Visual Learning: An Overview

Walter F. Bischof
Department of Computing Science
University of Alberta
Edmonton, T6G 2E8
Canada
Email: wfb@ualberta.ca

Abstract

We review modern approaches to the learning and recognition of complex patterns, including discriminant functions, neural networks, decision trees, and hidden Markov models. We then introduce several relational learning systems and discuss in detail one specific technique, conditional rule generation. This technique is shown to be very flexible and useful for the learning of static patterns, such as objects, as well as dynamic patterns, such as movement patterns. The technique is illustrated with a number of very difficult visual learning problems.

Keywords

Visual learning, neural networks, decision trees, relational learning, conditional rule generation

1 Introduction

Visual learning is concerned with creating systems that learn to analyze and interpret images, both static images and images that change over time. Visual learning shares many ideas with machine learning (Briscoe & Caelli 1996; Mitchell 1997), but many of the symbolic machine learning techniques cannot be used in visual learning because of the massive amounts of data that have to be dealt with in images. It also shares many ideas with pattern recognition (Duda et al. 2001), but puts a much stronger emphasis on learning rather than the design of efficient recognition procedures.

The aim of visual learning is to generate rules for recognizing and interpreting image data, for example rules for determining that a certain image region is a human face, or that a certain sequence of images shows a person lifting a heavy object. The process of labeling image data can be defined by criteria that can either be known *explicitly* or can be encoded *implicitly* in the specific algorithms used in a system. In both cases, learning techniques are useful for making generalizations from training data (e.g. predicting labels for newly observed image data), for estimating parameters of sub-processes (e.g. parameters controlling image segmentation), or for optimizing search and labeling procedures.

Machine learning covers problems in inductive and deductive learning, where induction refers to the process of generalizing from known examples, while deduction refers to learning to reason about data from given models. Most applications in visual learning involve inductive learning. In the following, we provide a brief overview over the more popular approaches. A representative sample of general inductive learning techniques used in visual learning is shown in Table 1 (Caelli & Bischof 1997). The techniques can be grouped into “supervised“ and “unsupervised“ learning. In supervised learning, a learning system is presented with both, training data and classification labels, and the goal is to find a set of rules or procedures for classifying the training examples. In unsupervised learning, on the other hand, training examples are not pre-classified, and the goal is to find regularities in the data, regularities that are implicitly defined in the learning and recognition procedures.

Unsupervised Learning		
	Model	Applications
Parametric	Bayesian classifiers Radial basis functions	Segmentation Image synthesis
Non-parametric	K-means clustering Self-organizing maps Vector quantization Conceptual clustering	Feature grouping Segmentation Image compression Visual taxonomies

Supervised Learning		
	Model	Applications
Feature-vector learning	Discriminant functions Neural Networks Decision trees	Classification Recognition Feature extraction
Relational learning	Evidence-based Systems Conditional rule generation Symbolic learners	Recognition by parts Symbolic descriptions

Table 1: Overview of techniques used in visual learning

1.1 Unsupervised learning

Classic examples of unsupervised learning models include clustering techniques where input data is grouped into sets using attribute proximity (Jain & Dubes 1988). The result of the grouping process depends on the characteristics of the proximity models and the parameterization. Parametric techniques include, for example, parametric Bayesian clustering where probabilistic cluster membership is modeled using multivariate Gaussian functions. In this case, clustering involves determining position and shape parameters of the best fitting functions for modeling the clusters, and the algorithms for finding them typically use gradient descent with respect to the function parameters.

Parametric clustering techniques have several drawbacks, including the complexity of the search procedure required, the lack of unique solutions and the fact that the modeling functions used may not fit the data. For these reasons, non-parametric clustering techniques have been more popular. Here, clustering is aimed at partitioning data such that within-cluster distances are minimized and between-cluster distances are maximized. The methods shown in Table 1 are all aimed at achieving this goal. For the K-means method, the formulation is direct in the sense that the search for clusters is guided by the mini-

max constraint (minimizing within-cluster distances and maximizing between-cluster distances). Kohonen maps enact a similar process through the formation of attractors in the data-attribute space (Pao 1989), in vector quantization clustering is achieved through binning techniques while in conceptual clustering systems partitions are created in attribute space by maximizing category utility measures (Briscoe & Caelli 1996).

1.2 Supervised learning

Supervised learning differs from unsupervised in so far as criteria for labelling data (e.g. classification labels) are known explicitly. Given training data described in terms of a set of features (attributes) and their class labels, the goal of supervised learning is to find a (simple) partitioning of the feature (attribute) space that allows correct classification of all training data, as well as generalization from training data to unseen, similar data.

Supervised learning methods differ with respect to the way they represent patterns and the way they partition feature spaces and thus the generalizations they produce. Supervised learning methods can be grouped into two major groups, here referred to as *feature-vector learning* and *relational learning*. In feature vector learning, each pattern is described by a vector of features that characterizes the pattern as a whole, and thus each pattern is represented as a point in a multi-dimensional feature space. In relational learning, on the other hand, each pattern is considered as being composed of multiple parts, and is thus described by feature vectors characterizing each pattern part and each part relation. Both groups are described in detail in the following sections.

1.3 Problems in Visual Learning

Whatever the approach is taken by a visual learning method, is has to address a number of questions:

Recognition of partial data: Many real image interpretation problems involve the recognition of patterns from partial data. We claim that in order to optimize this process, rule generation and recognition methods are required that are based on a “recognition-by-parts" approach. In this approach, pattern fragments or individual pattern parts (and their associated attributes) are learned to provide sufficient evidence for the classification of patterns.

Recognition of distorted data: Most image interpretation problems involve identification of known classes or patterns in data which is not identical to any of the training data. Solutions involve the ability of learning systems to generalize from training data. The problem is to optimize generalization while retaining recognition accuracy.

Finding patterns in complex data: One of the more difficult problems in image interpretation is that of finding patterns (or objects) in complex multi-object scenes. Solutions must involve classification of distinct pattern regions in an efficient way.

Correspondence between data parts and model parts: Beyond a mere classification of patterns or objects, it may be required to determine the correspondence between the parts of observed image data and those of stored models. This may be required to determine the pose of objects or in order to predict the existence of (hidden) pattern parts. Solutions to this problem depend on whether learning or encoding training data also includes pattern or object part labels. Part-indexed data structures used in relational learning implicitly solve the pose problem while simple attribute-indexed structures used in feature-vector learning do not. The latter may have to be supplemented by methods for additional hypothesis testing or model projection to solve the pose problem.

2 Feature-Vector Learning

In feature vector learning, patterns are described by a vector of features that describes the patterns as a whole. In this section, we introduce two main representatives of this approach, namely discriminant functions (and a modern incarnation in terms of neural networks) and decision trees.

2.1 Discriminant Functions and neural networks

The non-parametric statistical literature abounds with methods for classifying patterns (Devijver & Kittler 1980). Here, we introduce linear discriminant functions and then discuss briefly modern implementations in terms of neural networks. Given a set of patterns, each described by a set of continuous attributes $a_1, a_2, ..., a_N$, linear discriminant functions are used to separate the patterns by a linear decision boundary. A linear discriminant function can be written as a linear combination of components

$$g(\mathbf{a}) = w_0 + w \cdot a = w_0 + w_1 a_1 + w_2 a_2 + \ldots$$

where $g(\mathbf{a})$ is the discriminant function, $\mathbf{w}$ a weight vector, $\mathbf{a}$ a feature (or attribute) vector, and w_0 a threshold weight. The equation $g(\mathbf{a})=0$ defines a decision boundary, and the two regions defined by $g(\mathbf{a})<0$ and $g(\mathbf{a})>0$ define the two pattern classes, respectively. The weight coefficients are typically found using a gradient descent method.

In the case of multiple pattern classes, the feature space must be partitioned into multiple partitions, ideally each one associated with a single class. Using discriminant functions, this can be achieved in one of several ways. For K classes, one can, for example, use $K-1$ discriminant functions separating regions ω_i from not(ω_i), or one can use $K(K-1)/2$ discriminant functions separating ω_i from ω_j, and finally, one can define a linear machine that assigns $\mathbf{a}$ to ω_i if $g_i(\mathbf{a})>g_j(\mathbf{a})$ for all $i \neq j$. Other combination schemes are possible, and their usefulness depends precisely on the distribution of pattern classes in feature space.

More recently, neural networks, and in particular, multi-layered perceptrons have become very popular for pattern learning (Hertz et al. 1991). A closer inspection reveals a very close relationship between these networks and discriminant functions, so neural networks are comparable in their performance to discriminant functions.

This is illustrated with the following simple neural network, consisting of an input layer I, and hidden layer H, and an output layer O. A pattern with features $a_1, a_2, \ldots, a_N$ is presented at the input nodes I, and the output nodes correspond to the classifications $c_1, c_2, \ldots, c_K$. The activity of the hidden units H_i is defined as

$$H_i = f\left(\sum_j w_{ij}^H I_j\right)$$

where w_{ij}^H is the weight of the j-th input of the i-th hidden unit, and f is a non-linear function, typically a sigmoid function. The activity of the output units is defined similarly, namely

$$O_i = f\left(\sum_j w_{ij}^O H_j\right),$$

where w_{ij}^O and f are defined as before. The expressions

$$\sum_j w_{ij}^H I_j \quad \text{and} \quad \sum_j w_{ij}^O H_j$$

are discriminant functions in the input space and the hidden layer space, respectively. An analysis of learning in neural networks shows that they are typically equivalent to simple gradient descent methods.

In these approaches to pattern learning, patterns are represented by characteristic feature vectors $\boldsymbol{a}$. Pattern learning consists of finding a partitioning of the feature space such that, ideally, each partition corresponds to a single pattern class, and pattern classification is based on combinations of discriminant functions.

2.2 Decision Trees

Decision trees have become very popular for inductive learning of patterns (Quinlan 1992). Given a number of patterns, each described by a set of attributes $a_1, a_2, \ldots, a_N$, and a partitioning of the patterns into a number of classes $c_1, c_2, \ldots, c_K$, the decision tree method is aimed at ordering the patterns in a tree such that each tree node specifies a test of some attribute, and each child corresponds to a possible value of the attribute. Each leaf node of the decision tree corresponds to a set of patterns belonging to a single class. A path from the root of the tree to a leaf thus represents a hierarchically ordered sequence of decisions for classifying the patterns.

Figure 1 illustrates a one decision tree generated for the data shown in Table 2. In this example, a decision is made whether the weather conditions are suitable for playing tennis. Each of the training examples is described by four attributes, outlook, temperature, humidity and wind, and each training instance belongs to one of two classes, yes or no. Classification of each example is achieved by traveling through the decision tree from the root to a leaf. At each node, a particular attribute is tested, and, depending on the value of the attribute, a different branch is selected. Leaf nodes are either associated with patterns of a single class, or tree expansion was stopped (Quinlan 1992). For day 4, for example, the rightmost path through the decision tree would be followed, and for day 8 the leftmost.

Day	Outlook	Temperature	Humidity	Wind	Play
1	Sunny	hot	high	weak	no
2	Sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	Rain	mild	high	weak	yes
5	Rain	cool	normal	weak	yes
6	Rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	Sunny	mild	high	weak	no
9	Sunny	cool	normal	weak	yes
10	Rain	mild	normal	weak	yes
11	Sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	Rain	mild	high	strong	no

Table 2: Decision tree data. Each line describes weather conditions and whether tennis should or should not be played. Each pattern is described by four features (outlook, temperature, humidity, and wind) and belongs to one of two classes (play or do not play tennis).

Figure 1: Example of a decision tree generated for a the data shown in table 2. A pattern is classified by traveling through the decision tree from the root to a leaf. At each node of the tree, a particular attribute is tested (indicated by a box), and, depending on the value of the attribute, a particular child in the decision tree is selected. Each leaf node is associated with a single class, in this case the class yes / no.

For each training set, a number of decision trees can be generated, and the problem is to find a decision tree that is as shallow as possible, so that classification of patterns can be achieved in a small number of decisions. To this end, one should select at each node the attribute that is most useful for classifying all the patterns associated with the node. Attribute selection is typically based on classification entropy *H(S)* of a node S, defined as

$$H(S) = -\sum_{k=1}^{K} p_k \log_2 p_k$$

where p_k is the proportion of elements of node S in class c_k. So, for example, in the decision tree in Figure 1, there are 5 elements associated with the node labeled “wind” (namely elements 4,5,6,10, and 14), three in class "yes" and two in class "no". Hence the classification entropy of that node is 0.971. On the basis of classification entropy, Quinlan (1992) introduced a number of selection criteria, including the information gain measure *G(S,A)*, defined as

$$G(S,A) = H(S) - \sum_{T \in children(S)} p_T H(T)$$

where p_T is the proportion of patterns that belong to child T. Using this measure, one chooses at each node of the decision tree the attribute that maximizes *G(S,A)*. The decision tree in Figure 1 was generated using this measure. In addition to *G(S,A)*, a number of other selection criteria have been introduced, but it is difficult to determine under what conditions each of the criteria leads to an optimal decision tree.

3 Relational Learning

In relational pattern recognition, patterns are considered as being composed of constituent parts, and pattern are described by (unary) part features, and (binary) features of part relations. These part and part relation features can be linked together into an attributed relational structure (graph) that defines a pattern uniquely. A simple example of a relational graph representation is show in Figure 2. In an attributed relational graph, each of the vertices is described by a unary feature vector and each of the arcs is described by a binary feature vector. More formally, an attributed relational graph is defined as a tupel $<P,R,U,B>$, where $P=\{p_i, i=1,...,n\}$ is a set of vertices (pattern parts), $R=\{r_{ij}, i=1,...,n, j=1,...,n\}$ a set of arcs (part relations), $U=\{u_i, i=1,...,n_U\}$ a set of vertex attributes (part attributes), and $B=\{b_i, i=1,...,n_B\}$ a set of arc attributes (part relation attributes).

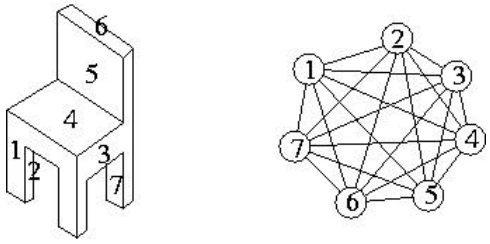

Figure 2: Example of a relational structure. The left hand shows a sketch of a chair consisting of seven image regions, the right hand side shows the corresponding relational structure. Each of the image regions (pattern parts) is represented by a vertex, and relations between image regions (pattern relations) are represented by arcs between vertices. Every vertex and arc is described by a set of features, corresponding to the features of the pattern parts and part relations.

Pattern classification is achieved using relational graph matching where a sample pattern is matched to a model pattern. This is achieved by searching for a label assignment (a mapping of vertices in the sample pattern to vertices in the model pattern) that maximizes some objective similarity function. Pattern classes are usually represented by enumerating all training (or model) instances, and classification of a sample pattern is achieved by searching through all model patterns to determine the one producing the best match. This representation and the associated graph matching approach have been the preferred architecture for object recognition (Flynn & Jain 1993).

The relational graph matching approach to pattern and object recognition has several problems. First, the computational complexity is exponential, i.e. matching time increases exponentially with the number of vertices in the sample and model graphs. Second, pattern generalizations are difficult to represent. One solution has been to represent pattern classes by a typical class member and a distance measure (Shapiro & Haralick 1982), but it is difficult to capture both, numerical feature differences and differences in the pattern structure into a single numerical similarity measure. Third, pattern learning has been considered only rarely in the literature. In most methods for generating relational structures for visual recognition, prior information about class membership is provided, and it is not clear how the model descriptions can be learned.

Below, we describe approaches that attempt to overcome these problems in different ways, by combining numerical learning techniques with limited structural pattern learning. The two examples described in

detail, evidence-based systems (EBS) and conditional rule generation (CRG), learn rules which are defined by regions in unary and binary feature spaces. The approaches differ in the way they ensure compatibility between unary and binary rules. In CRG, consistency between unary and binary features is explicitly represented, whereas in EBS this is done only implicitly.

3.1 Evidenced-Based Systems

EBS is aimed at learning to recognize visual patterns and objects in a supervised learning paradigm (Caelli & Dreier, 1994). Pattern descriptions are given in the form of attributed relational graphs, and the goal of EBS is produce a set of rules that, given a sample pattern, produce evidence for a particular pattern class. To this end, EBS proceeds in the following way:

First, all unary feature vectors $\boldsymbol{u}_i$ of all patterns are collected into a unary feature space U, and all binary feature vectors $\boldsymbol{b}_i$ are collected into a binary feature space B. In the context of 3D object recognition, unary features may include measures such as local surface curvature or perimeter of local patches, and binary features may include measures such as distances, angles or boundary relationships.

Both (unary and binary) feature spaces are then clustered using standard clustering algorithms, and the resulting clusters are bound by simple rectangles that are oriented along the feature space axes. The clusters need not be disjoint, and complex combinations of clusters could be constructed. For example, non-convex regions can be defined by rules which include some regions of feature space but exclude others. Each unary cluster U_i or binary cluster B_i can provide some evidence about the presence of a particular pattern class in the following way: If a sample pattern contains a part with unary features within the bound of U_i then there is certain evidence that the pattern belongs to each of the pattern classes $c_1, \ldots, c_K$. The same is true if a pattern contains a part relation with binary features within the bounds of a cluster B_i. With each cluster, an *evidence weight* $\boldsymbol{E}$ is associated which corresponds to the likelihood that activation of the rule contributes evidence for the existence of a particular pattern class.

The evidence vectors $\boldsymbol{E}$ could be derived from the relative frequencies of class samples in each cluster. However, EBS takes this one step further and uses supervised learning to solve the evidence estimation problem and, at the same time, to learn the relationships between unary and binary features. This relationship can be learned by

a neural network with input nodes *I* corresponding to clusters, output nodes *O* corresponding to classes, and with one hidden layer, with the number of nodes being the larger of the input or output node numbers (see Figure 3).

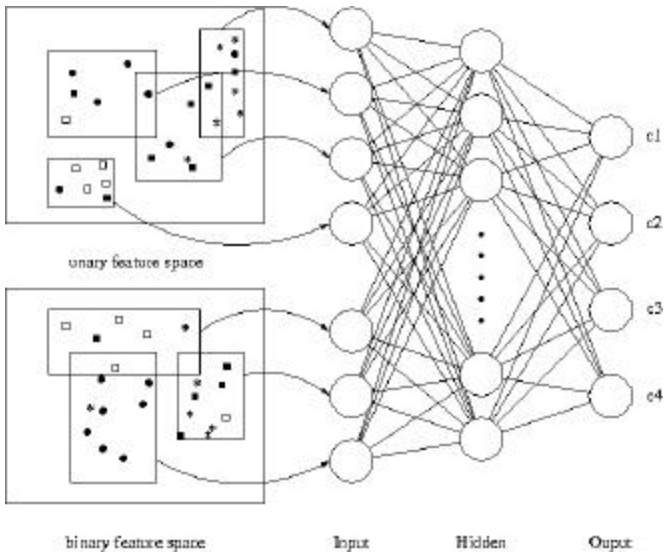

Figure 3: EBS System. The feature vectors of all pattern parts and part relations are collected into the unary and binary feature spaces, respectively. Membership of a feature vector in a cluster produces some evidence for the class membership of pattern. An optimal combination of evidence is learned by a neural network whose input nodes correspond to clusters in the feature spaces and whose output nodes correspond to classes.

The evidence weights are determined by the connections between input-hidden-output layer nodes. In particular, each hidden layer node is connected to every unary and binary rule. This allows for the reinforcement of co-occurrences between unary and binary feature values and thus allows implicit learning of relational structures. It does,

however, not guarantee label-compatibility in a strict sense since no specific model parts and relations are encoded in a given hidden unit.

The EBS approach differs from direct neural net implementations in two respects. First, feature space partitioning is not the same as that obtained with multi-layered perceptrons, and second, constraints are added on the hidden layer to determine evidence weights that are in accordance with a conjunctive rule form. For these reasons, the rules and weights are guaranteed to satisfy representational constraints, something which is not guaranteed in direct neural net implementations.

The main limitation of the EBS approach is that the representation is not unique, i.e. rules are generated without *explicitly* considering the relationships between specific unary and binary feature values that define specific objects. In other words, unary and binary rules are generated but not linked together into a label-compatible representation of each model. This is attained *implicitly* via the hidden units in the neural network (Figure 3) but it does not *guarantee* a unique representation of structural relations in the data. In the approach presented in the next section, this compatibility is explicitly built into the rule generation stage.

3.2 Conditional Rule Generation

We have explored methods for pattern learning that combine the expressiveness of symbolic machine learning techniques (inductive logic programming) with the generalization models of numerical machine learning (Bischof & Caelli 1994). This led to a class of numerical relational learners that induce over numerical attributes in ways which are constrained by relational pattern models. Our approach, CRG, generates rules that are linked together to satisfy relational constraints of the training data (see Figure 4). These relational constraints are introduced adaptively, i.e. they are added if they are required to improve the classification rules.

The goal of CRG is to derive a set of classification rules for pattern fragments on the basis of a set of training patterns in a supervised learning paradigm. Rather that work with fixed-size pattern fragments, CRG attempts to find the simplest classification rules, and then expands the rules to the extent that is required to achieve correct classification of all training data. Hence, for very simple patterns, CRG may produce classification rules that rely on isolated pattern parts only and thus are equivalent to rules produced by feature-vector learning. At

the other extreme, CRG may produce very complex classification rules that rely on features of all pattern parts and their relations, and thus may produce rules equivalent to those produced by the relational graph matching a pproach.

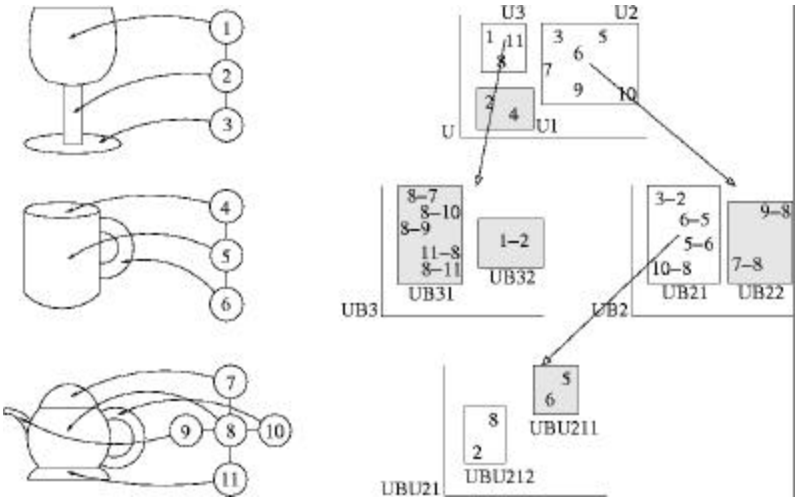

Figure 4: Example of input data and conditional cluster tree generated by CRG method. The left panel shows input data and the attributed relational structures generated for these data, where each vertex is described by a unary feature vector ***u*** *and each edge by a binary feature vector* ***b****. In this example, we assume that there are two unary and two binary features. We further assume that there are two pattern classes, class 1 consisting of the drinking glass and the mug, and class 2 consisting of the teapot. The right panel shows a cluster tree generated for the data on the left. Numbers refer to the vertices in the relational structures, rectangles indicate generated clusters (defined by lower and upper bounds on each feature), gray ones are unique, white one contain elements of multiple classes. Classification rules are derived directly from this tree.*

CRG produces classification rules for patterns described by an attributed relational graph where, as before, U_i are sets of unary features describing pattern parts and B_{ij} are sets of features describing part relations. For a set of patterns described by attributed relational

graphs, classification rules are generated for pattern fragments, i.e. for each chain of pattern parts $P_i - P_j - P_k ...$, CRG generates classification rules by partitioning the Cartesian feature product spaces $U_i ? B_{ij} ? U_j ? B_{jk} ? U_k ? ...$. This partitioning is done incrementally, starting with the unary feature space U_i and expanding it into conditional feature spaces B_{ij}, U_j, ... if it is required for correctly classifying the training data. Generation of a cluster tree proceeds as described in the following example (see Figure 4).

First, the unary features of all parts of all patterns are collected into a unary feature space U in which each point represents a single pattern part. The feature space U is partitioned into a number of clusters U_i. In the example in Figure 4, we assume that there are two unary features and that each cluster is defined by lower and upper bounds for each of these features. Some of these clusters may be unique with respect to class membership (e.g. cluster U_1) and provide a classification rule: If a pattern contains a part p_r whose unary features $u(p_r)$ satisfy the bounds of a unique cluster U_i then the pattern can be assigned a unique classification. The non-unique clusters contain parts from multiple pattern classes and have to be analyzed further. For every part of a non-unique cluster we collect the binary features of this part with all adjacent parts in the pattern to form a (conditional) binary feature space UB_i. The binary feature space is clustered into a number of clusters UB_{ij}. In the example in Figure 4, we assume that there are two binary features and, as before, each cluster is defined by lower and upper bounds for each of these features. Again, some clusters may be unique (e.g. clusters UB_{22} and UB_{31}) and provide a classification rule: 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 $b(p_r,p_s)$ of the pair $<p_r,p_s>$ satisfy the bounds of a unique cluster UB_{ij} then the pattern can be assigned a unique classification. For non-unique clusters, the unary features of the second part p_s are used to construct another unary feature space UBU_{ij} that is again clustered to produce clusters UBU_{ijk}. This expansion of the cluster tree continues until all classification rules are unique or a maximum rule length has been reached.

If there remain unresolved rules at the end of the expansion procedure (which is normally the case), the rules are split into more discriminating rules using an entropy-based splitting procedure. Consider the cluster tree in Figure 4 with the non-unique cluster UBU_{212}. One way to proceed would be to re-cluster feature space UBU_{21} into a larger number of clusters. Alternatively, one can simply split cluster

UBU_{212} along one of the feature dimensions. The latter method is used here.

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, $P_1=\{c|f(c)=T\}$ with n_1 elements and $P_2=\{c|f(c)>T\}$ with n_2 elements. We define the normalized partition entropy $H_P(T)$ as

$$H_P(T) = (n_1 H(P_1)+n_2 H(P_2))/(n_1+n_2).$$

The cut point T_F that minimizes $H_P(T_F)$ is considered the best point for splitting cluster C along feature dimension F. The best split of cluster C is considered the one along the feature dimension F that minimizes $H_P(T_F)$. Further, 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.

Cluster splitting continues until all clusters are unique or some termination criterion has been reached. This results in a tree of conditional feature spaces. Classification rules are derived directly from the final cluster tree. For the cluster tree shown in Figure 4, the classification rule derived for cluster UB_{32}, for example, can be described as follows: If there is a part p_i with unary features $\boldsymbol{u}(p_i)$ within the bounds defined by cluster U_3, and part p_i is connected to some other part p_j such that the binary features of their relation, $\boldsymbol{b}(p_i,p_j)$, are within the bounds defined by cluster UB_{32}, then the pattern fragment $p_i - p_j$ belongs to class 1. From the empirical class frequencies of all training patterns one can derive an expected classification (or evidence vector) $\boldsymbol{E}$ associated with each rule (e.g. $\boldsymbol{E}(UBU_{212})=[0.5,0.5]$, given that it contains one element of each class). Similarly, one can compute evidence vectors for partial rule instantiations, again from empirical class frequencies of non-terminal clusters (e.g. $\boldsymbol{E}(UB_{21})=[0.75,0.25]$). Hence an evidence vector $\boldsymbol{E}$ is available for every partial or complete rule instantiation.

Again, for the cluster tree in Figure 4, the classification rule derived from cluster UBU_{212} can be described as follows: If there is a part p_i with unary features $\boldsymbol{u}(p_i)$ within the bounds defined by cluster U_2, and

part p_i is connected to some other part p_j such that the binary features of their relation, $\boldsymbol{b}(p_i,p_j)$, are within the bounds defined by cluster UB_{21}, and part p_j has unary features $\boldsymbol{u}(p_i)$ within the bounds defined by cluster UBU_{212}, then the pattern fragment $p_i - p_j$ belongs to classes 1 and 2 with probabilities *[0.5,0.5]*.

CRG has been used successfully for the learning of objects and their recognition in complex scenes (Bischof & Caelli 1997a, 1997b). Many issues arise in the application of classification rules and in the combination of classification evidence by multiple rules. These are discussed in the next section, after learning of spatio-temporal patterns has been introduced.

4 Spatio-temporal Learning

We are now considering patterns that vary over time, such as, for example, a sequence of images of a person lifting a box, or an image sequence showing an arial view of a forest taken at weekly intervals. Our goal is to develop a system that can learn to recognize such spatio-temporal patterns in a supervised learning paradigm. The introduction of an additional (time) dimension leads to an increased data complexity with an ensuing need to increase the efficiency of learning and classification procedures.

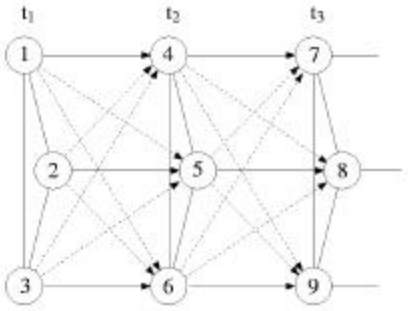

Figure 5: A spatio-temporal pattern consisting of three parts over three time-points. Undirected arcs indicate spatial binary connections, solid directed indicate temporal binary connections between the same part at different time-points, and dashed directed arcs indicate temporal binary connections between different parts at different time-points.

In general, a spatio-temporal pattern is defined by a set of labeled time-indexed attributed features. A pattern P_i is thus defined in terms of $P_i = \{p_{i1}(\boldsymbol{a}:t_{i1}), \ldots, p_{in}(\boldsymbol{a}:t_{in})\}$ where $p_{ij}(\boldsymbol{a}:t_{ij})$ corresponds to part j of pattern i with attributes $\boldsymbol{a}$ that are true at time t_{ij}. The attributes $\boldsymbol{a}$ are defined with respect to specific labeled features, and are either unary (single feature attributes) or binary (relational feature attributes, either over space or over space-time), that is, $\boldsymbol{a} = \{\boldsymbol{u}, \boldsymbol{b}_s, \boldsymbol{b}_t\}$ (see Figure 5). Examples of unary attributes $\boldsymbol{u}$ include area, brightness, position; spatial binary attributes $\boldsymbol{b}_s$ include distance, relative size; and temporal binary attributes $\boldsymbol{b}_t$ include changes in unary attributes over time, such as size, orientation change, or long range position change.

In the following, we discuss very briefly hidden Markov models, and then introduce a generalization of the CRG approach to the learning of spatio-temporal patterns.

4.1 Hidden Markov Models

Recently, hidden Markov models (HMMs, Rabiner & Juang 1993) have become very popular for modeling time series data in applications such as speech or gesture recognition, computational biology, and in computer vision applications such as object tracking and modeling of changes in images.

HMMs are used for modeling sequences of observations O_t over time. These observations are assumed to be generated by some process whose (discrete) states S are hidden from the observer. Further, the state of the hidden process is assumed to satisfy the Markov property: the process state at time t, S_t, depends only on the process state at time $t\text{-}1$, S_{t-1}, but not on previous ones, i.e. Pr $(S_t|S_{t-1}, S_{t-2}, \ldots, S_1)$ = Pr $(S_t|S_{t-1})$. The probability of a sequence of observations $O_1, O_2, \ldots O_T$ is then given by

$$Pr(O_1, O_2, \ldots, O_T) = Pr(O_1 \mid S_1) Pr(S_1) \prod_{t=2}^{T} Pr(O_t \mid S_t) Pr(S_t \mid S_{t-1})$$

i.e. it can be factored into a internal state transition component and an output component. This added flexibility permits to model a much larger range of temporal data than was possible with older models. Recent extensions to factorial HMM, tree-structured HMMs and Bayesian nets have further expanded the usefulness of HMMs for modeling spatio-temporal pattern, but the added complexity of these extensions often leads to problems in inference and learning (For a review, see Bunke & Caelli, 2001). We believe that the approach presented in the next section overcomes some of these problems.

4.2 Condition Rule Generation CRG_{ST}

We now turn to CRG_{ST}, a generalization of CRG from a purely spatial domain into a spatio-temporal domain (Bischof & Caelli, 2000, 2001). Here, data consist typically of time-indexed pattern descriptions, where pattern parts are described by unary features $\boldsymbol{u}$, spatial part relations by (spatial) binary features $\boldsymbol{b}_s$, and changes of pattern parts by (temporal) binary features $\boldsymbol{b}_t$. In the following sections, we discuss rule learning, rule generation models and applications of CRG_{ST}. In contrast to other temporal learners like hidden Markov models, the rules generated by CRG_{ST} are not limited to first-order time differences but can utilize more distant (lagged) temporal relations. At the same time, CRG_{ST} allows for the generation of non-stationary rules, unlike models like multivariate time series, which also accommodate correlations beyond first-order time differences but do not allow for the use of different rules at different time points.

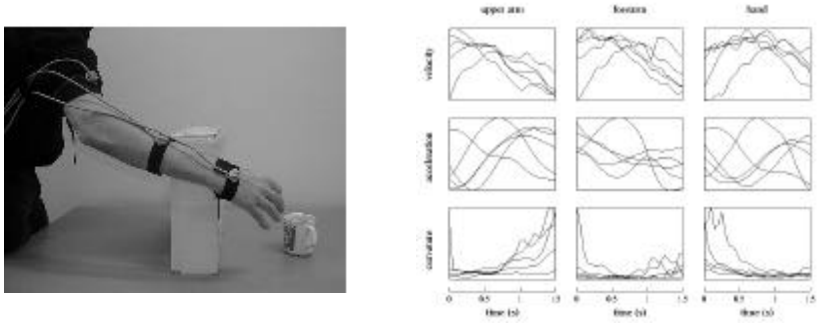

Figure 6: Left: Grasping movement around an obstacle. The movement sensors were placed on the upper arm, the forearm, and the hand. Right: Sample time-plots of the movement sequences. The left column shows traces for the upper arm, the middle column for the forearm and the right column for the hand. The first row shows time-plot for velocity (for a straight grasp movement), the second for acceleration (for a grasp movement over an obstacle), and the third for curvature (for a grasp movement around an obstacle). Each graph shows five samples for each action type. All measurements have been normalized for display purposes.

In the following, we illustrate each component of CRG_{ST} with an example where three different variations of grasp movements were learned: One where the hand moved in a straight path to the object, another where an obstacle in the direct path was avoided by moving over it, and a third where the obstacle was avoided by moving around it.

The movements were recorded using a Polhemus system running at 120Hz for three sensors, one on the upper arm, one on the forearm, and one on the hand (see Figure 6). From the position data (x(t),y(t),z(t)) of these sensors, 3-D velocity v(t), acceleration a(t), curvature k(t), and torsion ?(t) were extracted. Sample time-plots of these measurements are shown in Figure 6.

4.3 Representation of Spatio-Temporal Patterns

For the "grasp" example, the definition of the spatio-temporal patterns is straightforward. At every time point, the patterns consist of three parts, one for each sensor, each part being described by unary attributes 3D-position, velocity, acceleration, curvature and torsion, i.e. $\boldsymbol{u}(p_{i,t}) \; ? \; [x,y,z,v,a,k,?]$. Binary attributes were defined by simple differences, i.e. the spatial attributes were defined as $\boldsymbol{b}_s(p_{i,t},p_{j,t}) \; ? \; \boldsymbol{u}(p_{j,t}) \; ? \; \boldsymbol{u}(p_{i,t})$, and the temporal attributes were defined as $\boldsymbol{b}_t(p_{i,t},p_{j,t?1}) \; ? \; \boldsymbol{u}(p_{j,t?1}) \; ? \; \boldsymbol{u}(p_{i,t})$.

Our data model, and consequently our rules, are constrained to spatial and temporal adjacency (in the nearest neighbor sense) and temporal monotonicity, i.e. features are only connected in space and time if they are spatially or temporally adjacent, and the temporal indices for time must be monotonically increasing. Although this limits the expressive power of our representation, it is still more general than strict first-order discrete time dynamical models such as hidden Markov models.

For CRG_{ST}, finding an "interpretation" involves determining sets of linked lists of attributed and labeled features, that are causally indexed (i.e. the temporal indices must be monotonic), that maximally index a given pattern.

4.4 Rule Learning

CRG_{ST} generates classification rules for spatio-temporal patterns involving a small number of pattern parts subject to the following constraints: First, the pattern fragments involve only pattern parts that are

adjacent in space and time. Second, the pattern fragments involve only non-cyclic chains of parts. Third, temporal links are followed in the forward direction.

Rule learning proceeds in the following way: First, the unary features of all parts (of all patterns at all time points), $\boldsymbol{u}(p_{i,t})$, $i=1,...,n$; $j=1,...,T$, are collected into a unary feature space U in which each point represents the feature vector of one part at one time point. From this point onward, cluster tree generation proceeds exactly as described in Section 3.2, except that expansion into a binary space can now follow either spatial binary relations $\boldsymbol{b}_s$ or temporal binary relations $\boldsymbol{b}_t$. Furthermore, temporal binary relations $\boldsymbol{b}_t$ can be followed only in strictly forward direction, analyzing recursively temporal changes of either the same part, $\boldsymbol{b}_t(p_{i,t}, p_{i,t+1})$ (solid arrows in Figure 5), or of different pattern parts, $\boldsymbol{b}_t(p_{i,t}, p_{j,t+1})$ (dashed arrows in Figure 5) at subsequent time-points t and $t+1$. The decision about whether to follow spatial or temporal relations is simply determined by entropy-based criteria.

For the "grasp" movement, an example of a classification rule generated by CRG_{ST} is the following rule, which happens to be of the form $U - B_t - U - B_t - U$, with v = velocity; a = acceleration; Δx = displacement (over time) in x; Δy = displacement (over time) in y:

if $\boldsymbol{u}(p_{i,t})$	with $1.34 = v = 7.9$ and $-2.93 = a = 1.54$
and $\boldsymbol{b}_t(p_{i,t}, p_{j,t+1})$	with $-0.16 = \Delta x = 0.07$ and $-6.51 = \Delta y = 5.37$
and $\boldsymbol{u}(p_{i,t+1})$	with any value
and $\boldsymbol{b}_t(p_{i,+1}, p_{j,t+2})$	with $-5.39 = \Delta x = 0.08$ and $-6.51 = \Delta y = 5.37$
and $\boldsymbol{u}(p_{i,t+2})$	with $4.74 = v = 5.04$ and $-0.78 = a = -0.06$
then	pattern fragment $p_{i,t} - p_{j,t+1} - p_{k,t+2}$ is part of a grasping action moving over an obstacle.

In plain language, this classification rule reads as follows: If there is any sensor at any time point t with instantaneous velocity in the range $[-1.34, 7.9]$ and acceleration in the range $[-2.93, 1.54]$, and the sensor position changes by Δx in the range $[-0.16, 0.07]$ and by Δy in the range $[-6.51, 5.37]$ to the next time step, etc. then this is part of a grasping action over an obstacle. Note that the rules do not refer to particular sensors, even though this would be possible, and indeed helpful in this particular example. In general, however, identification of particular pattern parts may not be possible. As discussed earlier, it is one of the fundamental assumptions of CRG_{ST} that the correspondence between pattern parts and model parts is not known a

priori, and this is what sets it apart from approaches such as recurrent neural networks or hidden Markov models.

4.5 Rule Application

A set of classification rules is applied to a spatio-temporal pattern in the following way. Starting from each pattern part (at any time point), all possible sequences (chains) of parts are generated subject to the constraints the only adjacent parts are involved and no loops are generated. (Note that the same spatio-temporal adjacency constraints and temporal monotonicity constraints were used for rule generation.) Each generated chain $S_i = \langle p_{i1}, p_{i2}, \ldots, p_{in} \rangle$, is then classified using the classification rules, and the evidence vectors of all rules instantiated by S_i are averaged to obtain the evidence vector $\boldsymbol{E}(S_i)$ of the chain S_i. Further, the set $\boldsymbol{S}_p$ of all chains that start at p is used to obtain an initial evidence vector for part p:

$$\boldsymbol{E}(p) = \frac{1}{|\boldsymbol{S}_p|} \sum_{S \in \boldsymbol{S}_p} \boldsymbol{E}(S)$$

where $|\boldsymbol{S}_p|$ denotes the cardinality of the set $\boldsymbol{S}_p$. Evidence combination based on this equation is adequate if it is known that a single pattern is to be recognized. In the "grasp" example, this would be the case if it is known that a single movement is being presented.

However, CRG_{ST} is designed to work in more general situations where different pattern types overlap in space and time. This is the case, for example, when two or more different movement patterns are presented at the same time, or when the movement type changes over time. In this case, the simple scheme based on the equation above can produce incorrect results because some chains of parts may not be contained completely within a single pattern but "cross" different pattern types. Such "crossing" chains are likely to be classified in a arbitrary way, and to the extent that they can be detected and eliminated, part classification can be improved.

We use general heuristics for detecting rule instantiations involving parts belonging to different patterns. One such heuristic is based on the following idea. If a chain $S_i = \langle p_{i1}, p_{i2}, \ldots, p_{in} \rangle$ does not cross boundaries of objects then the evidence vectors $\boldsymbol{E}(S_{i1}), \ldots, \boldsymbol{E}(S_{in})$ are likely to be similar, and dissimilarity of the evidence vectors suggests that S_i may be a "crossing" chain. This similarity can be captured in the following way (McCane 1997): For a chain $S_i = \langle p_{i1}, p_{i2}, \ldots, p_{in} \rangle$, we compute the compatibility vector

$$\boldsymbol{w}(S_i) \leftarrow n^{-1} \sum_{k=1}^{n} \boldsymbol{E}(p_{ik})$$

This compatibility measure can be used directly in an iterative relaxation scheme (Rosenfeld & Kak 1982) for updating the part evidence vectors:

$$\mathbf{E}^{(t+1)}(p) \leftarrow \sigma(Z^{-1} \sum_{s \in S_p} \mathbf{w}^{(t)}(S) \otimes \mathbf{E}(S))$$

where σ is the logistic function $\sigma(z)=(1+\exp(-20(z-0.5)))^{-1}$, Z a normalizing factor, and the binary operator $\otimes$ is defined as a componentwise vector multiplication $[a \quad b]^T \otimes [c \quad d]^T \leftarrow [ac \quad bd]^T$. Convergence of the above relaxation scheme is typically obtained in about 10-20 iterations, and the updated part evidence vectors then reflect the partitioning of the test pattern into distinct subparts.

In summary, application of CRG_{ST} rules to a spatio-temporal pattern P_i (as shown in Figure 5) proceeds as follows:

- For all pattern parts p_i, extract all part chains $p_i - p_j - p_k$ - ... starting at p_i, up to a specified maximal length.
- Classify all part-chains S_i using the CRG_{ST} rules, each producing an evidence vector $\boldsymbol{E}(S)$.
- Compute initial compatibility vectors $\boldsymbol{w}(S_i)$.
- Run the iterative relaxation scheme until convergence. This produces a final evidence vector $\boldsymbol{E}(S_i)$ for each part.

4.6 Performance of CRG_{ST}

Performance of CRG_{ST} was tested with the "grasp" example in a leave-one-out paradigm, i.e. in each run, movement classes were learned using all but one patterns, and the resulting rule system was used to classify the remaining pattern. Results of these tests are shown in Table 3 for different attribute combinations for unary, spatial binary and temporal binary relations. The last column indicates what percentage of pattern parts was classified correctly on average. Although each test pattern consisted of a single movement, this was not assumed by the classification algorithm. Using the "single-movement" assumption, e.g. in a winner-take-all scheme, would lead to somewhat higher classification percentages.

μ	β_s	β_t	correct
xyx	xyz	xyz	95.4 %
-	xyz	xyz	96.3 %
-	-	xyz	43.1 %
va	va	va	52.2 %
-	va	va	46.6 %
-	-	va	28.3 %
k?	k?	k?	34.6 %
-	k?	k?	40.7 %
-	-	k?	28.9 %
xyzva	xyzva	xyzva	90.8 %
-	xyzva	xyzva	96.5 %
-	-	xyzva	33.1 %

Table 3: Performance of CRG_{ST} for learning three different types of grasping actions. The first three columns indicate what attributes were used for unary, spatial binary and temporal binary relations, and the last column indicates the percentage of test pattern points that was classified correctly. Dashes indicate that no feature was used. xyz: position in 3D; v: velocity: a: acceleration; k: curvature; ?: torsion.

The results show that classification performance varies, not unexpectedly, with the choice of attribute sets. For the simple movement patterns used here, position information, possibly enhanced by velocity and acceleration information, was clearly sufficient for encoding and learning the movement patterns. Curvature and torsion information alone was insufficient, which is not surprising given that the movements were fairly linear.

The results show that CRG_{ST} is a promising technique for the learning of motion patterns. Obviously, the movement patterns used here were very simple, but work is currently in progress on the encoding and learning of much more complex movement sequences.

4.7 Incorporating Model Constraints into Rule Generation

The definition of spatio-temporal patterns introduced in this section is very general and applies to situations where no domain knowledge is available. Learning of patterns may be made more efficient through introduction of relational constraints based on domain knowledge. For example, for the recognition of human body movements, the spatial relation between hand and elbow may be much more diagnostic than

the relation between hand and knee, or, more generally, intra-limb spatial relations are more diagnostic than inter-limb spatial relations. For these reasons, arbitrary model-based constraints can be introduced into the underlying relational structure, thus covering the range from fully-connected non-directed relational models to specific directed relational models (Bischof & Caelli, 2001). Obviously, in situations where no domain knowledge is available, the most general model should be used, and learning is consequently slower and sub-optimal. Conversely, when sufficient domain knowledge is available, strong constraints can be imposed on the relational model, and learning is consequently more efficient.

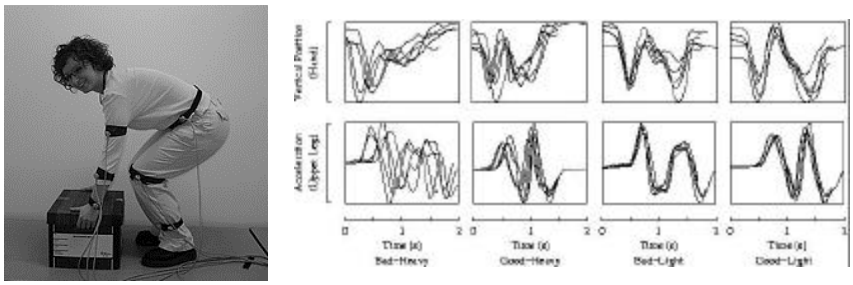

Figure 7: Left: Lifting a heavy object. The movement sensors were placed on the hip, above the knee, above the foot, on the upper arm, on the forearm, and on the hand of the left body side. Right: Sample time-plots of the movement sequences. The first row shows time-plots for the vertical position of the sensor placed on the hand, the second row the acceleration of the sensor placed above the knee. The four columns show traces the four movement classes.

The model-based CRG_{ST} approach is illustrated in an example where the classification of four different variations of lifting movements were learned, two where a heavy object was lifted, and two where a light object was lifted. Both objects were either lifted with a knees bent and a straight back ("good lifting"), or with knees straight and the back bent ("bad lifting"). Thus there were four movement classes, 1) good lifting of heavy object, 2) good lifting of light object, 3) bad lifting of a heavy object, and 4) bad lifting of a light object. The movements are quite difficult to discriminate, even for human observers. This was done in order to test the limits of the movement learning system.

The movements were recorded using a Polhemus system running at 120Hz for six sensors, located on the hip, above the knee, above the foot, on the upper arm, on the forearm, and on the hand of the left body side (see Figure 7). From the position data *(x(t),y(t),z(t))* of these sensors, 3-D velocity *v(t)* and acceleration *a(t)* were extracted, both with respect to arc length $ds(t)=(dx^2(t)+dy^2(t)+dz^2(t))^{1/2}$, i.e. $v(t) = ds(t)/dt$ and $a(t) = d^2s(t)/dt^2$.

The spatio-temporal patterns were defined in the following way: At every time point, the patterns consisted of six parts, one for each sensor, each part being described by unary attributes $\mathbf{u} \in [x,y,z,v,a]$. Binary attributes were defined by simple differences, i.e. the spatial attributes were defined as $\mathbf{b}_s(p_{i,t},p_{j,t}) = \mathbf{u}(p_{j,t}) - \mathbf{u}(p_{i,t})$, and the temporal attributes were defined as $\mathbf{b}_t(p_{i,t},p_{j,t-1}) = \mathbf{u}(p_{j,t-1}) - \mathbf{u}(p_{i,t})$.

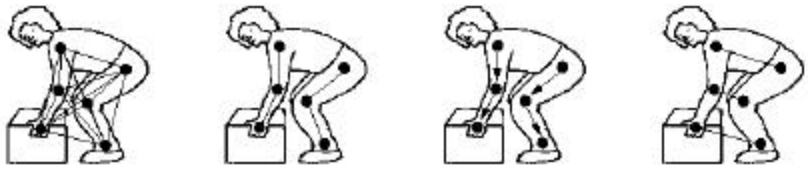

Figure 8: Sketch of the four pattern models used for the recognition of lifting movements. From left to right, the sketches show the fully connected relational model, the non-directional intra-limb model, the directional intra-limb model and an inter-limb model. See text for further explanations.

Performance of CRG_{ST} was tested with a leave-one-out paradigm, i.e. in each test run, movement classes were learned using all but one sample, and the resulting rule system was used to classify the remaining pattern, as described in Section 4.3. The system was tested with three attribute combinations and four pattern models. The three attributes combinations were 1) $\mathbf{u}=[x,y,z]$, 2) $\mathbf{u}=[v,a]$, and 3) $\mathbf{u}=[x,y,z,v,a]$. The four pattern models were 1) a fully connected relational model (i.e. binary relations were defined between all six sensors), 2) a non-directional intra-limb model, i.e. binary relations were defined between hip - knee, knee - foot, upper arm - forearm, and forearm - hand, 3) a directional intra-limb model (i.e. binary relations were defined as in 2) but only in one direction), and finally 4) an inter-

limb model (i.e. binary relations were defined between hip - upper arm, knee - forearm, and foot - hand) (see Figure 8).

Results of these tests are shown in Table 4, for the attribute subsets and the pattern models just described. The results show that performance is fairly high, in spite of the fact that the movement patterns are not easy to discriminate for human observers. Best performance is reached for the intra-limb directional model (see Figure 8) and the full feature combination *xyzva*. Even though performance for feature combination *va* is very low, the two features improve, performance for the *xyz* feature combination.

Model	xyz	va	xyzva
fully connected	85 %	30 %	75 %
intra-limb non-directional	75 %	32 %	75 %
intra-limb directional	85 %	20 %	90 %
inter-limb non-directional	60 %	5 %	70 %

Table 4: Performance of CRG_{ST} for learning four different types of lifting actions. The first column indicates what relational model was used, and the three remaining columns give average performance for three different attributes combinations (xyz = position in 3D; v = velocity: a = acceleration). Each cell shows percentage correct identifications under the assumption that a single movement pattern is present.

5 Conclusions

Visual learning is concerned with systems that learn to recognize complex spatial and spatio-temporal patterns. We have presented a number of such learning approaches, from simple discriminant functions over decision trees nad evidence-based systems to advanced learning systems like HMMs and CRG_{ST}.

Although these systems are fairly effective in learning moderately sized data sets as they arise in vision, their efficiency, their flexibility and their power is still a far cry from what is required to match human performance in visual learning.

6 References

Bischof, Walter F & Caelli, Terry (1994). Learning structural descriptions of patterns: A new technique for conditional clustering and rule generation. Pattern Recognition, 27, 689-697

Bischof, Walter F & Caelli, Terry (1997a). Visual learning of Patterns and objects. *IEEE Transactions on Man, Machine, and Cybernetics, 27*, 907-917.

Bischof, Walter F & Caelli, Terry (1997b). Scene understanding by rule evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 19*, 1284-1288.

Bischof, Walter F & Caelli, Terry (2000). Learning complex action patterns with CRGst. In S. Singh, N. Murshed and W. Kropatsch (Eds.) Advances in Pattern Recognition - IACPR 2001, pp. 280-289.

Bischof, Walter F & Caelli, Terry (2001). Learning spatio-temporal relational structures. Journal of Applied Intelligence, 15, 707-722.

Briscoe, Gary & Caelli, Terry (1996). *A Compendium of Machine Learning*. Norwood, NJ: Ablex.

Bunke, Horst & Caelli, Terry (Eds.) (2001). *Hidden Markov Models: Applications in Computer Vision*. Singapore: World Scientific.

Caelli, Terry & Bischof, Walter F (Eds.) (1997). *Machine Learning and Image Interpretation*. New York: Plenum.

Caelli, Terry & Dreier, Ashley (1994). Variations on the evidence-based object recognition theme. *Pattern Recognition, 27,* 185-204.

Devijver, P. A. & Kittler, J. (1980). *Pattern Recognition: A Statistical Approach*. Englewood Cliffs, NJ: Prentice-Hall.

Duda, Richard, Hart, Peter & Stork, David (2001). *Pattern Classification*. New York: Wiley.

Flynn, Patrick & Jain, Anil (1993). *Handbook of Pattern Recognition and Image Processing.* New York: Academic.

Hertz, John, Krogh, Anders & Palmer, Richard (1991). *Introduction to the Theory of Neural Computation.* Redwood City, CA: Addison-Wesley.

Jain, Anil & Dubes, Richard (1988). *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice-Hall.

Mitchell, Tom (1997). *Machine Learning*. New York: McGraw-Hill.

Pao, Y. (1989). *Adaptive Pattern Recognition and Neural Networks*. Reading, MA: Addison-Wesley.

Quinlan, Ross (1992*). C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Rabiner, Lawrence & Juang Biing-Huang (1993). Fundamentals of Speech Recognition, Chapter 6. Englewood Cliffs, NJ: Prentice-Hall.

Rosenfeld, Azriel & Kak, Avi (1982). Digital Picture Processing, 2nd Edition. New York: Academic.

Shapiro, Linda & Haralick, Robert (1981). Structural descriptions and inexact matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *3*, 501-519.