previously possible with Traditional Subgraph Isomorphism.

Finally, in Rulegraph Matching, learning is done posterior to rule generation in so far as label compatibilities and probability information is collected after the rules have been generated. The possibility of using label-compatibility information prior to this process—in the very generation of the rules—has also been recently addressed using a Conditional Rule Generation technique.[35,36] The integration of Conditional Rule Generation and Rulegraphs into a single framework of Relational Evidence Theory is the subject of ongoing investigation.

In summary, Rulegraphs capitalize on the efficiency and generalization components of Evidence-Based Learning and the relational structures of Graph Matching. Together they permit an application of A* search to the graph matching problem and a reduction in the cardinality of the matching process.

Acknowledgements—Adrian Pearce was supported by an Australian Postgraduate Research Award. Terry Caelli and Walter F. Bischof were supported by a grant from the Australian Research Council and Center for Intelligent Decision Systems. Walter F. Bischof was supported by grant OGP38251 from the Canadian Natural Sciences and Engineering Research Council.

REFERENCES

1. L. G. Shapiro and R. M. Haralick, Structural descriptions and inexact matching, *IEEE Trans. Pattern Analysis Mach. Intell.* **3**(5), 504–519 (September 1981).
2. R. E. Tarjan and A. E. Trojanowski, Finding a maximum independent set, *SIAM J. Computing* **6**(3), 537–546 (1977).
3. M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, New York (1979).
4. B. G. Buchanan and E. H. Shortliffe, *Rule-Based Expert Systems—The MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley (1984).
5. J. Gashnig, R. O. Duda and P. Hart, Model design in the prospector consultant system for mineral exploration, In *Expert Systems for the Microelectronic Age* (D. Michie, Ed.) Scotland: Edinburgh University Press, Scotland (1979).
6. T. Caelli and A. Dreier, Variations on the evidence-based object recognition theme, *Proc. 11th IAPRA Int. Conf. on Pattern Recognition* pp. 450–454 (1992).
7. T. Caelli and A. Dreier, Variations on the evidence-based object recognition theme, *Pattern Recognition* **27**, 185–204 (1994).
8. P. J. Flynn and A. K. Jain, Cad-based computer vision: From cad models to relational graphs, *IEEE Trans. Pattern Analysis Mach. Intell.* **13**(2), 114–132 (February 1991).
9. J. R. Quinlan, Decision trees and decision making, *Trans. Systems Man Cybernetics* **20**(2), 339–346 (April 1990).
10. M. A. Eshera and King-Sun Fu, An image understanding system using attributed symbolic representation and inexact graph-matching, *IEEE Trans. Pattern Analysis Mach. Intell.* **8**(5), 604–618 (September 1986).
11. P. Cheeseman, R. Kanefsky and W. M. Taylor, Where the really hard problems are, *Proc. 12th Int. J. Conf. on Artificial Intell.* pp. 331–337 (1991).
12. Patrick J. Flynn and Anil K. Jain, 3D object recognition using invariant feature indexing of interpretation tables, *Comput. Vision Graphics Image Process.: Image Understanding* **55**(2), 119–129 (1992).
13. W. E. L. Grimson, *Object Recognition By Computer: The Role of Geometric Constraints.* The MIT Press (1990).
14. E. K. Wong, Model matching in robot vision by subgraph isomorphism, *Pattern Recognition* **25**(3), (1992).
15. P. M. Pardalos and G. P. Rodgers, Computational aspects of a branch and bound algorithm for quadratic zero-one programming, *Computing* **45**, 131–144 (1990).
16. R. Carraghan and P. M. Pardalos, An exact algorithm for the maximum clique problem, *Operations Research Letts.* **9**, 375–382 (November 1990).
17. A. Rosenfeld, R. A. Hummel and S. W. Zucker, Scene labeling by relaxation operations, *Trans. Systems Man Cybernetics* **SMC-6**(6), 420–433 (June 1976).
18. L. Kitchen and A. Rosenfeld, Discrete relaxation for matching relational structures, *Trans. Systems Man Cybernetics* **SMC-9**(12), 869–874 (December 1979).
19. Whoi-Yul Kim and A. C. Kak, 3-D object recognition using bipartite matching embedded in discrete relaxation, *IEEE Trans. Pattern Analysis Mach. Intell.* **13**(3), 224–251 (March 1991).
20. E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines.* Wiley, New York (1989).
21. A. K. Jain and R. Hoffman, Evidence-based recognition of 3-D objects, *IEEE Trans. Pattern Analysis Mach. Intell.* **10**(6), 783–801 (1988).
22. John A. Hartigan. *Clustering Algorithms.* Wiley, New York (1971).
23. A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data.* Prentice-Hall, Englewood Cliffs, New Jersey (1988).
24. J. R. Quinlan, Learning logical definitions from relations, *Machine Learning*, **5**, 239–266 (1990).
25. Keung-Chi Ng and B. Abramson, Uncertainty management in expert systems, *IEEE Expert*, 29–48 (April 1990).
26. T. Caelli and A. Pennington, An improved rule generation method for evidenced-based classification systems, *Pattern Recognition* **26**(5), 733–740 (1993).
27. R. P. Lippmann, An introduction to computing with neural nets, *IEEE ASSP Magazine*, 4–22 (April 1987).
28. J. Pearl, *Probabilistic Reasoning in Intelligent Systems:*

Networks of Plausible Inference. Morgan Kaufmann (1988).
29. S. J. Henkind and M. C. Harrison, An analysis of four uncertainty calculi, *Trans. Systems Man Cybernetics* **18**(5), 700–714 (1988).
30. H. G. Barrow and R. M. Burstall, Subgraph isomorphism, matching relational structures and maximal cliques, *Inform. Process. Letts.* **4**(4), 83–84 (January 1976).
31. P. H. Winston, *Artificial Intelligence.* Addison-Wesley (1984).
32. N. Yokoya and M. Levine, Range image segmentation based on differential geometry: A hybrid approach, *IEEE Trans. Pattern Analysis Mach. Intell.* **11**, 643–649 (1989).
33. S. Dance and T. Caelli, A symbolic object-oriented picture interpretation network: SOO-PIN. In *Advances in Structural and Syntactic Pattern Recognition, Proceedings of the International Workshop* (Horst bunke, Ed.) Series on Machine Perception and Artificial Intelligence, Bern, Switzerland, spring 1993. International Association for Pattern Recognition, World Scientific Publishing Co.
34. S. Dance and T. Caelli, On the symbolic interpretation of traffic scenes, *ACCV93 Proc. Asian Conf. on Computer Vision* pp. 798–801, Osaka, Japan, (November 1993).
35. W. F. Bischof and T. Caelli, Learning structural descriptions of patterns: A new technique for conditional clustering and rule generation, *Pattern Recognition* (in press).
36. W. F. Bischof and T. Caelli, Visual learning of patterns and objects, 1994. (Submitted.)
37. L. G. Shapiro and R. M. Haralick, A metric for comparing relational descriptions, *IEEE Trans. Pattern Analysis Mach. Intell.* **7**(1), 90–94 (January 1985).

APPENDIX A

Label Compatibility Checking Method algorithm

The Label Compatibility Checking Method operates in two steps using the Compatibility_check() and Update_state() functions. First, the compatibility of a candidate rule is checked against the current clique. Second, if the candidate rule is compatible then the matching state is updated.

The Compatibility_check() function checks the existence of edges in the sample with respect to their existence in the model. This process is carried out for edges between the candidate rule and each rule in the clique separately. If an edge exists in the model between a unary candidate rule and a unary rule in the clique then at least one edge must exist in the sample to validate a label mapping. Compatibility occurs when label mappings are possible for all unary rules. Labels are not mapped if and only if they are not present in any mapping state.

The Update_state() function adds the candidate rule to the clique and updates the label mapping states between the model and the sample. The mapping states are updated by instantiation and elimination and the binary rules are used to update labels in decreasing order of evidence weights from the edges. In order to allow backtracking during search the new mapping states for the extension of the present clique must be saved as separate search states along with the set of rules in the clique (matched) and candidate rules (unmatched).

```
/* Determines the compatibility of candidate rule R_c^u with the clique */
/* and itself, then sets candidate label mapping states. */
Compatibility_check(){
  for all R_i^u where R_i^u ∈ clique OR R_c^u = R_i^u {
    at_least_one = false; model_edge_exists = false; sample_edge_exists = false;
    for all model labels L_1 where (L_1 ∈ R_c^u){
    for all model labels L_2 where (L_2 ∈ R_i^u){
        model_edge_exists = true;
        for all sample labels K_1 where (K_1 ∈ R_c^u){
        for all sample labels K_2 where (K_2 ∈ R_i^u){
          sample_edge_exists = true;
          if (((L_1 and K_1 are not mapped) OR (mapping(L_1 → K_1)))
          AND (mapping(L_2 → K_2) OR (L_2 and K_2 are not mapped)))
          then {
            if (∃R_j^b:((L_1L_2 ∈ R_j^b) AND (K_1K_2 ∈ R_j^b))
            then {
              set cand_mapping(L_1 → K_1);
              set cand_mapping(L_2 → K_2);
              at_least_one = true;
            }
          }
        }
        }
      }
      }
      if ((model_edge_exists) AND (sample_edge_exists)
      AND (NOT at_least_one)) then return(false);
    }
    return(true);
}
/* Adds candidate rule R_c^u to the clique and updates the label mapping states */
/* with the candidate label mapping states by elimination and instantiation. */
Update_state(){
  add R_c^u to clique;
  for all R_i^b in decreasing order of evidence weight w(R_i^b){
    for all model labels L where (∃X:LX ∈ R_i^b){
      for all sample labels K where (∃Y:KY ∈ R_i^b){
        if (L and K are not mapped) then
```

```
            if (cand_mapping(L→K) OR mapping(L→K) then
            set new_mapping (L→K) else clear new_mapping(L→K);
          else
            if (cand_mapping(L→K) AND mapping (L→K) then
            set new_mapping(L→K) else clear new_mapping(L→K);
        }
      }
      update mapping states with new_mapping states;
    }
  }
```

APPENDIX B

The relational evidence metric

A Bayesian framework[28,29] is used to evaluate the matching. Evidence weights for rules are derived from the posterior probabilities of each rule given the class $w(R_i|\text{class})$. These are simply calculated from the frequencies of points for each class which activate each rule (see Fig. 2(b)).

A relational metric is used to sum evidence from unary and binary rules. This is based on the relational distance metric scheme used by Shapiro and Haralick,[1,37] except that the graphs being compared are graphs of rules, not graphs of parts. The evidence weight metric for the match of a sample to a class model is computed in two steps.

First, the weight for each rule is adjusted by the proportion of the number of labels in the sample to the number of labels in the model, provided this is less or equal than 1. Assume there are v single labels and γ label pairs present in the current label mapping states. This results in normalized unary evidence $U_i(R_i^u|\text{class})$ from rule R_i^u and binary evidence $B_i(R_i^b|\text{class})$ for rule R_i^b for the model class.

$$\mathrm{U}(R_i^u|\text{class}) = w(R_i^u|\text{class}) \min\left(1, \frac{\Sigma_{v_j \in \text{sample} \wedge v_j \in R_i^u} v_j}{\Sigma_{v_k \in \text{model} \wedge v_k \in R_i^u} v_k}\right)$$

$$\mathrm{B}(R_i^b|\text{class}) = w(R_i^b|\text{class}) \min\left(1, \frac{\Sigma_{\gamma_j \in \text{sample} \wedge \gamma_j \in R_i^b} \gamma_j}{\Sigma_{\gamma_k \in \text{model} \wedge \gamma_k \in R_i^b} \gamma_k}\right).$$

Second, independence is assumed between unary and binary evidence within a label-compatible interpretation clique. The evidence weights for the unary rules are multiplied by the proportion of the connecting binary rule weights and the binary rules are thus considered supporting evidence for each unary rule in the evidence summation. The result of the summation is then normalized by dividing by the union of evidence from rules activated by labels and label pairs—$R \in$ *sample* in the sample and $R \in$ *model* in the model. This returns an evidence weight between in the interval [0, 1]

$$w(\text{class}|\text{clique}) = \frac{\Sigma_{R_i^u \in \text{clique}} \mathrm{U}(R_i^u|\text{class}) \dfrac{\Sigma_{R_j^b \in \text{clique}} \mathrm{B}(R_j^b|\text{class})}{\Sigma_{R_j^b \in \text{sample} \wedge R_j^b \in \text{model}} w(R_j^b\text{class})}}{\Sigma_{R_i^u \in \text{sample} \wedge R_i^u \in \text{model}} w(R_i^u|\text{class})}$$

About the Author—ADRIAN R. PEARCE is a doctoral student in computer science at The University of Melbourne and funded by an Australian Post Graduate Research Award. His interests lie in the application of machine learning to visual pattern and object recognition.

About the Author—TERRY CAELLI received his Ph.D. from the University of Newcastle in 1975. Since then he has been active in research in the areas of human and machine vision. As Professor of Computer Science at The University of Melbourne his interests focused on human and machine pattern/object recognition and Machine Learning. He has recently moved to Curtin University of Technology where he is Head of the Computer Science Department.

About the Author—WALTER F. BISCHOF received the Ph.D. degree in Psychology in 1982 from the University of Bern, Switzerland. He is currently Associate Professor of Psychology and Adjunct Professor in Computer Science at the University of Alberta. His research interests are equally distributed between human and computer vision. In human vision, he works on early spatio-temporal analysis in the visual system with an emphasis on motion analysis. In computer vision, his interests include shape-from methods, object recognition and machine learning.