ECAI BERLIN 20.8.-25.8. 2000

ECAI-2000 Workshop Programme

Workshop W27

Machine Learning in Computer Vision

Berlin, August 22nd, 2000

Workshop Notes

Edited by Floriana Esposito and Donato Malerba

Dipartimento di Informatica, Università degli Studi di Bari
via Orabona 4, I-70125 Bari, Italy
`{esposito | malerba}@di.uniba.it`

Organisation Commitee:

Home page: http://www.di.uniba.it/~malerba/ws-ecai2000/

Acknowlegements:

MLNET II: European Network of Excellence on Machine Learning
ECAI 2000: Programme Committee and Workshop Chair

Preface

An agent is anything that perceives the surrounding environment through its sensors and performs actions upon it through its effectors. AI research aims to describe and build rational agents, which try to optimise their performance, given the information perceived from the environment and their background knowledge. Computer vision and machine learning investigate two important capabilities of rational agents: Human-like perception of the sensed data and improvement of agent performance with time. In recent years, there has been an increased interest in the synergetic contribution of these two fields to the development of agents that can solve "real-world" problems. This workshop is multidisciplinary in that it provides a forum for discussing current research in AI and pattern recognition that pertains to machine learning in computer vision systems.

From the standpoint of computer vision systems, machine learning can offer effective methods for automating the acquisition of visual models, adapting task parameters and representation, transforming signals to symbols, building trainable image processing systems, focusing attention on target object. To develop successful applications, however, we need to address the following issues:

- How is machine learning used in current computer vision systems?
- What are the models of a computer vision system that might be learned rather than hand-crafted by the designer?
- What machine learning paradigms and strategies are appropriate to the computer vision domain?
- How do we represent visual information?
- How does machine learning help to transfer the experience gained in creating a vision system in one application domain to a vision system for another domain?

From the standpoint of machine learning systems, computer vision can present interesting and challenging problems. Many studies in machine learning assume that a careful trainer provides internal representations of the observed environment, thus paying little attention to the problems of perception. Unfortunately, this assumption leads to the development of brittle systems with noisy, excessively detailed or quite coarse descriptions of the perceived environment. Some specific machine learning research issues raised in the computer vision domain are:

- How can noisy observations be dealt with?
- How can large sets of images with no annotation be used for learning?
- How can mutual dependency of visual concepts be dealt with?
- What are the criteria for evaluating the quality of learning processes in computer vision systems?
- When should a computer vision system start/stop the learning process and/or revise acquired models?
- When is it useful to adopt several representations of the perceived environment with different levels of abstraction?

This workshop provides some answers to some of these questions, and maintains a balance between theoretical issues and descriptions of implemented systems to promote synergy between theory and practice. It follows the tradition of similar events organized in the past decade, such as: NSF/ARPA Workshop on "Machine Vision and Learning", Harpers Ferry, West Virginia (October 1992); AAAI Symposium on “Machine Learning in Computer Vision: What, Why, and How?”, Raleigh, North Carolina (October 1993); International Workshop on “Learning in Computer Vision”, Sydney, New South Wales, Australia (April, 1996); ECCV (European Conference on Computer Vision) Workshop on “Learning in Computer Vision”, Freiburg, Germany (June 1998), ICML (International Conference on Machine Learning) Workshop on “Machine Learning in Computer Vision”, Bled, Slovenia (June 1999); First International Workshop on “Machine Learning and Data Mining in Pattern Recognition”, Leipzig, Germany (September 1999).

We wish to thank the members of the Program Committee for their assistance in setting up this workshop and in reviewing submitted papers. We wish also to thank the ECAI 2000 Programme Committee and in particular the Workshop Chair, Prof. Marie-Odile Cordier, for supporting the organisation of this workshop and the European Network of Excellence in Machine Learning (MLNET) for the economical support. Finally, we wish to thank authors and invited speakers for their excellent contributions in promoting discussion and the development of new ideas and methods on the workshop topics.

Floriana Esposito and Donato Malerba
August, 2000

Table of Contents

Schedule

8:30	–	9:10	Registration
9:10	–	9:20	*F. Esposito and D. Malerba* Opening remarks
9:20	–	10:10	Invited talk: *E. Hancock* A factorisation framework for structural pattern matching
10:10	–	10:40	*W. Bischof and T. Caelli* Learning actions: induction over spatio-temporal relational structures
10:40	–	11.00	Coffee break
11:00	–	11:30	*F. Esposito, D. Malerba and F. Lisi* Understanding multi-page printed documents: a multiple concepts learning problem
11:30	–	12:00	*P. Perner and C. Apte* Improving the accuracy of C4.5 by feature pre-selection
12:00	–	12:30	*E. Ardizzone, A. Chella and R. Pirrone* Feature-based shape recognition by Support Vector Machines
12:30	–	13:50	Lunch break
13:50	–	14:40	Invited talk: *L. Saitta* An abstraction model of visual perception
14:40	–	15:10	*R. Cucchiara, P. Mello, M. Piccardi and F. Riguzzi* An application of machine learning and statistics to defect detection
15:10	–	15:30	Coffee break
15:30	–	16:00	*A. Chella, D. Guarino, I. Infantino and R. Pirrone* A high-level vision system for the symbolic interpretation of dynamic scenes by the ARSOM neural networks
16:00	–	17:00	Discussion and closing remarks

A Factorisation Framework for Structural Pattern Matching

Edwin Hancock

Department of Computer Science,
University of York,
York, Y01 5DD, UK
erh@cs.york.ac.uk

Abstract

Relational representations are of critical importance in high-level vision. They can be used to represent the arrangement of image primitives in a manner which captures the structure of both objects and scenes. Moreover, they can convey important semantic information which is not captured by using object attributes alone. In this talk we describe some important steps in the direction of learning relational descriptions for high-level vision.

The talk commences by showing how the EM algorithm can be used to compute a measure of relational similarity between pairs of graphs. Next, we show how the statistical measure of relational similarity, which results from this analysis, can be cast into a matrix setting. This opens the possibility for performing a number of important operations on relational graphs using matrix factorisation methods, such as singular value decomposition and eigendecomposition. First, we show how to find correspondence matches between graphs of different size, i.e. subgraph isomorphisms, using singular value decomposition. Next, we suggest how to use the new matrix representation to learn structural representations. Finally, we show how the framework can be used to edit the structure of graphs so as to remove relational errors using an eigenclustering method.

We demonstrate the new framework on a number of problems from high-level vision, including model alignment, content-based image retrieval and perceptual grouping. This work can be viewed as combining ideas from statistical and structural pattern recognition, and from spectral graph theory.

An Abstraction Model of Visual Perception

Lorenza Saitta

Università del Piemonte Orientale
Dipartimento di Scienze e Tecnologie Avanzate
Corso Borsalino 54, 15100 Alessandria
saitta@unipmn.it, saitta@di.unito.it

Abstract

The talk presents a computational model of abstraction, based on perceptual principles, well suited to describe and handle visual and spatial concepts and data. The model spans two dimensions: one concerns the type of representation and processing applied to visual input, starting from the signal and ending with an abstract theory; the onter one concerns the selection of a suitable level of detail for representing the spatial concepts when a specific goal has to be reached. In order to concretely implement the transformation between different levels, a number of abstraction operators are defined. The semantics of the operators, as well as conditions for applicability, will be analyzed.

The model will be shown to encompass and to give a precise basis to previous attempts, in vision, to describe changes of representation, as well as to visual perceptual findings from cognitive sciences. A practical application to cartographic generalization in GIS will also be described.

Feature-Based Shape Recognition by Support Vector Machines

Edoardo Ardizzone[1], **Antonio Chella**[2] and **Roberto Pirrone**[3]

Abstract. A model identification technique for the objects in a gray-level image is proposed, based on the extraction of a compact shape feature in terms of the statistical variance pattern of the objects' surface normals.

A shape recognition system has been developed, that detects automatically image ROIs containing single objects, and classifies them as belonging to a particular class of shapes.

We use the eigenvalues of the covariance matrix computed from the pixel rows of a single ROI. These quantities are arranged in a vector form, and are classified using Support Vector Machines (SVMs). The selected feature allows us to recognize shapes in a robust fashion, despite rotations or scaling, and, to some extent, independently from the light conditions.

Theoretical foundations of the approach are presented, together with an outline of the system, and preliminary experimental results.

1 INTRODUCTION

Object recognition is one of the main goals of a computer vision system. Particular attention has been paid, during the last years, to the recognition of 3D objects in real scenes, and the several applications have been prposed in autonomous robotics.

Identification and location are the two basic steps in whatever model-based object recognition approach [2, 4, 14]. We present an identification technique developed as a part of a 3D vision system aimed to the recognition of objects in semi-structured environments. Currently the identification system is being tested as a scene description module within a content-based image retrieval application.

The proposed architecture is arranged as follows. The image is automatically scanned to locate ROIs containing single objects. The objects' shape is described in terms of the eigenvalues of the covariance matrix computed from the pixel rows of the ROI: the eigenvalues are arranged as a vector. We use a pool of suitably designed Support Vector Machines [11, 15] to classify different shape classes such as cubes, spheres, cones, pyramids, cylinders. The system provides a simple description of the shapes present inside the image, together with their 2D displacement.

The use of this technique is based on several considerations. First, a theoretical analysis proves that, under the assumption of Lambertian surface, the eigenvalues vector is directly related to the change of surface normals of objects under investigation. Experimental evidence shows that the selected feature performs as "almost invariant" with respect to illumination conditions and changes in the object's attitude. Consequently, similar shapes form clusters in the feature space, and learning techniques are needed to perform classifications.

The eigenvalues vector is a compact and efficient way to describe the statistical variance pattern of the shape profile, due to the complete de-correlation performed on the input patterns by the KL transform [5] which is strictly related to the Principal Component Analysis [6]. We perform a KL analysis on the grey-level images of the objects as in the classical appearance based approaches [8]. Our eigenvalues vector acts as a characterization of the whole object shape, and allows us to avoid storing large amounts of image data to describe several views of a particular model.

The last consideration is that the implicit application of KLT to the ROI pixel rows provides zero mean transformed vectors. In this way, comparisons between the eigenvalue vectors of two different ROIs are meaningful because they're insensible to bias effects in the pixel values.

We performed our experiments on real scenes made by colored wooden blocks of various shapes, placed on a table and viewed by a CCD camera. Lighting conditions were the normal diffuse ones of our laboratory. Even some artificial images produced by a solid modeler, with a single directional light source were used to make the training set for SVMs as general as possible. Despite the particular experimental setup, the approach we developed is a general one, and we have used it even in the analysis of real generic scenes.

The paper is arranged as follows. In section 2 theoretical issues on the eigenvalues vector will be addressed. Section 3 will explain in detail the entire system performance, while the experimental results will be reported in section 4. Finally, conclusions will be drawn in section 5.

2 THEORETICAL REMARKS

The use of KLT features for pattern recognition is a useful technique in the computer vision community [9, 13] but, in general, it is applied to the image as a whole, and the transformed vectors are used for the classification task.

In our approach, KLT is applied to the scan-lines of a generic sub-image, and only the eigenvalues of their covariance matrix are taken into account. Formally, given a $N \times M$ rectangular region of an image, we compute the matrix:

$$\mathbf{C_r} = E\{(\mathbf{r_k} - \mathbf{r_m})(\mathbf{r_k} - \mathbf{r_m})^T\} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{r_k}\mathbf{r_k}^T - \mathbf{r_m}\mathbf{r_m}^T,$$

[1] DIAI - University of Palermo and CERE - National Research Council Viale delle Scienze 90128, Palermo, Italy, email: ardizzon@unipa.it
[2] DIAI - University of Palermo and CERE - National Research Council Viale delle Scienze 90128, Palermo, Italy, email: chella@unipa.it
[3] DIAI - University of Palermo and CERE - National Research Council Viale delle Scienze 90128, Palermo, Italy, email: pirrone@unipa.it

where

$$\mathbf{r_m} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{r_k}.$$

In the previous equation $\mathbf{r_k}$ is the generic pixel row vector of the ROI under investigation, considered in column form. Then, we compute the vector:

$$\lambda = diag(\mathbf{C_q}) \tag{1}$$

Here $\mathbf{C_q} = \mathbf{A}\mathbf{C_r}\mathbf{A}^T$, is the covariance matrix of the KL transformed vectors $\mathbf{q_k}$, while $\mathbf{A}$ is the transformation matrix whose rows are the eigenvectors of $\mathbf{C_r}$.

The matrix $\mathbf{C_q}$ is diagonal, so KLT performs total decorrelation of the input vectors. Moreover, the mean value of the transformed vectors is always zero. These properties will be used to reinforce our conclusions about the choice of λ as a global shape descriptor.

The first step towards the justification of the usability of λ, is the proof of the relation between λ and the actual shape of the object depicted in the selected ROI. We'll consider a weak perspective camera model and the Lambert law to model the light reflection process upon the object surface. These constraints are not so restrictive, and are widely used to model perceptive processes. In particular, weak perspective is introduced for simplicity, while the Lambertian surface constraint holds for most real objects.

If an object is imaged by a camera under the weak perspective assumption, each point $\mathbf{p}_o = (x, y, z)$ of the object, expressed in the camera coordinate system, is mapped onto an image point $\mathbf{p} = (u, v)$ where $\mathbf{p} = \mathbf{W_P}\mathbf{p}_o$ is the perspective transformation. According to the Lambert law the image irradiance E in each point is equal to the image intensity value in the same point, and is expressed by:

$$I(i,j) = E(\mathbf{p}) = H\rho\mathbf{l}^T\mathbf{n}(\mathbf{p}_o)$$

In the previous equation H is a constant value related to the lens model, ρ is the albedo of the object surface, $\mathbf{l}$ is the illuminant (constant) vector and $\mathbf{n}(\mathbf{p}_o)$ is the surface normal at the point $\mathbf{p}_o$. The first equality takes into account the coordinate change from the image center to the upper-left corner, which is a linear transformation.

When we consider a vertical slice of the image, then each pixel row vector is defined as:

$$\mathbf{r}_k = \{I(k,j) : j = 0, \ldots, M-1\}^T, k = 0, \ldots, N-1 \tag{2}$$

Here the transpose symbol is used to define $\mathbf{r}_k$ as a column vector. Substituting the expression of the generic pixel value $I(i,j)$ in equation 2 we obtain:

$$\mathbf{r}_k = H\rho\{\mathbf{l}^T\mathbf{n}_{kj} : j = 0, \ldots, M-1\}^T, k = 0, \ldots, N-1 \tag{3}$$

In equation 3 $\mathbf{n}_{kj}$ refers to the surface normal vector that is projected onto position (k,j) in the image plane.

Now, we derive the expression for the generic element $\mathbf{C}_r(i,j)$ of the pixel rows covariance matrix, using the equation stated above:

$$\mathbf{C}_r(i,j) = \frac{1}{N}\sum_k \mathbf{r}_{ki}\mathbf{r}_{kj} - \frac{1}{N^2}\sum_k \mathbf{r}_{ki}\sum_k \mathbf{r}_{kj} \tag{4}$$

Substituting equation 3 in equation 4 we obtain:

$$\begin{aligned}\mathbf{C}_r(i,j) = \; & \tfrac{H\rho}{N}\textstyle\sum_k(\mathbf{l}^T\mathbf{n}_{ki})(\mathbf{l}^T\mathbf{n}_{kj}) - \\ & \tfrac{H\rho}{N^2}\left(\textstyle\sum_k \mathbf{l}^T\mathbf{n}_{ki}\right)\left(\textstyle\sum_k \mathbf{l}^T\mathbf{n}_{kj}\right)\end{aligned} \tag{5}$$

We rewrite equation 5:

$$\begin{aligned}\mathbf{C}_r(i,j) = \; & H\rho\mathbf{l}^T\left[\tfrac{1}{N}\textstyle\sum_k \mathbf{n}_{ki}\mathbf{n}_{kj}^T - \right. \\ & \left.\tfrac{1}{N^2}\left(\textstyle\sum_k \mathbf{n}_{ki}\right)\left(\textstyle\sum_k \mathbf{n}_{kj}\right)^T\right]\mathbf{l}\end{aligned} \tag{6}$$

Finally, equation 6 is rewritten in two different forms for diagonal and off-diagonal terms:

$$\mathbf{C}_r(i,j) = \begin{cases} H\rho\mathbf{l}^T\mathbf{C}_n^{(i)}\mathbf{l} & , i = j \\ H\rho\mathbf{l}^T\left(\mathbf{K}_n^{(ij)} - \mathbf{n}_m^{(i)}\mathbf{n}_m^{(j)T}\right)\mathbf{l} & , i \neq j \end{cases} \tag{7}$$

The last equation states that diagonal terms of the pixel rows covariance matrix can be computed directly from the covariance matrices $\mathbf{C}_n^{(i)}$ of the object surface normals projecting themselves onto a single slice column. The off-diagonal terms of the same matrix are computed from the difference between the correlation matrix $\mathbf{K}_n^{(ij)}$ of the normals related to two different columns minus the term obtained from the product of their mean vectors.

From the previous result we argue that the matrix $\mathbf{C}_r$ is well suited to express the statistical variance pattern of the object surface shape along both rows (off-diagonal terms) and columns (diagonal terms) despite it is not referred to the entire slice, but it's computed starting from its rows. We achieve a considerable reduction of computational time, without losing the expressiveness of the selected feature, because we compute only M eigenvalues, while the application of the KLT to the entire region involves the computation of $N \times M$ coefficients.

The use of the eigenvalues allows us to transform our feature in a compact way. The λ vector still expresses the rows variance pattern because it results from the covariance matrix (equation 1) of the KL transformed pixel rows that are completely uncorrelated.

Moreover, the λ vector allows performing comparisons between different regions in the same image or from different ones in order to search for similar shapes. In general, two different sets of rows cannot be directly compared, due to the presence of bias effects in the pixel values deriving from noise and/or local lighting conditions. The implicit application of KLT deriving from the use of λ implies that if we compare two different regions we refer to their transformed rows which have zero mean value: these are correctly compared because they've the same mean value and no bias effect is present.

3 DESCRIPTION OF THE SYSTEM

We performed our experiments on real scenes made by colored wooden blocks of various shapes, placed on a table and viewed by a CCD camera. Lighting conditions were the normal diffuse ones of our laboratory. Even some artificial images produced by a solid modeler, with a single directional light source were used to make the training set for SVMs as general as possible. Despite the particular experimental setup, the approach we developed is a general one, and we have used it even in the analysis of real generic scenes (see figure 5).

We've analyzed the histogram of the components of λ computed from several images both synthetic and real, depicting single shapes under varying attitudes and lighting (see figure 1). This histogram performs as an "almost invariant" under varying illuminant conditions and attitude of the object.

The histogram exhibits some dominant modes, whose relative position and amplitude depend on the shape observed. The amplitude and position of these histogram modes remain almost unchanged under rotation, translation, and scaling of the object. The light direction acts as a scaling factor for all the terms of $\mathbf{C}_r$ (equation 7) thus affecting in a uniform manner all the components of λ. We have experimental evidence that in our setup varying $\mathbf{l}$ doesn't affect the histogram too much.

The almost invariant behavior of the λ vector implies that similar shapes tend to form clusters in the feature space. Shape models are de-

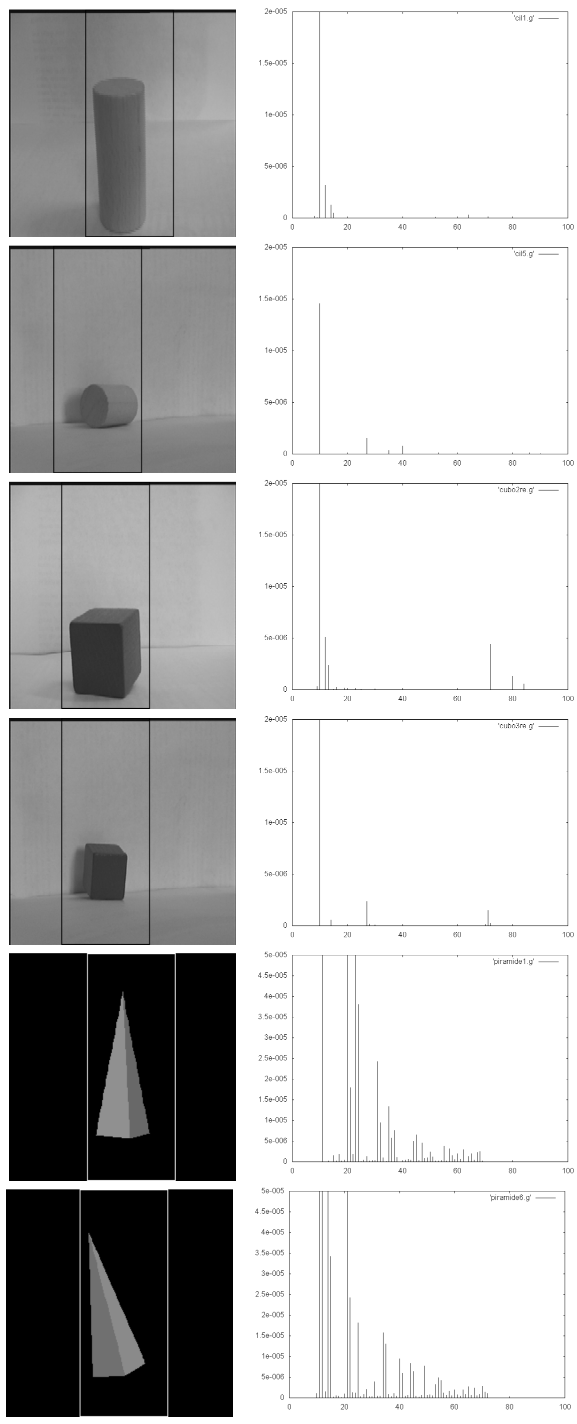


Figure 1. Shape examples together with the relative λ histogram. Selected ROIs are 256×100 wide. Comparing the couples along each row, it can be noted that changes in attitude and lighting don't affect the histogram too much.

fined in terms of these clusters, and an unknown object is recognized in terms of the closest cluster. According to these experimental evidence, a learning machine trained on the model clusters, is a suitable approach to perform recognition.

We've set up a classifier based on the use of a pool of suitably tuned SVMs that operate in the eigenvalues space.Moreover, a search algorithm for the automatic analysis of the test images has been derived, which is based on the maximization of correlation between the actual λ vector and some sample vectors from the different shape classes.

The complete system acts in the following way: first, the image is scanned from left to right and from top to bottom by moving windows of fixed size in order to locate some possible regions of interest. Then, the height of each window is resized to enclose at most a single complete object. Finally, all the selected regions are classified by the SVMs pool.

3.1 Automatic search algorithm

The search algorithm we implemented is based on a two-pass strategy. The first step performs a rough location of the ROIs for the horizontal and vertical displacement. The second step defines the windows' dimensions for all the selected positions.

The search criterion is the correlation maximization between the λ vector of a fixed size slice and a sample of each shape class computed as the mean vector between those used as training set for the various SVMs. The correlation values with all the class samples are computed by scanning the image from left to right with a 256×100 fixed slice, and the maximum is considered time by time. This information is used only to detect if there's something without looking at a particular shape. Positive peaks of the correlation defined above vs. the position of the left side of the slice, indicate a region of interest.

Relevant positions are selected as follows. The cumulative function of the correlation is computed, and the selected points are the zero crossings of its second order derivative: these are the slope inversion points of the cumulative function, that in turn correspond approximately to the correlation maxima (see figure 2). We found convenient to use the cumulative function in order to avoid noisy spikes that can be present near a peak when detecting maxima directly from the correlation plot.

For each selected ROI, single objects are detected using a 20×100 fixed slice that moves from top to bottom. Again the correlation maxima are computed with the previous strategy.

In the second step of the algorithm, we use the variance maximization as guiding criterion to resize windows' height in order to enclose a single complete object. Here the variance is approximated as the maximum eigenvalue in the λ vector of the current slice. Starting from the position of each correlation peak, windows are enlarged along their height by a single row at a time, and the λ vector is computed, taking into account its maximum component. Positive peaks correspond approximately to the upper and lower edge of the object. Again we use the second order derivative of the cumulative function in order to avoid mismatches due to the presence of variance spikes near the actual maximum. Search results are depicted in figure 3.

3.2 Shape classification using SVM

We adopted SVMs as learning strategy. The analytical approach to the classification performed by SVMs ensures us that correct separating hyperplane between classes will be computed. In general, this approach has to cope with two problems: the more or less precise location of the separating hyperplane depends on the number of support

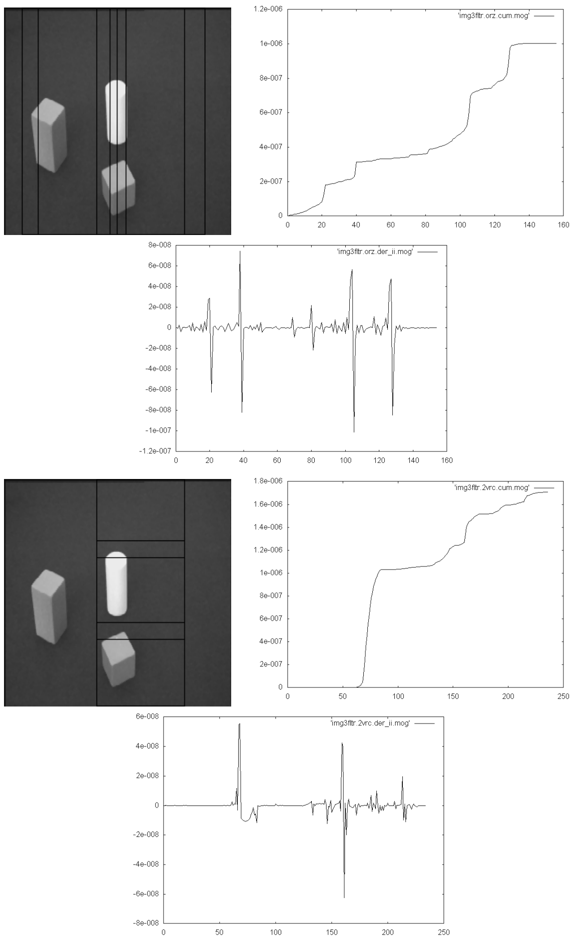


Figure 2. Example of correlation maximization search. In the topmost row there is a sample with the slices corresponding to the correlation maxima, the cumulative function plot, and its second order derivative. Maxima have been found in position 21, 39, 105 and 128. In the lower row there are the vertical sub-slices of the ROI in position 105 along with the cumulative function and its second order derivative.

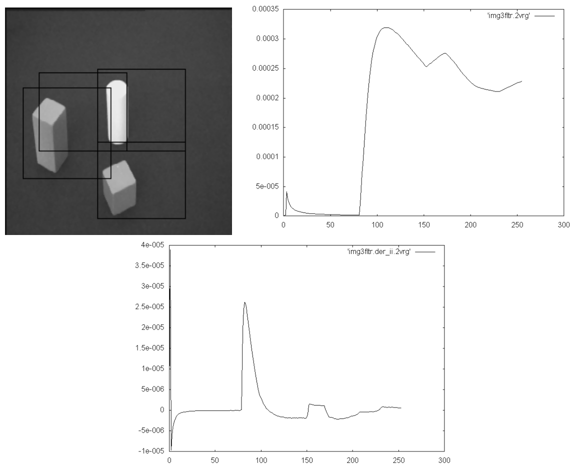


Figure 3. Example of variance maximization search. On the left, final slices of the picture in figure 2 are depicted along with the plot of variance, and its second order derivative for the slice in position 105.

vectors that is related to the number of training examples. SVMs are, in general, computationally expensive because they involve a function maximization. Nevertheless, SVMs represent a good development environment for tuning the classification strategy.

In our implementation, SVMs performed very well in terms of computational time: in average, less than 10 seconds on a Pentium II processor are sufficient in order to train a machine to discriminate between a singular shape class and all the others.

The SVM in its original formulation is designed for two-class discrimination, so we used a particular training strategy, in order to cope with our multi-class task. Two different kinds of SVMs have been trained on six shape classes: cube, cylinder, pyramid, cone, ellipsoid, and box. First, six SVMs have been trained in a *one-versus-others* fashion, each of them being able to discriminate between a particular class and all other objects. Besides, a second pool of 15 SVMs have been trained using a *pair-wise* strategy: each SVM is trained to discriminate between a single pair of the desired classes, so for K classes we need $K(K-1)/2$ different machines.

The use of two learning strategies is related to the need to avoid mismatches in classification. Many researchers [7, 10, 12] studied the problem of the optimum placing of borderlines between classes. A *one-versus-others* training leaves some uncertainty regions in the feature spaces where we're not able to decide correctly to which class belongs the actual sample. A way to provide a refinement of the boundary locations between multiple classes is the use of a *pair-wise* learning strategy.

In our experiments the use of *one-versus-others* or *pair-wise* strategy alone is not sufficient to obtain a correct classification. So, in the test phase, we use the first set of machines in order to provide a rough discrimination, which is then refined by the use of the second ones. The *one-versus-others* machines provide their own estimate in a *winner-takes-all* fashion: the distances between the object's λ vector and the optimal hyperplanes defining each shape class are computed, and the class with the highest positive distance is taken as the winner. In this way the class where the actual sample vector is more "inside" is selected.

In some cases this approach doesn't allow a very sharp classification, and the sample vector results inside two or more classes. In this case, the *pair-wise* machines are used in order to provide a disambiguation. The result of testing the vector with each machine is accumulated for each class in a sort of round-robin challenge. The class with the highest score wins the tournament, and the sample vector is classified according to this outcome.

4 EXPERIMENTAL SETUP

To set up the classifier, a training set has been used, which consists of 118 real and synthetic images representing single objects belonging to all six classes. These images have been taken under varying lighting conditions, and they represent both real and synthetic shapes with different orientation and scaling.

The same training set has been used to train both *one-versus-others* and *pair-wise* SVMs in order to allow the second ones to act as a refinement of the boundaries between the various classes with respect to the first set of machines.

A 3×3 median filter is used to reduce noise and undesired mismatch due to artifacts in the background. Moreover, all the input images are normalized with respect to the quantity $\sum_{i,j} I(i,j)^2$ that is a measure of the global energy content of the image. In this way we obtain that the λ vector components range almost in the same interval for all images.

In the test phase, each ROI is tested against the *winner-takes-all* machines, and only when the recognition percentage score is below 60% we use the *pair-wise* machines. The above threshold has been selected on the basis of the experimental evidence.

Experiments have been carried on both images depicting single objects, and complex scenes with many objects even partially occluded. Experiments have been performed on images depicting real scenes as in figure 5. Tables 1, 2 and 3, and figures 4 and 5 illustrate the very good results of experiments.

Table 1. Performance of the system on the scene depicted in figure 3. The position values are referred to the upper left corner of each slice. In slice two the PW classification is incorrect, but it's not used to the high score obtained in WTA mode. PW refines the outcome of the WTA only in the last case: in fact the PW outcome is near the lower threshold.

Slice n.	Pos.	WTA (%)	PW
0	$(21, 91)$	box (87.03)	cube
1	$(39, 74)$	box (100)	box
2	$(105, 70)$	cylinder (89.52)	box
3	$(105, 152)$	box (68.67)	cube

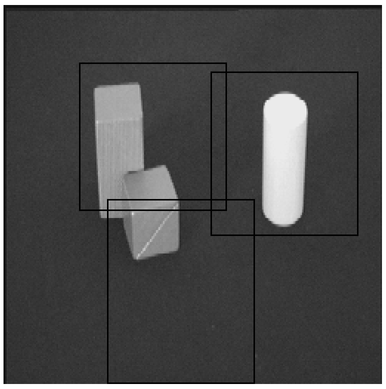


Figure 4. The performance of the system on a multiple objects image. From top to bottom: the image with the selected slices, and a table reporting the output of the system.

Table 2. Performance of the system on the scene depicted in figure 4. It can be noted that for the first slice we obtain a weak correct response from the WTA machines, while the PW classification is almost wrong due to the closeness between the two shape classes. Many others slices, detected by the search algorithm, have been discarded by the classifier.

Slice n.	Pos.	WTA (%)	PW
0	$(52, 40)$	box (48.76)	cube
1	$(71, 132)$	box (69.87)	cube
2	$(141, 46)$	cylinder (100)	cylinder

5 CONCLUSIONS AND FUTURE WORK

The presented work is a first step towards a robust object recognition system, that is suitable both as an identification stage of a 3D vision system, and as a content description module for content-based image retrieval applications. Early results are satisfactory and provide us with many cues about future developments in complex scenes.

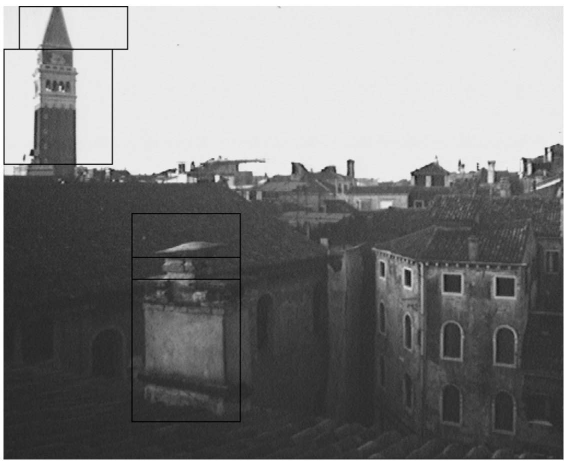


Figure 5. The performance of the system on a real image, depicting the city of Venice. Here, the slices have been selected interactively.

Table 3. Performance of the system on the scene depicted in figure 5. Slice 1 is misclassified as a cone due to the strong similarity between one side of the actual pyramid and the background. Slice 3 is correctly classified by the WTA machine, and the PW response is not taken into account.

Slice n.	Pos.	WTA (%)	PW
0	$(1, 40)$	box (100)	box
1	$(15, 1)$	cone (57.10)	cube
2	$(118, 190)$	box (57.49)	box
3	$(118, 230)$	box (89.48)	box/cylinder

The use of a statistical approach makes the system quite robust with respect to noise, but the system fails in presence of textures. One might think to specialize the system to the recognition of textures as a global feature, while shape could be argued using some other approach. Another interesting extension is the integration of our approach with a database of different views of each shape, to provide disambiguation in situations like the second slice in figure 5.

We are investigating in more detail the influence of the illuminant direction. Our approach has proven itself robust with respect to this parameter due to the fact that $\mathbf{l}$ affects all the elements of the covariance matrix in the same way.

An open field of investigation is the correct segmentation of the scene to provide useful data to the location module of the vision system. Our approach performs a sort of rough region location, even in presence of partially occluded objects (see figure 4). This can be a suitable starting point for a detailed segmentation which could be performed by the reconstruction module. In our 3D vision system an *a priori* color based 2D segmentation of the scene is implemented, which provides the identification module with suitable regions of interest [1].

Finally, we are investigating the use of the SVMs in order to derive model parameters for unknown objects by interpolation from those used to describe the training samples. In the current implementation of our vision system we model shapes by means of superquadrics [3].

ACKNOWLEDGEMENTS

This work has been partially supported by the Italian MURST Project "Galileo 2000" and MURST-CNR Biotechnology Program l. 95/95, and MURST Project "CERTAMEN".

REFERENCES

[1] C. Amoroso, E. Ardizzone, V. Morreale, and P. Storniolo. A New Technique for Color Image Segmentation. In *Proc. of International Conference on Image Analysis and Processing*, pages 352–357, Venice, Italy, 1999.

[2] D.H. Ballard and C.M. Brown. *Computer Vision*. Prentice Hall, Englewood Cliffs, NJ USA, 1982.

[3] A.H. Barr. Superquadrics and Angle-preserving Transformations. *IEEE Computer Graphics and Applications*, 1:11–23, 1981.

[4] O. Faugeras. *Three-Dimensional Computer Vision, A Geometric Viewpoint*. The MIT Press, Cambridge, MA USA, 1990.

[5] R.C. Gonzalez and P. Wintz. *Digital Image Processing*. Addison-Wesley, ii edition, 1987.

[6] I.T. Joliffe. *Principal Component Analysis*. Springer–Verlag, New York, 1986.

[7] U. Kreßel and J. Shürmann. Polynomial Classifiers and Support Vector Machines. In *Artificial Neural Networks - ICANN'97*, pages 397–402, Amsterdam, 1997. North-Holland.

[8] H. Murase and S.K. Nayar. Visual Learning and Recognition of 3-D Objects from Appearance. *International Journal of Computer Vision*, 14:5–24, 1995.

[9] A. Pentland and M. Turk. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[10] M. Schmidt and H. Gish. Speaker Identification via Support Vector Classifiers. In *Proc. ICASSP'96*, pages 105–108, Atlanta, GA USA, 1997.

[11] B. Schölkopf, C. Burges, and A.J. Smola, editors. *Support Vector Learning*. Advances in Kernel Methods. The MIT Press, Cambridge, MA USA, 1999.

[12] J. Shürmann. *Pattern Classification: a Unified View of Statistical and Neural Approaches*. Wiley, New York, 1996.

[13] A. Talukder and D. Casaent. General Methodology for Simultaneous Representation and Discrimination of Multiple Object Classes. *Optical Engineering*, 37(3):904–913, March 1998.

[14] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, Upper Saddle River, NJ USA, 1998.

[15] V.N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

Learning Actions: Induction over Spatio-Temporal Relational Structures - CRG$_{\rm ST}$

Walter F. Bischof and Terry Caelli [1]

Abstract. We introduce a rule-based approach for learning and recognition of complex actions in terms of spatio-temporal attributes of primitive event sequences. During learning, spatio-temporal decision trees are generated that satisfy relational constraints of the training data. The resulting rules, in form of Horn clause descriptions, are used to classify new dynamic pattern fragments, and general heuristic rules are used to combine classification evidences of different pattern fragments.

1 Introduction

Most current techniques for the encoding and recognition of actions use numerical machine learning models which are not relational in so far as they typically induce rules over numerical attributes which are not indexed or linked via an underlying data structure (e.g. a relational structure description or a directed acyclic graph, DAG). Therefore most learning models assume that the correspondence between candidate and model features is known *before* rule generation (learning) or rule evaluation (matching) occurs. This assumption is dangerous when large models or test data are involved, as is the case in complex actions involving, for example, the tracking of multiple limb segments of human operators. On the other hand well known symbolic relational learners like Inductive Logic Programming (ILP) are not designed to apply efficiently to numerical data. So, although they are suited to induction over relational structures (e.g. Horn clauses), they typically generalize or specialize over the symbolic variables and not so much over numerical attributes. More specifically, it is very rare that symbolic representation *explicitly* constrains the types of permissible numerical learning or generalizations obtained from training data.

Over the past six years we have explored methods for combining the strengths of both sources of model structures [1, 2, 3] by combining the expressiveness of ILP with the generalization models of numerical machine learning to produce a class of numerical relational learning which induce over numerical attributes in ways which are constrained by relational pattern or shape models. Our approach, Conditional Rule Generation (CRG), generates rules that consist of attributed linked lists of pattern (shape) features which, together, completely cover the training data but the generated rules are ordered with respect to their discriminatory power with respect to both attributes and features (see Figure 1).

Since CRG induces over a relational structure it requires general model assumptions, the most important being that the models (shapes) are defined by a labeled graph where relational attributes are defined only with respect to neighboring vertices. Such assumptions constrain the types of unary and binary features which can be used to resolve uncertainties (Figure 1).

[1] Department of Computing Science, University of Alberta, Edmonton, T6G 2H1, Canada, Email: (wfb,tcaelli)@ualberta.ca

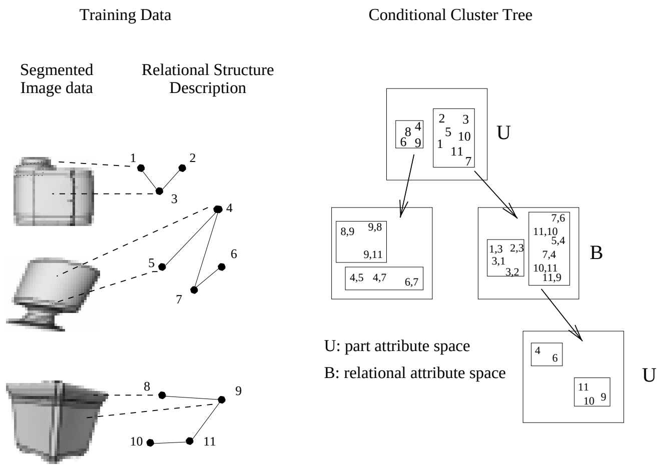


Figure 1. Example of input data and conditional cluster tree generated by CRG method. The left panel shows segmented input data with a sketch of the relational structure descriptions generated for these data. The right panel shows a cluster tree generated for the data on the left. Classification rules of the form $U_i - B_{ij} - U_j \ldots$ are derived directly from this tree [6].

In this paper, we describe CRG$_{\rm ST}$, a spatio-temporal extension of CRG for learning dynamic patterns and its application to animated scenes. We discuss representational issues, rule generation models and rule application. The inclusion of time makes modeling and algorithmic issues more challenging and requires the addition of further assumptions to make the problem tractable.

2 Conditional Rule Generation

In Conditional Rule Generation [1], classification rules for patterns or pattern fragments are generated that include structural pattern information to the extent that is required for classifying correctly a set of training patterns. CRG analyzes unary and binary features of connected pattern components and creates a tree of hierarchically organized rules for classifying new patterns. Generation of a rule tree proceeds in the following manner (see Figure 1).

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. Some of these clusters may be unique with respect to class membership and provide a classification rule: If a pattern contains a part p_r whose unary features $\vec{u}(p_r)$ 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}. Again, some clusters may be unique 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 $\vec{b}(p_r, p_s)$ of the pair $\langle p_r, p_s \rangle$ 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 resolved 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 generated rules are split into more discriminating rules using an entropy-based splitting procedure where the elements of a cluster are split along a feature dimension such that the normalized partition entropy $H_P(T) = (n_1 H(P_1) + n_2 H(P_2))/(n_1 + n_2)$ is minimized, where H is entropy. Rule splitting continues until all classification rules are unique or some termination criterion has been reached. This results in a tree of conditional feature spaces (Figure 1), and within each feature space, rules for cluster membership are developed in the form of a decision tree. Hence, CRG generates a tree of decision trees.

CRG generates classification rules for pattern fragments in the form of symbolic, possibly fuzzy Horn clauses. When the classification rules are applied to some new pattern one obtains one or more (classification) evidence vectors for each pattern fragment, and the evidence vectors have to be combined into a single evidence vector for the whole pattern. The combination rules can be learned [12], they can be knowledge-guided [7], or they can be based on general compatibility heuristics [2]. In the latter approach, sets of instantiated classification rules are analyzed with respect to their compatibilities and rule instantiations that lead to incompatible interpretations are removed. This is particularly important in scenes composed of multiple patterns where it is unclear whether a chain $p_i - p_j - \ldots - p_n$ of pattern parts belongs to the same pattern or whether it is "crossing the boundary" between different patterns. Through application of these compatibility heuristics, we solve two problems at the same time, namely classification of pattern parts and segmentation of different patterns, eliminating the requirement of having to group the image into regions corresponding to single objects before the image regions have been classified.

3 CRG$_{\mathrm{ST}}$

We now turn to CRG$_{\mathrm{ST}}$, a generalization of CRG from a purely spatial domain into a spatio-temporal domain. Here, data consist typically of time-indexed pattern descriptions, where pattern parts are described by unary features, spatial part relations by (spatial) binary features, and changes of pattern parts by (temporal) binary features. In the following sections, we discuss representational issues, rule generation models, learning paradigms and applications of the CRG$_{\mathrm{ST}}$ approach. In contrast to more popular temporal learners like hidden Markov models [11] and recurrent neural networks [4], the rules generated from CRG$_{\mathrm{ST}}$ are not limited to first-order time differences but can utilize more distant (lagged) temporal relations as a function of the data model and uncertainty resolution strategies. At the same time, CRG$_{\mathrm{ST}}$ allows for the generation of non-stationary rules, unlike stationary 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 periods.

3.1 Representation of Spatio-Temporal Patterns

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 $P_i = \{p_{i1}(\vec{a} : t_{i1}), \ldots, p_{in}(\vec{a} : t_{in})\}$ where $p_{ij}(\vec{a} : t_{ij})$ corresponds to part j of pattern i with attributes $\vec{a}$ that are true at time j. The attributes $\vec{a}$ are defined with respect to specific labeled features, and are restricted to arity 1 (unary, i.e. single feature attributes) or 2 (binary, i.e. relational feature attributes), that is, $\vec{a} = \{\vec{u}, \vec{b}_s, \vec{b}_t\}$ (see Figure 2). Examples of unary attributes $\vec{u}$ include area, brightness, position; spatial binary attributes $\vec{b}_s$ include distance, relative size, and temporal binary attributes $\vec{b}_t$ include changes in unary attributes over time, such as size, orientation change, long range position change, etc. Our data model and consequent rules are subject 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 that the temporal indices for time must be monotonically increasing ("predictive" model) or decreasing ("causal" model). Although this limits the expressive power of our representation, it is still more general than strict first-order discrete time dynamical models such as, for example, hidden Markov models or Kalman filters.

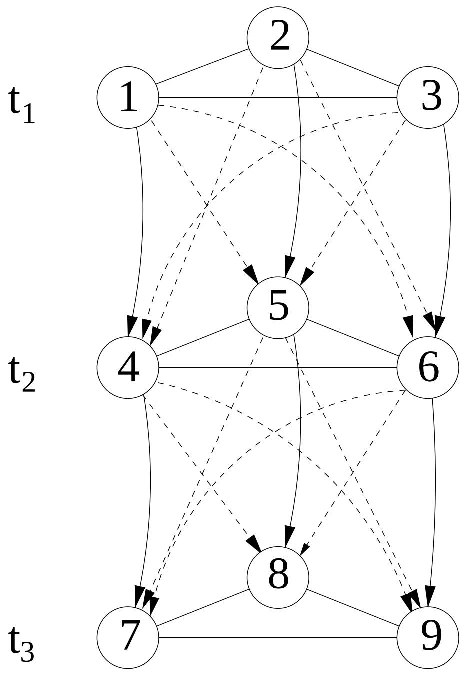


Figure 2. Illustration of a spatio-temporal pattern consisting of three parts over three time-points. Undirected arcs indicate spatial binary connections, solid directed indicate temporal binary connections between the same part at different time-points, and dashed directed arcs indicate temporal binary connections between different parts at different time-points.

For CRG$_{\mathrm{ST}}$ an "interpretation" then involves determining the smallest set of linked lists of attributed and labeled features, causally indexed (i.e. the starting times must be monotonically indexed) over time, which maximally index a given pattern, and it is defined by directed paths within the directed acyclic graph (DAG) which covers all examples and classes in the training set as,illustrated in Figure 2.

3.2 Rule Learning

CRG$_{\mathrm{ST}}$ generates classification rules for spatio-temporal patterns involving a small number of pattern parts subject to the following constraints: 1) The pattern fragments involve only pattern parts that are adjacent in space and time, 2) the pattern fragments involve only non-cyclic chains of parts, 3) temporal links are followed in the forward direction only to produce causal classification rules that can be used in classification and in prediction mode.

Rule learning proceeds in the following way: First, the unary features of all parts (of all patterns at all time points), $\vec{u}(p_{it})$, $i = 1, \ldots, n$, $t = 1, \ldots, T$, are collected into a unary feature space U in which each each point represents a single pattern part at any time point $t = 1, \ldots, T$. From this unary feature space, cluster tree expansion can proceed in two directions, in the spatial domain and in the temporal domain. In the spatial domain cluster tree generation proceeds exactly as described in Section 2 following spatial binary relations, etc. In the temporal domain, binary relations can be followed only in strictly forward (predictive) or backward (causal) directions, analyzing recursively temporal changes of either the same part, $\vec{b}_t(p_{it}, p_{it+1})$ (solid arrows in Figure 2), or of different pattern parts, $\vec{b}_t(p_{it}, p_{jt+1})$ (dashed arrows in Figure 2) at subsequent timepoints. This leads to a conditional cluster tree as shown in Figure 1, except that the relational attribute spaces B can be either spatial or temporal, in accordance with the usual Minimum Description Length (MDL) criterion for Decision Trees[9].

CRG$_{\mathrm{ST}}$ produces classification rules of the form $U_i - B_{ij} - U_j - B_{jk} - \ldots$ involving spatial and/or temporal binary relations. The resultant Horn clause rules are of the form:

class $\Leftarrow$ part(i at time j with attributes k) AND
part relations(ij at time j+n with attributes u) AND
part(l at time j+m: with attributes s) AND
...

These rules cover all training examples and define a path in the DAG discussed above.

From the empirical class frequencies of all training patterns one can derive an expected classification vector, or evidence vector $\vec{E}$ associated with each rule. We can also compute evidence vectors for partial rule instantiations, again from empirical class frequencies of non-terminal cluster spaces. Hence, an evidence (classification) vector $\vec{E}$ is available for every partial or complete rule instantiation, as discussed in the next section.

3.3 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 using parallel, iterative deepening, subject to the constraints the only adjacent parts are involved and no loops are generated. Note, again, that spatio-temporal adjacency and temporal monotonicity constraints are used for rule generation. Each chain is classified using the classification rules. Expansion of each chain $S_i = < p_{i1}, p_{i2}, \ldots, p_{in} >$ terminates if one of the following conditions occurs: 1) the chain cannot be expanded without creating a cycle, 2) all rules instantiated by S_i are completely resolved, or 3) the binary features $\vec{b}_s(p_{ij}, p_{ij+1})$ or $\vec{b}_t(p_{ij}, p_{ij+1})$ do not satisfy the features bounds of any rule.

If a chain S cannot be expanded, the evidence vectors of all rules instantiated by S are averaged to obtain the evidence vector $\vec{E}(S)$ of the chain S. Further, the set $\mathcal{S}_p$ of all chains that start at p is used to obtain an initial evidence vector for part p:

$$\vec{E}(p) = \frac{1}{\#(\mathcal{S}_p)} \sum_{S \in \mathcal{S}_p} \vec{E}(S). \tag{1}$$

where $\#(\mathcal{S})$ denotes the cardinality of the set $\mathcal{S}$. Evidence combination based on (1) is adequate if it is known that a single pattern is to be recognized. However, if the test pattern consists of multiple patterns then this simple scheme can easily produce incorrect results because some some part chains may not be contained completely within a single pattern but "cross" spatio-temporal boundaries between patterns. This occurs when actions corresponding to different types cross can intersect in time and/or space. These chains are likely to be classified in a arbitrary way. To the extent that they can be detected and eliminated, the part classification based on (1) can be improved.

We use general heuristics for detecting rule instantiations involving parts belonging to different patterns. They are based on measuring the compatibility of part evidence vectors and chain evidence vectors. More formally, the compatibility measure can be characterized as follows. For a chain $S_i =< p_{i1}, p_{i2}, ..., p_{in} >$,

$$\vec{w}(S_i) = \frac{1}{n} \sum_{k=1}^{n} \vec{E}(p_{ik}) \tag{2}$$

where $\vec{E}(p_{ik})$ refers to the evidence vector of part p_{ik}. Initially, this can be found by averaging the evidence vectors of the chains which begin with part p_{ik}. Then the compatibility measure is used for updating the part evidence vectors using an iterative relaxation scheme [8]:

$$\vec{E}^{(t+1)}(p) = \Phi\left(\frac{1}{Z} \sum_{S \in S_p} \vec{w}^{(t)}(S) \otimes \vec{E}(S)\right), \tag{3}$$

where Φ is the logistic function, Z a normalizing factor $Z = \sum_{S \in S_p} w^{(t)}(S)$, and the binary operator $\otimes$ is defined as a component-wise vector multiplication $[a\ b]^T \otimes [c\ d]^T = [ac\ bc]^T$. The updated part evidence vectors then reflect the partitioning of the test pattern into distinct subparts.

4 Example

The CRG$_{\mathrm{ST}}$ approach is illustrated in an example where three different variations of grasp movements were learned: 1) where the hand moved in a straight path to the object, 2) where an obstacle in the direct path was avoided by moving over it, and 3) where the obstacle was avoided by moving around it.

The movements were recorded using a Polhemus system [10] running at 120Hz for three sensors, one on the upper arm, one on the forearm, and one on the hand (see Figure 3). Each movement type was recorded five times. 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 $\tau(t)$ were extracted. Sample time-plots of these measurements are shown in Figure 4.

For these data, the definition of the spatio-temporal patterns is straightforward. At every time point, the patterns consist of three

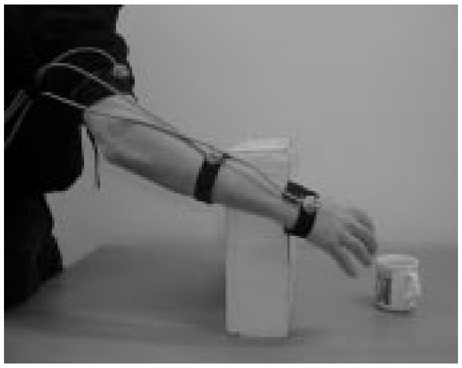


Figure 3. Grasping movement around an obstacle. The movement sensors were placed on the upper arm, the forearm, and the hand.

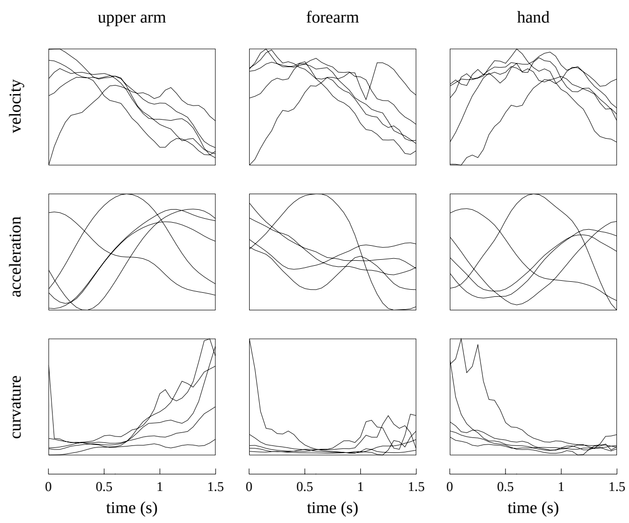


Figure 4. Sample timeplots of the movement sequences illustrated in Figure 3. The left column shows traces for the upper arm, the middle column for the forearm and the right column for the hand. The first row shows time-plot for velocity (for a straight grasp movement), the second for acceleration (for 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.

parts, one for each sensor, each part being described by unary attributes $\vec{u} = [x, y, z, v, a, k, \tau]$. Binary attributes were defined by simple differences, i.e. the spatial attributes were defined as $\vec{b}_s(p_{it}, p_{jt}) = \vec{u}(p_{it}) - \vec{u}(p_{jt})$, and the temporal attributes were defined as $\vec{b}_t(p_{it}, p_{jt+1}) = \vec{u}(p_{jt+1}) - \vec{u}(p_{it})$.

Performance of CRG$_{\mathrm{ST}}$ was tested with 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. Pattern learning and pattern classification proceeded exactly as described in Sections 3.2 and 3.3. Results of these tests are shown in Table 1 for different attribute combinations for unary, spatial binary and temporal binary relations. The last column indicates what percentage of pattern points was classified correctly on average. Although each test pattern consisted of a single movement, this was not assumed by the classification algorithm in order to show the basic classification performance. Using the "single-movement" assumption, e.g. in a winner-take-all scheme, would lead to somewhat higher classification percentages.

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.

$\vec{u}$	$\vec{b}_s$	$\vec{b}_t$	correct
xyz	xyz	xyz	95.4%
-	xyz	xyz	96.3%
-	-	xyz	43.1%
va	va	va	52.2%
-	va	va	46.6%
-	-	va	28.3%
kτ	kτ	kτ	34.6%
-	kτ	kτ	40.7%
-	-	kτ	28.9%
xyzva	xyzva	xyzva	90.8%
-	xyzva	xyzva	96.5%
-	-	xyzva	33.1%

Table 1. Performance of CRG$_{\mathrm{ST}}$ 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; τ: torsion.

An example of a classification rule generated by CRG$_{\mathrm{ST}}$ 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 U(t)	$-1.34 \leq V \leq 7.9$ and
	$-2.93 \leq A \leq 1.54$
and T(t,t+1)	$-0.16 \leq \Delta X \leq 0.07$ and
	$-6.51 \leq \Delta Y \leq 5.37$
and U(t+1)	any value
and T(t+1,t+2)	$-5.39 \leq \Delta X \leq 0.08$
	and $-6.51 \leq \Delta Y \leq 5.37$
and U(t+2)	$4.74 \leq V \leq 5.04$ and
	$-.78 \leq A \leq -0.06$
then	this is part of a grasping action
	moving over an obstacle

The results show that $\mathrm{CRG}_{\mathrm{ST}}$ 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, as well as on extensions of temporal coding to allow temporal interval modeling.

5 Conclusions

We have considered an extension of a spatial relational learning model to learning of spatio-temporal patterns such as complex human actions and gestures. What differentiates our $\mathrm{CRG}_{\mathrm{ST}}$ approach from models like hidden Markov models is that the rules are capable of generalizing over higher-order spatial and temporal relations. Further, the resultant rule forms are Horn clauses whose structures and lengths are constrained by the general topology of the underlying models and by a Minimum Description Length criterion.

REFERENCES

[1] W. F. Bischof and T. Caelli, ‘Learning structural descriptions of patterns: A new technique for conditional clustering and rule generation’, *Pattern Recognition*, **27**, 1231–1248, (1994).

[2] W. F. Bischof and T. Caelli, ‘Scene understanding by rule evaluation’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **19**, 1284–1288, (1997).

[3] *Machine Learning and Image Interpretation*, eds., T. Caelli and W. F. Bischof, Plenum, New York, NY, 1997.

[4] T. Caelli, L. Guan, and W. Wen, ‘Modularity in neural computing’, *Proceedings of the IEEE*, **87**, 1497–1518, (1999).

[5] T. Caelli, A. McCabe, and G. Binsted, ‘On the 3D measurement and representations of human actions’, (2000).

[6] T. Caelli, G. West, M. Robey, and E. Osman, ‘A relational learning method for pattern and object recognition’, *Image and Vision Computing*, **17**, 391–401, (1999).

[7] C. Dillon and T. Caelli, ‘Cite – scene understanding and object recognition’, in *Machine Learning and Image Interpretation*, eds., T. Caelli and W. F. Bischof, 119–187, Plenum, New York, NY, (1997).

[8] B. McCane and T. Caelli, ‘Fuzzy conditional rule generation for the learning and recognition of 3d objects from 2d images’, in *Machine Learning and Image Interpretation*, eds., T. Caelli and W. F. Bischof, 17–66, Plenum, New York, NY, (1997).

[9] J. R. Quinlan, ‘Mdl and categorical theories (continued)’, in *Proceedings of the 12th International Conference on Machine Learning*, pp. 464–470, (1995).

[10] F. H. Raab, E. B. Blood, T. O. Steiner, and H. R. Jones, ‘Magnetic position and orientation tracking system’, *IEEE Transactions on Aerospace and Electronic Systems*, **AES-15**, 709–, (1979).

[11] L. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, New York, NY, 1993.

[12] D. H. Wolpert, ‘Stacked generalization’, *Neural Networks*, **5**, 241–259, (1992).

An High-Level Vision System for the Symbolic Interpretation of Dynamic Scenes by the ARSOM Neural Networks

Antonio Chella[1], **Donatella Guarino**[2], **Ignazio Infantino**[3] and **Roberto Pirrone**[4]

Abstract. We describe an artificial high-level vision system for the symbolic interpretation of data coming from a video camera that acquires the image sequences of moving scenes. The system is based on ARSOM Neural Networks that learn to generate the perception grounded predicates obtained by image sequences. The ARSOM Neural Networks also provide a 3D estimation of the movements of the relevant objects in the scene. The vision systems has been employed in two scenarios: the monitoring of a robot arm suitable for space operations, and the surveillance of a EDP center.

1 INTRODUCTION

We describe an artificial high-level vision agent for the symbolic interpretation of data coming from a video camera that acquires image sequences of moving objects and persons. The agent generates the perception grounded predicates that suitably describe the dynamic scenes.

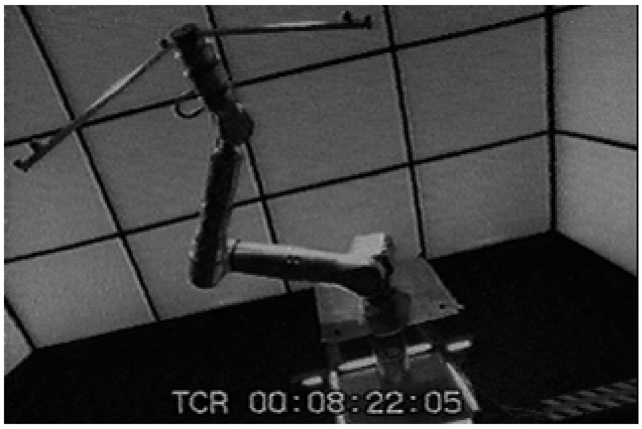


Figure 1. The SPIDER scenario.

The agent integrates a *perception component* which is based on robust techniques of computer vision, with a *scene description component* based on ARSOM Neural Networks [8, 9] that generate the symbols describing the dynamic scene. The results of these components are the input of the *visualization component* that presents the data results by an advanced user interface.

The vision agent is aimed at advancing the state of art in the field of robotics by introducing and integrating different AI techniques that offer a unique opportunity for providing effective greater degrees of autonomy for robotic systems [2, 3].

The vision agent has been employed in two scenarios of interest:

SPIDER In this scenario, images come from a video camera that acquires the movements of the SPIDER robot arm [4, 5, 11] (built by the Italian Space Agency for space applications) during its operations. The agent generates the perception grounded predicates obtained by image sequences thus allowing the scientist user of SPIDER to receive meaningful feedback of his operations on the arm during a scientific experiment (Fig. 1).

CUC In this scenario, images come from a video camera posed at the entrance of the EDP Center (CUC) of the University of Palermo. The agent generates the description of the postures of a person at the entrance of the center, for generating attention degrees of the surveillance persons (Fig. 2).

Figure 2. The CUC scenario.

2 THE PERCEPTION COMPONENT

The perception component of the proposed system processes the image data coming from a video camera that acquires the images of the

[1] Dip. diIngegneria Automatica e Informatica, Univ. di Palermo and CERE-CNR, Palermo, Italy, email: chella@unipa.it
[2] Dip. diIngegneria Automatica e Informatica, Univ. di Palermo and CERE-CNR, Palermo, Italy, email: dony@cere.pa.cnr.it
[3] Dip. di Ingegneria Elettrica, Univ. di Palermo, Italy, email: ignazio@csai.unipa.it
[4] Dip. diIngegneria Automatica e Informatica, Univ. di Palermo and CERE-CNR, Palermo, Italy, email: pirrone@unipa.it

moving scene. The main task of this component in both scenarios is to find the interesting points in the dynamic scene along with their motion.

In particular, in the SPIDER scenario the interesting points are the joint positions of the arm. It should be noted that this estimation, which is solely generated by the visual data, may be useful for fault identifications of the position sensors placed on the joints of the arm.

Similarly, in the CUC scenario, the interesting points are the characteristic points describing the posture of the persons at the entrance of the EDP center.

The images acquired by the camera are processed to extract the contours of the object of interest by a suitable algorithm based on *snakes* [1]. A snake is a deformable curve that moves in the image under the influence of forces related to the local distribution of the gray levels. When the snake reaches an object contour, it is adapted to its shape.

Formally, a snake as an open or closed contour is described in a parametric form by:

$$v(s) = (x(s), y(s)) \tag{1}$$

where $x(s)$ and $y(s)$ are the coordinates along the shape contour and s is the normalized arc length:

$$s \in [0, 1] \tag{2}$$

In the SPIDER scenario, the adopted snake model is based on circles and squares to better extract the arm components; in the CUC scenario the adopted snake model is based on a closed contour adapting to the person shape.

The snake model defines the snake energy of a contour E_{snake}, to be:

$$E_{snake}(v(s)) = \int_0^1 (E_{int}(v(s)) + E_{image}(v(s))ds \tag{3}$$

The energy integral is a functional since its variable s is a function (the shape contour). The internal energy E_{int} is formed from a Tikhonov stabilizer and is defined by:

$$E_{int}(vvs)) = a(s)\left|\frac{dv(s)^2}{ds^2}\right| + b(s)\left|\frac{dv(s)^2}{ds^2}\right|^2 \tag{4}$$

where $|.|$ is the Euclidean norm.

The first order continuity term, weighted by $a(s)$, let the contours behave elastically, whilst the second order curvature term, weighted by $b(s)$, let it be resistant to bending. For example, setting $b(s) = 0$ at point s, allows the snake to become second-order discontinuous at point and to generate a corner.

The image functional determines the features which will have a low image energy and hence the features that attract the contours. In general, this functional is made up by three terms:

$$E_{image} = w_{line}T_{line} + w_{edge}E_{edge} + w_{term}E_{term} \tag{5}$$

where w denotes a weighting constant. The w and E corresponds to lines, edges and termination, respectively.

The snake model adopted in our scenarios presents only the edge functional which attracts the snake to points with an high edge gradient:

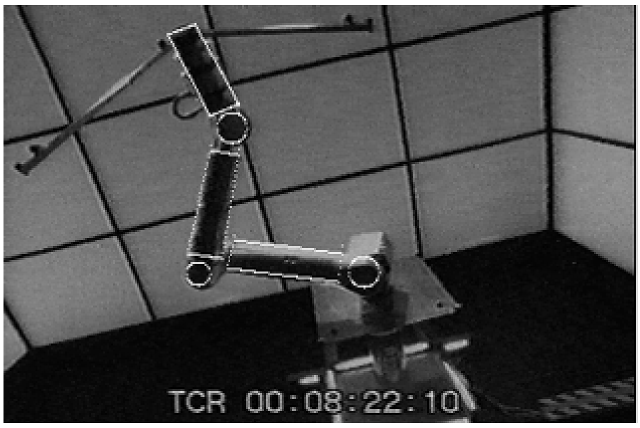


Figure 3. Contours extracted in the SPIDER scenario.

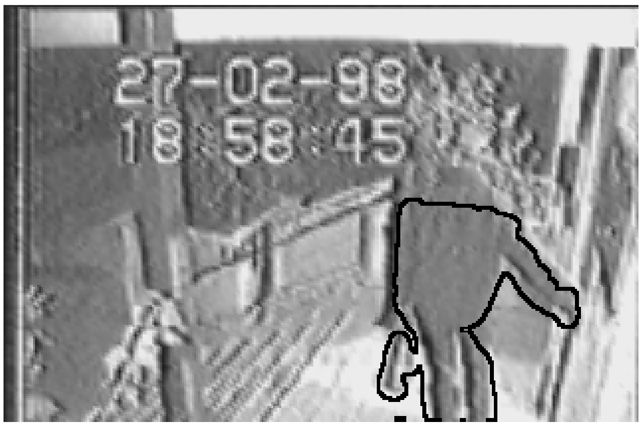


Figure 4. Contours extracted in the CUC scenario.

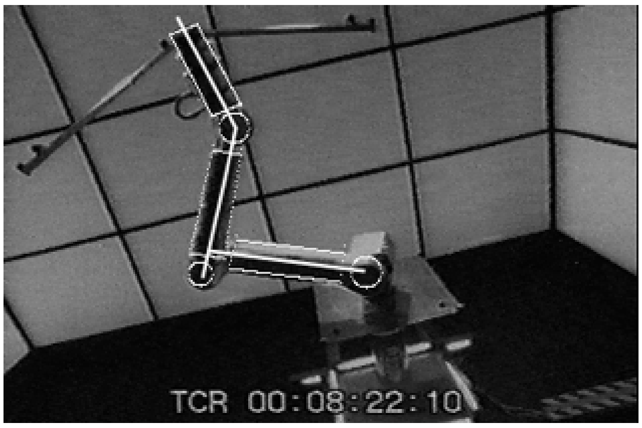


Figure 5. The skeleton extracted in the SPIDER scenario.

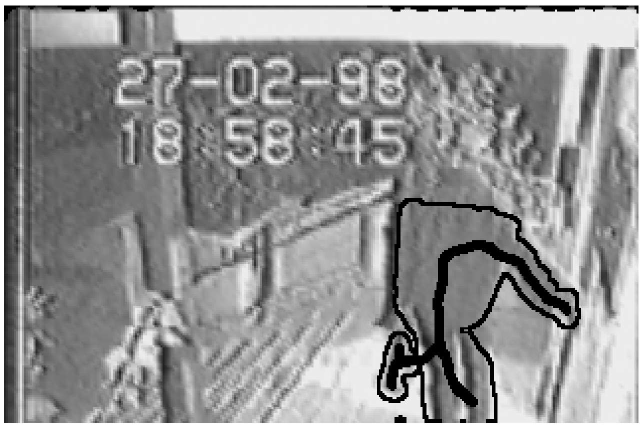


Figure 6. The skeleton extracted in the CUC scenario.

$$E_{image} = E_{edge} = -(G_\sigma * \nabla^2 I(x,y))^2 \tag{6}$$

This is the image functional proposed by Kass, Witkin and Terzoupolos [6]. It is a scale based edge operator that increases the locus of attraction of energy minimum. G_σ is a Gaussian of standard deviation sigma which controls the smoothing process prior to edge operator. Minima of E_{edge} lies on zero-crossing of $G_\sigma * \nabla^2 I(x,y)$ which defines the edges according to the theory of Marr [10].

The implemented snake allows to extract the interesting parts of the scenarios in a simple way and in short time. Fig. 3 shows the results in the SPIDER scenario and Fig. 4 shows the results in the CUC scenario.

After this step, we employ the well known skeletonizing algorithm of Zhang and Suen [13] to extract the skeletons of the areas so found. Fig. 5 shows the skeleton in the SPIDER scenario and Fig. 6 shows the skeleton in the CUC scenario.

3 THE SCENE DESCRIPTION COMPONENT

From the extracted skeletons it is immediate to estimate the position of the interesting points previously described, characterizing the posture of the SPIDER arm or the posture of a person in the CUC scenario.

Let us consider a generic interesting point i of the scene in a scenario at time t; the point is characterized by its 3D coordinates:

$$x_i(t), y_i(t), z_i(t) \tag{7}$$

A generic posture at time t of the robot arm or of a person is characterized by the vector $\mathbf{x}(t)$ which individuates the m interesting points describing the posture itself:

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t), y_1(t), z_1(t) \\ x_2(t), y_2(t), z_2(t) \\ \vdots \\ x_n(t), y_n(t), z_m(t) \end{bmatrix} \tag{8}$$

The snake information allows us to estimate only the first and the second coordinates of each point, i.e., their projection in the image plane:

$$\mathbf{x}'(t) = \begin{bmatrix} x_1(t), y_1(t), . \\ x_2(t), y_2(t), . \\ \vdots \\ x_n(t), y_m(t), . \end{bmatrix} \tag{9}$$

The scene description component receives as input the vector $\mathbf{x}'$ from the perception component and it generates a symbolic description of the posture. This component is based on the ARSOM neural network, a self-organizing neural network with a suitable explicit representation of time sequences [8, 9].

Each unit of the ARSOM is an autoregressive (AR) filter, able to classify and recognize variable inputs. Therefore, each unit characterizes a sequence of movements of the posture points. The map auto-organizes itself during an unsupervised learning phase, as a standard SOM map.

Let us consider a generic movement of the robot arm or of a person. The movement is characterized by a sequence of n posture points:

$$\mathbf{x}(t), \mathbf{x}(t-1), \ldots, \mathbf{x}(t-(n-1)) \tag{10}$$

The AR model associated with this movement is:

$$\begin{aligned} \mathbf{x}(t+1) = \mathbf{A}_0\mathbf{x}(t) \quad &+ \quad \mathbf{A}_1\mathbf{x}(t-1) + \ldots \\ \ldots \quad &+ \quad \mathbf{A}_{n-1}\mathbf{x}(t-(n-1)) + \mathbf{e}(t) \end{aligned} \tag{11}$$

The order of this AR model is n, the $\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_{n-1}$ matrices are the weights of the model, and $\mathbf{e}(t)$ is the error matrix.

Let us denote with $\mathbf{B}$ the global matrix related to the weight matrices:

$$\mathbf{B} = [\mathbf{A}_0, \mathbf{A}_1, \ldots, \mathbf{A}_{n-1}]^T \tag{12}$$

and with $\mathbf{X}(t)$ the global matrix related to the time evolution of the posture points:

$$\mathbf{X}(t) = [\mathbf{x}(t), \mathbf{x}(t-1), \ldots, \mathbf{x}(t-(n-1))]^T \tag{13}$$

We may write Eq. (11) in a more compact form:

$$\mathbf{x}(t+1) = \mathbf{X}^T(t)\mathbf{B} + \mathbf{e}(t) \tag{14}$$

The optimal weights matrices are found by minimizing the error matrix $\mathbf{e}(t)$. We have adopted the *LMS* iterative method, that is:

$$\mathbf{B}_{new} = \mathbf{B}_{old} + h_{ci}\mathbf{e}(t)\mathbf{X}(t) \tag{15}$$

where h_{ci} is the neighborhood kernel:

$$h_{ci} = \begin{cases} 1/2r^2 & \text{if} \quad i \in N_c \\ 0 & \text{if} \quad i \notin N_c \end{cases} \tag{16}$$

In this equation, r is a suitable parameter and N_c is the width of the learning window.

The neural network, after a careful training phase, is able to classify the temporal sequences of movements of the interesting points into meaningful prototypical predicates.

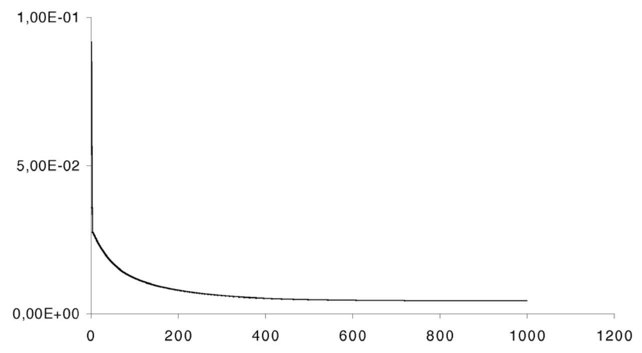


Figure 7. Error vs. learning epochs of the ARSOM network.

Currently, we have two similar ARSOM neural networks: one for the SPIDER scenario and the other for the CUC scenario. We are experimenting the possibility to have one networks for both scenarios.

Fig. 7 shows the diagram of the error of the neural network during the training phase. It should be noted that, after a few hundred learning steps, the error of the network is near zero value. The figure is related with the SPIDER scenario; a similar behavior occurs in the CUC scenario.

When the estimation of the coordinates of the interesting point in the image plane are presented to the network:

$$\mathbf{x}'(t), \mathbf{x}'(t-1), \ldots, \mathbf{x}'(t-(n-1)), \tag{17}$$

the network is able to predict the full vector $\mathbf{x}(t+1)$, i.e., the vector with all the three coordinates of the posture.

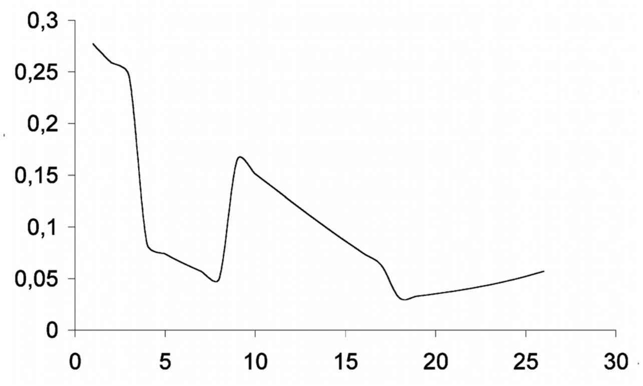


Figure 8. Prediction error of the ARSOM network.

Fig. 8 shows the prediction error of the network during its operations in the SPIDER scenario. It should be noted that the error, while is variable, it maintains in a reasonable limit. Similar behaviors have been observed in the CUC scenario.

Figs. 9 and 10 show the recovered 3D situations, respectively in the SPIDER and CUC scenarios, as predicted by the ARSOM neural networks.

The network is also able to perform a classification of the global arm movement and to present as output a symbolic predicate describing the movement itself.

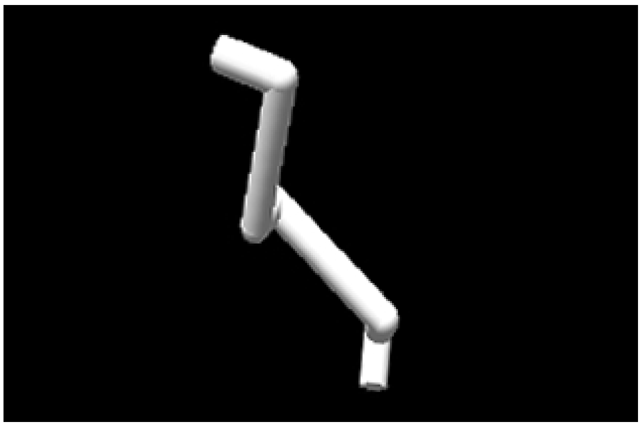


Figure 9. 3D recovery of the SPIDER scenario.

Examples of the learned predicates describing the operations of the arm in the SPIDER scenario are: *Stretching_up*, *Stretching_down*, *Seizing*, *Grasping*. Examples of the learned predicates in the CUC scenario are: *Entering*, *Exiting*, *Opening*, *Closing*, *Looking_inside*, *Staying*.

The neural network approach presents the main advantage that it avoids an explicit description of the discrimination functions for the arm operations, as this function is learned during the training phase.

Furthermore, the neural network is robust with respect to the noise, as it is able to correctly classify the arm operations also when the movements estimations of some links are missing or corrupted.

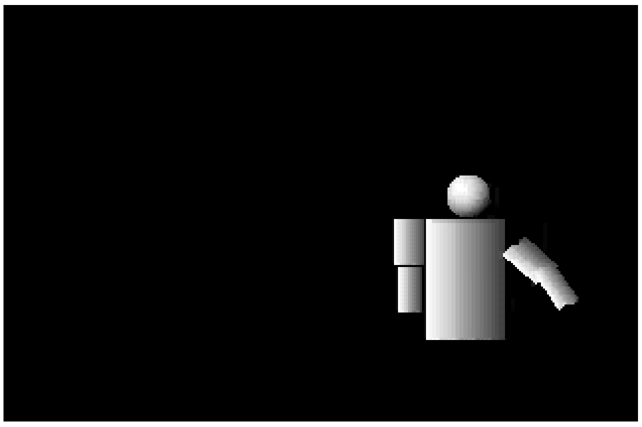


Figure 10. 3D recovery of the CUC scenario.

In the operation tests performed in the SPIDER scenario, the network has been able to perform the 100% success on the classification task. To analyze the operation of the network, tests are performed on the recognition task when some links information is missed. Tab. 11 reports the obtained results. It should be noted that in the worst case, when the two links 1 and 3 are missing, the network is able to perform 51% of success recognition.

Also the performances of the system in the CUC scenario are good. Up to now, we have obtained about 94% of recognition success on the classification task. It should be noted that in this case, we have chosen to take into account only simple actions, as previously described. Currently, we are generalizing the system on a larger set

Miss. link	% Rec.
0	100
1	75
2	74
3	62
1,3	51

Figure 11. Recognition % vs missing link in the SPIDER scenario.

of more rich and realistic situations.

4 THE VISUALIZATION COMPONENT

The scene description component of the system receives as input the data coming from the perception component and of the scene description component, and it generates a graphic 3D representation of the scene. The visualization component provides also the graphic interface for the whole agent. In the following we will describe the interface for the SPIDER scenario; similar consideration hold for the CUC scenario.

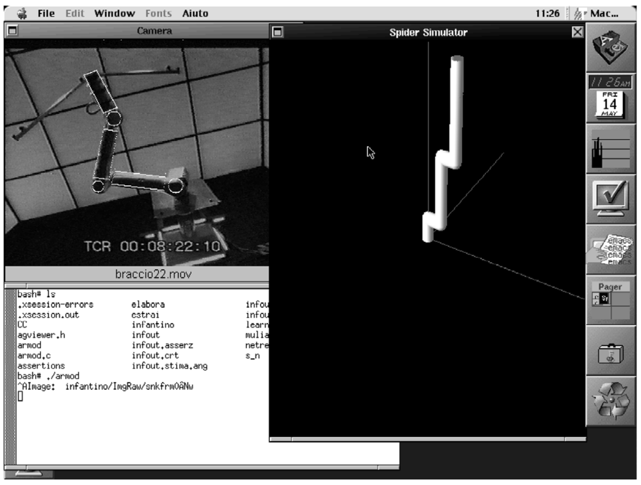

```
bash# ls
.xsession-errors        elabora                infou
.xsession.out           estrai                 infou
CC                      infantino              learn
agviewer.h              infout                 mulia
armod                   infout.asserz          netre
armod.c                 infout.crt             s_n
assertions              infout.stima.ang
bash# ./armod
^AImage:  infantino/ImgRaw/snkfrm0ÂNw
```

Figure 12. The user interface of the system.

Fig. 12 shows the results of the visualization component of the system for the SPIDER scenario. The scientist user of the system may view the arm operations from different point of views and he may navigate in the reconstructed environment.

He may also supervise and intervene in all the processing steps occurring in the agent itself: e.g., he may change the parameters of the perception component modules or he may tune the learning phase of the neural network in the scene description component.

The interface of the system presents several windows in order to provide the user scientist with a full control of the system.

The *camera* window shows the output image sequences of the video camera acquiring the real robot arm operations along with superimposition of the snake representing the output of the contour extraction module.

The 3D window shows the images representing the 3D reconstruction of the arm during its operations, and the *description* window shows the symbolic descriptions generated by the scene description component in terms of symbolic predicates.

A simple user interface based on buttons allows the scientist to modify the inner parameters of the agent in order to tailor the agent processing steps.

The graphical interface has been realized by using the OpenGL and the GLUT library [7, 12].

5 CONCLUSION

The research demonstrated how the implemented artificial high-level vision agent may be an effective tool for monitoring operations. In the case of the SPIDER scenario, the user scientist of the arm can monitor his own operations by providing high-level feedback descriptions of the arm movements during the scientific experiments. In the case of the CUC scenario, the surveillance persons can be alerted of possible dangerous situations near the EDP center that require special attention.

The described system is fully general and it may be employed in all the fields in which the interactive autonomy of the space robotic systems is a mandatory requirement. The system will also give a valuable contribution to the use of the expensive and state of the art equipment related to space robotics and to surveillance robotics. Of great importance are the possible industrial application of the described system. It could be employed in all the applications that require high automatic tasks in interactive autonomy, as the submarine robots and autonomous systems acting in nuclear plants.

ACKNOWLEDGEMENTS

This work has been partially supported by ASI project “Un Sistema Intelligente per la Supervisione di Robot Autonomi nello Spazio” and by MURST project “Progetto Cofinanziato CERTAMEN”.

REFERENCES

[1] A. Blake and M. Isard. *Active Contours*. Springer-Verlag, Berlin, 1998.
[2] A. Chella, M. Frixione, and S. Gaglio. A cognitive architecture for artificial vision. *Artif. Intell.*, 89:73–111, 1997.
[3] A. Chella, M. Frixione, and S. Gaglio. An architecture for autonomous agents exploiting conceptual representations. *Robotics and Autonomous Systems*, 25(3-4):231–240, 1998.
[4] S. Di Pippo, G. Colombina, R. Boumans, and P. Putz. Future potential applications of robotics for the international space station. *Robotics and Autonom. Systems*, 23(1-2):37–43, 1998.
[5] F. Didot, J. Dettmann, S. Losito, D. Torfs, and G. Colombina. JERICO. *Robotics and Autonom. Systems*, 23(1-2):29–36, 1998.
[6] M. Kass, A. Witkin, and D. Terzoupolos. Snakes: active contour models. In *Proc. of First Intern. Conf. on Computer Vision*, pages 259–268. Springer-Verlag, 1987.
[7] M.J. Kilgard. *OpenGL programming for the X Window system*. Addison-Wesley, Reading, MA, 1996.
[8] T. Kohonen. *Self-Organizing Maps, II ed.* Springer-Verlag, Berlin, 1997.
[9] J. Lampinen and E. Oja. Self-organizing maps for spatial and temporal AR models. In *Proc. of the 6th Scandinavian Conference on Image Analysis*, pages 120–127, Oulu, Finland, 1989.
[10] D. Marr. *Vision*. W.H. Freeman and Co., New York, 1982.
[11] R. Mugnuolo, S. Di Pippo, P.G. Magnani, and E. Re. The SPIDER manipulation system (SMS). The italian approach to space automation. *Robotics and Autonom. Systems*, 23(1-2):79–88, 1998.
[12] J. Neider, T. Davis, and M. Woo. *OpenGL programming guide*. Addison-Wesley, Reading, MA, 1996.
[13] T.Y. Zhang and C.Y. Suen. A fast parallel algorithm for thinning digital patterns. *Comm. ACM*, 27(3):236–239, 1984.

An Application of Machine Learning and Statistics to Defect Detection

R. Cucchiara[1], P. Mello[2], M. Piccardi[3], F. Riguzzi[3]

Abstract. We present an application of Machine Learning and Statistics to the problem of distinguishing between defective and non-defective industrial workpieces, where the defect takes the form of a long and thin crack on the surface of the piece. The images of the pieces are described by means of a set of visual primitives, including the Hough transform and the Correlated Hough transform. We have compared an attribute-value learner, C4.5, a backpropagation neural network, NeuralWare Predict, and the statistical techniques linear, logistic and quadratic discriminant for the classification of pieces. Moreover, two feature sets are considered, one containing only the Hough transform and the other one containing also the Correlated Hough Transform. The results of the experiments show that C4.5 performs best for both feature sets and gives an average accuracy of 93.3 % for the first dataset and 95,9 % for the second dataset.

1 INTRODUCTION

We present an application of Machine Learning and Statistics to a problem of Automated Visual Inspection (AVI) that consists of automatically inspecting the quality of metallic industrial workpieces. The aim is to classify each piece as defective or non-defective depending on whether it contains or not surface defects, visible only under UV light. The surface defect is a crack that is visible under UV light as a bright, thin and roughly rectilinear shape.

In order to recognize cracks, a set of visual primitives have been selected for characterizing the images of pieces. In this way, each image is described by a set of numerical attributes and machine learning can be applied in order to find a classifier for new images.

In particular, we use the Hough transform (HT) that has been proposed in the literature of image analysis for detecting straight lines [1]. The HT transforms the image space into another two dimensional space (called Hough space) where each local maximum point corresponds to a straight edge in the image space. Moreover, another transformation is used, the Correlated Hough transform (CHT), that has the specific aim of detecting shapes that are bright, rectilinear and thin [2]. The CHT transforms an image from the Hough space to the Correlated Hough Space where each local maximum point represents a couple of close, straight edges in the image.

In order to test the effectiveness of these various primitives on classification, we have considered two different datasets, one containing features from the Hough and the Correlated Hough space, and another one containing features from the Hough space only.

On the two datasets, we have compared an attribute-value learner, C4.5, a backpropagation neural network, NeuralWare Predict, and the statistical techniques linear, logistic and quadratic discriminant.

The paper is divided as follows: next section introduces the specific application. Section 3 discusses the adopted visual primitives. Section 4 discusses the results of experiments, providing a comparative analysis among the different algorithms. Finally, the last section provide final conclusions.

[1] Dipartimento di Scienze dell'Ingegneria, Università di Modena, Italy, e-mail: rita.cucchiara@unimo.it
[2] D.E.I.S., Università di Bologna, Italy, e-mail: pmello@deis.unibo.it
[3] Dipartimento di Ingegneria, Università di Ferrara, Italy e-mail: {mpiccardi, friguzzi}@ing.unife.it

2 DEFECT DETECTION

The application goal is visual quality inspection of metallic industrial workpieces and in particular the location of surface and subsurface discontinuities in ferromagnetic materials.

This target can not be reached by normal, visible-light inspection but is usually accomplished by adopting a "Magnetic-Particle Inspection" technique (MPI) [3]. First, the piece is magnetized and dipped in a water suspension of fluorescent ferromagnetic particles; then, it is exposed under ultraviolet light and examined by a human inspector. When surface or subsurface defects are present, they produce a leakage field that attracts and concentrates the ferromagnetic particles. Defects can then be easily perceived by the human eye, since ultraviolet light greatly enhances fluorescence. Off-the-shelf CCD cameras and frame grabbers hosted by commercial PCs are used in order to acquire the images.

Examples of images with cracks are shown in figures 1, 2 and 3. Figure 1 shows a whole image, while figures. 2 and 3 show in details two cracks, more and less evident respectively.

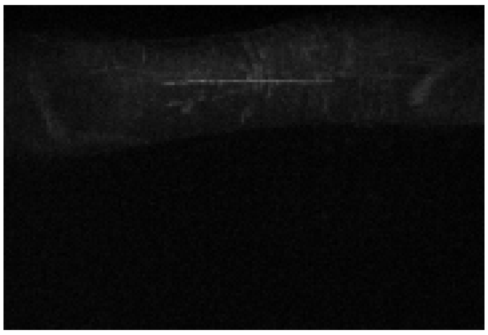


Figure 1. Image with a crack.

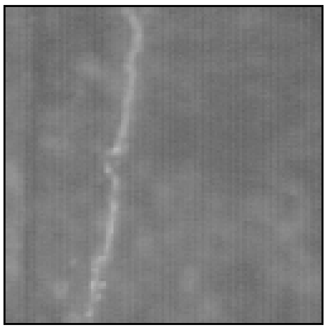


Figure 2. Detail of an evident crack.

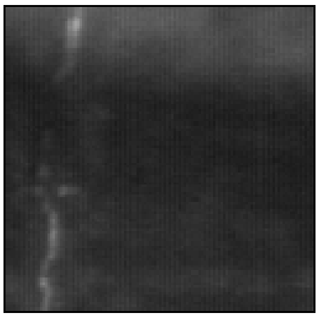


Figure 3. Detail of a less evident crack.

3 CLASSIFICATION BY VISUAL PRIMITIVES

The defect shape was a-priori known by means of a qualitative model provided by human inspectors. They defined it as a "thin, roughly rectilinear and very bright shape".

On the basis of this rather generic model, we elicited a set of visual primitives (properties) that can be used for describing the defects, by associating measurable object properties with the attributes of the qualitative model:

- *bright shape* $\rightarrow$ high local gradient of luminosity in the proximity of its edges;
- *rectilinear* $\rightarrow$ with two main edges approximately straight;
- *thin* $\rightarrow$ with an upper-bounded distance between the two main edges.

Once elicited the visual properties, a set of quantitative image operators able to reflect them has to be defined. Usually, the approach consists of defining a rather large set of image operators, or features, each of them somehow related to one or more visual properties, which will be later selected by a machine learning phase (feature selection). However, the choice of the initial feature set is critical since the information lost at this step cannot be recovered later.

To this aim, we defined and compared two different feature sets, motivated by opposite rationale: in the first set, we included a specialized primitive called Correlated Hough transform (CHT [2]), which has been proposed for detection of objects corresponding exactly to our model; in the second set, we used only image operators of general use. The two feature sets reflect a different control of the visual aspects of the problem, the first one calling for the insight on image operators typical of a computer vision specialist, while the second one requires just an off-the-shelf use of well-known image operators.

Both feature sets include the Hough transform (HT), which essentials is sketched hereafter. The HT has been proposed in the computer vision literature to detect straight lines [1]. It consists of the following space transformation from the image space to a 2-coordinate parameter space: "collinear" points forming a straight line segment in the image space are collected into a single point of the parameter space, where the point's first co-ordinate, ϑ, is the slope of the straight line and the second co-ordinate, ρ, is its distance from the origin. Each point in the Hough space has a value which is exactly the number of collinear points in the straight line segment; thus, the longer the line segment, the higher is the point's value in the Hough space. Furthermore, in this work we adopted a refined version of the Hough transform, called gradient-weighted Hough transform (GWHT, [1]) in which each collinear point is weighted by its luminosity gradient. Therefore, peaks in the Hough space (i.e., points with high values) correspond to the existence of straight, bright lines in the image space, or we could also say that the problem of detecting lines in the image space is converted in the much easier problem of detecting peaks in the Hough space.

In our images, a crack has two edges with similar gradient magnitude (with same direction but opposite orientation); since the crack is thin, the distance between the two edges is upper-bounded. Therefore, two peaks must be detected in the Hough space, with similar values and their ρ, ϑ parameters mutually constrained. In alternative to the separate detection of these two peaks, it is possible to exploit the Correlated Hough transform. The CHT performs a post-processing of the GWHT Hough space by correlating the area where the first peak is detected with the one where the second peak should be located: if it is actually present, the resulting correlation value is very high and can be easily detected. The CHT has been proven robust to non-ideality and noise, since the detection after correlation is more reliable than the detection of the two separate peaks in the Hough space. However, the CHT itself is not enough for detecting cracks when they strongly differ from their ideal aspect, and therefore we added in the feature set many other features related with the model.

The set based on the CHT (called CH dataset) contains the following features:

1. *CH (Correlated_Hough_Peak)*: this is the maximum value in the correlated Hough space; its ρ, ϑ co-ordinates correspond to the parameters of a straight line in the image located on the crack, in case a crack is present.
2. *H1 (First_Hough_Peak)*: this is the value in the same point of the Hough space before correlation, in the range $\vartheta \in [0,\pi]$, where the first peak is formed in case a crack is present.
3. *H2 (Second_Hough_Peak)*: it is the peak in the Hough space between π and 2π at the ideal point were a second straight edge should be found.
4. *H22 (Second_Hough_Average)*: this feature is CH divided by H1; it measures how much the correlation operation increases the evidence of the crack with respect to the uncorrelated Hough space.
5. *Thickness*: the mutual distance between H1 and H2. It represents the object thickness.
6. *Number_of_Points*: the number of voting points accumulated in H1, which estimates the edge length.
7. *Average_Vote*: the average "vote" of the voting points, i.e. the average luminosity gradient of each point voting for H1 (it is computed by dividing H1 by the Number_of_Points); it

measures the average luminosity gradient along the crack profile.

8. *Average_Image_Gradient*: the average luminosity gradient of the image; it is a different property with respect to the others, since it is global, meaning that it is an overall feature of the whole image. It might be used by the classifier as a corrective weight, since images with low values of the average gradient have proportionally lower CH and Hough space values.

Operationally, we acquire images with relevant views of the mechanical piece and for each image we compute the CHT. Then, we detect the CHT maximum (the CH feature) and record a tuple with CH and the other associated feature values. We then detect all the points of the correlated Hough space whose value is greater or equal an assigned percentage of the maximum (75% was used in the experiments), and record a tuple for each of them; this is done in order to catch multiple cracks that can be present in a single image. After acquiring the tuples, we pre-classified each of them into the two categories of *Defect* or *NoDefect* by checking manually if the straight line segment corresponding with the tuple was located on a real crack in the image or not.

In the approach followed, the CHT plays a major role, since the CH maximum is the feature that determines the position where the crack may be located. However, the CHT is a highly specialized operator, and it is interesting to approach the problem with a feature set with more standard features, and comparing the performance of the resulting classifiers.

Therefore, in the second dataset set (called H1 H2 dataset) we excluded the CH value and included the following features:

1. *H1*: the value of the Hough maximum in the range $\vartheta \in [0,\pi]$, where the first peak is formed in case a crack is present; its ρ, ϑ co-ordinates correspond to the parameters of a straight line located on one edge of the crack.
2. *H2*: the value of the Hough maximum in the range $\vartheta \in [\pi, 2\pi]$, where the second peak is formed in case a crack is present; its ρ', ϑ' co-ordinates correspond to the parameters of a straight line located on the other edge of the crack. However, if multiple cracks are present, H1 and H2 may not be associated with the same crack.
3. *Number_of_Votes*: the sum of the number of image points that were transformed into H1 and H2.
4. *Distance*: the mutual distance between H1 and H2 in the Hough space. It represents the object thickness if H1 and H2 correspond to the same crack.
5. *Delta_rho*: the $|\rho' - \rho|$ value, and
6. *Delta_theta*: the $|\vartheta' - \vartheta - \pi|$ value. *Delta_rho* and *Delta_theta* express the distance between the two peaks along the ρ and ϑ directions, respectively. In case of a same real crack, *Delta_theta* should be close to 0 and *Delta_rho* upper bounded. *Delta_rho* and *Delta_theta* are related to *Distance* by the following formula :
$$Distance = \sqrt{Delta\_rho^2 + Delta\_theta^2} \ .$$
7. *Delta_product*: the product delta_rho * delta_theta. It correlates the Delta_rho and Delta_theta values, expecting small values for the product in case of a same real crack.
8. *Average_Image_Gradient*: The average luminosity gradient of the image.

Since there is not an explicit correlation operation between H1 and H2, we also added some basic arithmetic functions of the H1 and H2 values:

9. *Product*: the product H1 * H2: should be high in case of a real crack (about the square of each of the two values).
10. *Ratio*: the ratio H1 / H2: should be close to 1 in case of a real crack.
11. *Sum*: the sum H1 + H2: should be high in case of a real crack (about double each of the two values).
12. *Difference*: the difference H1 - H2: should be close to 0 in case of a real crack.

These arithmetic functions are just combinations of other features and thus may be considered redundant, but they have been explicitely included in the feature set since they are related with the model and may improve the classifiers' performance in case the classifier does not explore linear or quadratic combinations or ratios of the feature values.

Operationally, we acquire images with relevant views of the mechanical piece and for each image we compute the Hough space with the GWHT. Then, we detect the H1 and H2 maxima and record them in a tuple with the other associated feature values. We then repeat the process for all the points of the Hough space in the range $[0, \pi]$ and $[\pi, 2\pi]$ whose value is greater or equal an assigned percentage of H1 and H2, respectively, and record a tuple for each couple; this is done in order to catch multiple cracks that can be present in a single image. After acquiring the tuples, we pre-classified each of them into the two categories of *Defect* or *NoDefect* by checking manually if the straight line segments corresponding with H1 and H2 were located on a same real crack.

4 EXPERIMENTS

We have experimented and compared two different machine learning techniques: attribute-value learning and backpropagation neural networks. Moreover, due to the numeric nature of all the attributes, we have used statistical techniques as well in order to compare their performance with that of machine learning tools.

For attribute-value learning we have used C4.5 [4] that is able to learn both decision trees and rules. For backpropagtion neural networks, we have employed a commercial system, Predict by NeuralWare[1]. As regards statistical techniques, we have used the algorithms Discrim, Logdisc and Quadisc, developed under the Statlog project [5], that implement respectively linear discriminant, logistic discriminant and quadratic discriminant.

In the following, we first give a brief description of each algorithm and then we present the results of experiments.

5 *Discrim*

Discrim finds a linear discriminant, i.e., an hyperplane in the p-dimensional space of the attributes. Given the values of the attributes of a new pattern, its class is found by looking at the position of the corresponding point with respect to the hyperplane.

[1] More information about Predict can be found at http://www.neuralware.com/ .

The hyperplane equation is found on the assumption of normal probability distribution: the attribute vectors for the examples of class A_i are independent and follow a certain probability distribution with probability density function (pdf) f_i. A new point with attribute vector x is then assigned to that class for which the probability density function $f_i(\mathbf{x})$ is greatest. This is a maximum likelihood method. The distribution are assumed normal (or Gaussian) with different means but the same covariance matrix. The probability density function of the normal distribution is

$$f_i(\mathbf{x}) = \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_i)' \Sigma^{-1} (\mathbf{x} - \mu_i) \right) \qquad (1)$$

where μ is a p-dimensional vector denoting the theoretical mean for class i and Σ, the theoretical covariance matrix, is a $p \times p$ matrix that is necessarily positive definite. In this case the boundary separating the two classes, defined by the equality of the pdfs, can be shown to be an hyperplane that passes through the mid-point of the two centres. Its equation is

$$\mathbf{x}'\Sigma^{-1}(\mu_1 - \mu_2) - \frac{1}{2}(\mu_1 + \mu_2)' \Sigma^{-1}(\mu_1 - \mu_2) = 0 \qquad (2)$$

where μ_i is the population mean for class A_i. When using this formula for classification the exact distribution is usually not known and the parameters must be estimated from the available sample. With two classes, if the sample means are substituted for μ_i and the pooled sample covariance matrix for Σ, then Fisher's linear discriminant [6] is obtained. The covariance matrix for a dataset with n_i examples from class A_i is

$$S_i = \frac{1}{n_i - 1} X^T X - \bar{\mathbf{x}}^T \bar{\mathbf{x}} \qquad (3)$$

Where X is the $n_i \times p$ matrix of attribute values and $\bar{x}$ is the p-dimensional row vector of attribute means. The *pooled covariance matrix S* is

$$S = \frac{\sum (n_i - 1) S_i}{n - q} \qquad (4)$$

where the summation is over all the classes and *(n-q)* is chosen to make the pooled covariance matrix unbiased.

6 Quadisc

Quadisc performs a quadratic discrimination. Quadratic discrimination is similar to linear discrimination with the difference that the surface separating the two regions is quadratic. This means that in the discriminating function will contain not only the attributes but also their square and the product of two attributes. With respect to the case of probability maximization seen in the previous case, if we remove the assumption that the normal distributions have the same covariance matrix S, we obtain a quadratic surface, for example an ellipsoid or an hyperboloid.

The simplest quadratic discrimination function for a class is defined as the logarithm of the corresponding probability density function and is given by equation 5 in the case of differing prior probabilities. The suffix i is used to indicate class A_i.

$$\log \pi_i f_i(\mathbf{x}) = \log \pi_i - \frac{1}{2}\log(\det(\Sigma_i)) - \frac{1}{2}(\mathbf{x} - \mu_i)' \Sigma_i^{-1} (\mathbf{x} - \mu_i) \qquad (5)$$

In this equation π_i stands for the prior probability of class A_i. As before, the means and covariance matrix are substituted by their sample counterparts obtained from the training set. In the same way, π_i is substituted by the sample proportion of class A_i examples. For classification, the discriminant is calculated for each class and the one giving the higher value is chosen.

The most frequent problem with quadratic discriminants is caused when some attribute has zero variance in one class, for then the covariance matrix can not be inverted. One way of avoiding this problem is to add a small positive constant term to the diagonal terms in the covariance matrix (this corresponds to adding random noise to the attributes).

7 Logdisc

Logdisc performs a logistic discrimination. As linear discriminants, a logistic discriminant consists of an hyperplane separating the classes in the best possible way, but the criterion used to find the hyperplane is different. The method adopted in this procedure is to maximize a conditional probability. In theory, when the attributes have a normal distribution with equal covariances and are independent from each other, linear and logistic discriminants are equivalent. Different result are obtained when this hypothesis are not satisfied.

The method here described is partially parametric, as the actual pdfs for the classes are not modeled, but rather the ratio between them. In particular the logarithms of the ratios of the probability density functions for the classes are modelled as linear functions of the attributes. Thus, for two classes

$$\log \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} = \alpha + \beta' \mathbf{x} \qquad (6)$$

where α and the p-dimensional vector β are the parameters of the adopted model and must be estimated. The case of normal distribution is a special case in which these parameters are functions of the prior probabilities, of the class means and of the common covariance matrix.

The parameters are estimated by maximum conditional likelihood. The model implies that, given attribute values **x**, the conditional class probabilities for classes A_1 and A_2 take the forms:

$$P(A_1 \mid \mathbf{x}) = \frac{\exp(\alpha + \beta'\mathbf{x})}{1 + \exp(\alpha + \beta'\mathbf{x})} \qquad (7)$$

$$P(A_2 \mid \mathbf{x}) = \frac{1}{1 + \exp(\alpha + \beta'\mathbf{x})} \qquad (8)$$

Given independent samples for the two classes, the parameters are estimated by maximizing the probability:

Table 1. Average accuracies

	Discrim	Logdisc	Quadisc	Predict	C4.5 tree	C4.5 rules
CH	0,853	0,857	0,853	0,873	0,959	0,959
H1 H2	0,855	0,928	0,316	0,864	0,933	0,933

Table 2. Average false positive and false negative errors

	Discrim		Logdisc		Quadisc		Predict		C4.5 tree		C4.5 rules	
	FN	FP	FN	FP	FN	FP	FN	FP	FN	FP	FN	FP
CH	37	9	36	9	31	15	15	25	6	7	6	7
H1 H2	26	19	14	9	40	177	3	40	13	8	12	9

Table 3. Values for the *t* statistics for the CH dataset

|*t*|	Discrim	Logdisc	Quadisc	Predict	C4.5 tree	C4.5 rules
Discrim		1,000	0,000	0,452	**1,947**	**1,947**
Logdisc			0,166	0,376	**1,959**	**1,959**
Quadratic				0,615	**2,352**	**2,352**
Predict					**2,031**	**2,031**
C4.5 tree						0,000

Table 4. Values for the *t* statistics for the H1 H2 dataset

|*t*|	Discrim	Logdisc	Quadisc	Predict	C4.5 tree	C4.5 rules
Discrim		**1,753**	**2,114**	0,118	**1,586**	**1,689**
Logdisc			**2,411**	0,867	0,127	0,135
Quadisc				**2,509**	**3,068**	**3,050**
Predict					0,843	0,858
C4.5 tree						0,000

$$L(\alpha,\beta) = \prod_{\{A_1, sample\}} P(A_1 \mid \mathbf{x}) \prod_{\{A_2, sample\}} P(A_2 \mid \mathbf{x}) \qquad (9)$$

Iterative methods have been proposed in order to estimate the parameters for example by [7] and [8]. Since in practice there is often little difference between logistic and linear discriminant, the latter are taken as a starting point for the former.

8 NeuralWorks Predict

Predict by Neural Works is a system for training multi-layer neural nets. Predict use an adaptive gradient learning rule which is a form of back-propagation. Predict does not start from a fixed network architecture but uses a constructive method for determining a suitable number of hidden nodes. This constructive method is referred to as "Cascade Learning" [10] and is loosely characterised by the fact that hidden nodes are added one or a few at a time. New hidden nodes have connections from both the input buffer and the previously established hidden nodes. Construction is stopped when performance on an independent test set shows no further improvement.

9 C4.5

C4.5 [4] is a system for learning rules and decision trees. Its peculiarity is the heuristics it adopts in order to select the test to perform at each steps. These heuristics are based on the notion of entropy from information theory that represents the amount of "dis-uniformity" of examples in the training set with respect to the class attributes: at each step a test is selected that makes the resulting subsets as uniform as possible with respect to the class attribute, i.e., subsets containing examples from only one class or from a small number of classes.

10 Results

All systems have been tested on the CH and H1 H2 datasets employing 10-fold cross validation. Both datasets contain 317 tuples of which 67 belong to the Defect class and 250 to the NonDefect class. The spread of attribute values is larger for the Defect class.

Table 1 shows the average accuracies of the classification algorithms for both datasets, while table 2 shows the total number of false negative and false positive errors summed over the ten folds. False negatives are defective pieces that are classified as non defective and false positive are non defective pieces that are classified as defective. It is important to distinguish between these two types of errors because the damage that derives from a false negative is much higher than the one deriving from a false positive. Therefore, we should prefer an algorithm that minimizes the number of false negatives.

In order to evaluate if the accuracy differences between algorithms are significant, we have computed a 10-fold cross-validated paired *t* test for every pair of algorithms (see [11] for an overview of statistical tests for the comparison of machine learning algorithms).

This test is computed as follows. Given two algorithms A and B, let $p_A^{(i)}$ (respectively $p_B^{(i)}$) be the observed proportion of test examples misclassified by algorrithm A (respectively B) in trial i.

If we assume that the 10 differences $p^{(i)}=p_A^{(i)}-p_B^{(i)}$ are drawn independently from a normal distribution, then we can apply Student *t* test by computing the statistic

$$t = \frac{\overline{p} \cdot \sqrt{n}}{\sqrt{\frac{\sum_{i+1}^{n}(p^{(i)} - \overline{p})^2}{n-1}}} \qquad (10)$$

where n is the number of folds (10) and $\overline{p}$ is

$$\overline{p} = \frac{1}{n}\sum_{i=1}^{n} p^{(i)} \qquad (11)$$

In the null hypothesis, i.e. that A and B have the same accuracy, this statistic has a t distribution with n-1 (9) degrees of freedom. If we consider a probability of 90%, then the null hypothesis can be rejected if

$$|t| > t_{9,0.90} = 1.383 \qquad (12)$$

Table 3 shows the values of the statistic for the CH dataset, while table 4 shows the values of the statistic for the H1 H2 dataset. The value of the statistic for algorithms A and B can be found at the crossing of line A with column B. The numbers in bold are those that provide a probability of 90% or more of rejecting the null hypothesis.

From these tables can be seen that, for the CH dataset, the accuracy difference is statistically significant only between C4.5 algorithms and the other ones, while it is not statistically significant among the statistical and neural algorithms. Therefore, for the CH dataset, we can state that the best performance has been obtained by C4.5, both for the case of trees and rules.

On the H1 H2 dataset there is a significant difference between the best performing algorithms, C4.5 and Logdisc, and Discrim and Quadisc. The difference among the best performing algorithms and Predict is instead a little less certain, having 80% probability.

In conclusion, for both datasets, the best overall accuracy has been obtained by C4.5 both for the case of trees and rules. The comparison of machine learning and statistical techniques shows that C4.5 performs better than statistical techniques for the CH dataset, while on H1 H2 dataset Logdisc is equivalent to C4.5. Instead, for Predict, the differences with statistical techniques are less significant.

As can be seen, the CH feature is very important because it leads to more accurate classifiers for all systems apart from Logdisc and Discrim.

As regards the number of false negatives, C4.5 yields the lowest number of them for the CH dataset, while for the H1 H2 dataset the lowest number is given by Predict.

These results show that machine learning tools can outperform statistical classifiers on the domain examined.

11 RELATED WORKS

Machine learning has been widely exploited for object classification in computer vision. Learning is often essential for defining an effective classifier in the case of unstructured objects or shapes, which are difficult to model in terms of geometric, topologic or other metric features. Examples of the use of learning in computer vision are for instance recognition of hand gestures, landscape inspection, medical images analysis, and appearance-based recognition [11,12,13,14]. However, the most comprehensive work concerning the use of learning for classification is the StatLog project [5]. StatLog includes several classification algorithms, covering machine learning, neural and statistical techniques. The algorithms are compared against several different classification tasks, nine of which consists of classifying images (Dig44, KL, Vehicle, Letter, Chrom, Landsat, SatIm, Segm, Cut20, Cut50). Some of the tasks address mostly classification of pixel areas, while others address classification of derived features computed from the pixel values. The ranking of classifiers' error rates varies with the image classification task. The k-NN and Quadisc classifiers seem to achieve generally the best error rates, but with some exceptions (Vehicle and Segm for k-NN and SatIm, Segm, Cut20, Cut50 for Quadisc). The machine learning algorithm C4.5 tends to assess good performance for tasks which do not require direct classification of pixel areas, but rather of some derived features (Segm, Cut20, Cut50). In particular, C4.5 largely outperforms Quadisc on the Cut classification tasks, where the number of classes is mimimum (two), like in the defect classification task we addressed in this work.

12 CONCLUSION

We have presented an application of machine learning and statistics to the problem of recognizing surface cracks on metallic pieces. In order to learn from the images of the pieces, we have identified a set of visual primitives for characterizing each image. One of these primitives, the average gradient of luminosity, is computed on the image itself, while the others are computed on transformed versions of the image obtained with the Hough Transform (HT) and the Correlated Hough Transform (CHT). We use these primitives because they have been expressively designed for the recognition of straight lines and rectilinear shapes.

In order to test the effectiveness of these various primitives on classification, we have considered two different datasets, one containing features from the Hough and the Correlated Hough space, and another one containing features from the Hough space only.

Various machine learning and statistical techniques have been applied to the problem. As regards machine learning, we have employed an attribute value learner, C4.5, and a neural network trainer, NeuralWare Predict. As regards statistical techniques, we have employed linear, logistic and quadratic discriminant.

The results of the experiments show that, of the two feature sets, the one containing the CHT leads to more accurate classifiers for all learning methods apart from Logdisc and Quadisc, thus confirming the importance of highly specialized operators for Computer Vision.

Among all systems, C4.5 had a performance significantly higher than the other systems for the CH dataset, while for the H1 H2 dataset it was significantly higher than Discrim and Quadisc.

Even if the features were all numeric, C4.5 provided a very good performance. This is probably due to spread in the attribute values, especially for the Defect class, that requires the adaptiveness of machine learning tools.

REFERENCES

[1] J. Illingworth, J. Kittler, “A survey of the Hough transform”, Comp. Vision Graphics, Image Process. (43): 221-238.

[2] R. Cucchiara, M. Piccardi, “Eliciting visual primitives for detection of elongated shapes”, Image and Vision Computing, v. 17, n.5-6, pp. 347-355, Elsevier, 1999.

[3] R. Mason, editor, Magnetic Particle Inspection. Nondestructive Testing (33), pp. 6-12.

[4] J. R. Quinlan, C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, California, 1993.

[5] D. Michie, D.J.Spiegelhalter and C.C.Taylor (eds.), “Machine Learning, Neural and Statistical Classification”, Ellis Horwood, 1994.

[6] R. Fisher, “The use of multiple measurements in taxonomic problems”, Annals of Eugenics, 7, pp. 179-188, 1936.

[7] D. R. Cox, “Some procedures associated with the logistic qualitative response curve”, in Research Papers on Statistics: Festschrift for J. Neyman, Wiley, pp. 55-77, 1966.

[8] N. Day, D. Kerridge, “A general maximum likelyhood discriminant”, Biometrics, 23, pp. 313-324, 1967.

[9] S. E. Fahlmann, C. Lebiere, "The Cascade-Correlation Learning Architecture", Advances in Neural Information Processing Systems 2, Morgan Kaufmann, 1988.

[10] T. Dietterich, “Approximate satistical tests for comparing supervised classification learning algorithms”, Neural Computation, in press (draft version available at http://www.cs.orst.edu/~tgd/projects /supervised.html).

[11] K. Cho, S. M. Dunn, “Learning shape classes”, IEEE Trans. on PAMI 16 (1994) n. 9, pp. 882-887.

[12] B. A. Drapter, C. E. Brodley, P. E. Utgoff, “Goal directed classification using Linear Machine Decision tree”, IEEE Trans. on PAMI 16 (1994) n. 9, pp. 888-893.

[13] P. Pellegretti, F. Roli, S. Serpico, G. Vernazza, “Supervised learning of descriptions for image recognition purposes”, IEEE Trans. on PAMI 16 (1994) n. 1, pp. 92-98.

[14] H. Murase, S. K. Nayar, “Learning by a generation approach to appearance based object recognition”, Proc. of 13th ICPR, Vienna 1 (1996) , pp. 24-30.

Understanding Multi-Page Printed Documents: A Multiple Concepts Learning Problem

Floriana Esposito and **Donato Malerba** and **Francesca A. Lisi** [1]

Abstract. Document understanding denotes the recognition of semantically relevant components in the layout extracted from a document image. This recognition process is based on some visual models, whose manual specification can be a highly demanding task. In order to automatically acquire these models, the application of machine learning techniques has been proposed. In this paper, problems raised by possible dependencies between concepts to be learned are illustrated. A novel computational strategy based on the separate-and-parallel-conquer search is proposed and tested on a set or real multi-page documents processed by the system WISDOM++. Preliminary results confirm the validity of the proposed strategy and show some limits of the machine learning system used in this work.

1 INTRODUCTION

Recently, many publishing companies have started creating online bibliographic databases of their journal articles. However, a large number of publications are still available solely on paper, and document image analysis tools are essential to support data entry from printed journal and proceedings [24]. A straightforward application of OCR technology produces poor results because of the variability of the layout structure of printed documents. Form definition, a function available in many commercial OCR systems, support extraction of text from exactly defined zones, but semantically relevant document components (e.g., title and authors) are practically never printed in the very same zone of a page. An error of few millimeters in bounding the zone of interest can cause an unrecoverable loss of information. The complexity of the problem is even more evident when the data entry is extended to the whole article, which takes more than one page.

The goal of the WISDOM project undertaken at the University of Bari is to develop intelligent document processing tools that automatically transform a large variety of printed multi-page documents, especially periodicals, into a web-accessible form such as XML. This transformation requires a solution to several image processing problems, such as the separation of textual from graphical components in a document image (*document analysis*), the recognition of the document (*document classification*), the identification of semantically relevant components of the page layout (*document understanding*), the transformation of portions of the document image into sequences of characters (OCR), and the *transformation* of the page into HTML/XML format. A large amount of knowledge is required to effectively solve these problems. For instance, the segmentation of the document image can be based on the layout conventions (or *layout structure*) of specific classes of documents, while the separation of text from graphics requires knowledge on how text blocks can be discriminated from non-text blocks. In many applications presented in the literature, a great effort is made to hand-code the necessary knowledge according to some formalism, such as block grammars [18]. In the WISDOM project, we investigated the application of various inductive learning algorithms in order to solve the knowledge acquisition problem. Some of them are:

- Incremental top-down induction of decision trees. Decision trees are used to classify basic blocks extracted by the image segmentation algorithm (segmentation and block classification are two steps of the document analysis process). Incrementality satisfies an important design requirement, namely online training of the document processing system. On the other hand, it can raise space inefficiency problems [8].
- Induction of a set of first-order rules (or logical theory) from a set of training examples. Rules are used to perform a layout-based classification and understanding of segmented document images. Resorting to a first-order representation formalism is unavoidable, since each page layout consists of a variable number of spatially distributed components. Obviously, the choice of a suitable set of attributes and relations is crucial for the success of the application. Some experimental results obtained with the learning system INDUBI/CSL [16] confirmed Connel and Brady's [5] observation that both numeric and symbolic descriptions are essential to generate models of visual objects, since the former increase the sensitivity while the latter increase the stability of the internal representation of visual objects.

In this paper, a further issue is investigated in the specific context of document understanding, namely multiple dependent concept learning. Actually, learning rules for document understanding is more difficult than learning rules for document classification, since semantically relevant layout components (also called *logical components*) refer to a part of the document rather than to the whole document. Logical components may be related to each other, therefore the recognition of a logical component can be correctly performed only if its *context* is considered. Thus, it would be more appropriate to learn rules that reflect these dependencies among logical components. For instance, in the case of papers published in the IEEE Transactions on Pattern Analysis and Machine Intelligence, the following clause:
$author(X) \leftarrow ontop(Y, X), title(X)$
captures the typographical convention of printing the authors just under the title. The main benefits in learning, if possible, this kind of contextual rules are:

- *Learnability of correct concept definitions*. For instance, some learning systems that do not take concept dependencies into account, such as the well-known FOIL [21], cannot learn the defini-

[1] Dipartimento di Informatica, Università degli Studi di Bari, Via Orabona 4, I-70126 Bari, Italy, email: {esposito, malerba, lisi}@di.uniba.it

tions of ”appending two lists” and ”reversing a list” independently, since the former concept is essential to give a reasonably compact definition of the second concept.

- *Rendering explicit some concept dependencies*, which would be otherwise hidden in a set of flat, independent rules. A correct logical theory structured around a number of dependent concepts does not contain those redundancies of its equivalent theory with independent concepts; therefore it is more comprehensible and easier to be validated by experts.

In this paper, a new approach to the problem of learning multiple dependent concepts is briefly presented. This approach is that adopted by ATRE [13], a machine learning system interfaced by WISDOM++, a document processing system developed in the WISDOM project.[2]

The paper is organized as follows. Section 2 describes the document processing steps performed by WISDOM++. Section 3 show how ATRE solves some problems related to learning multiple dependent concepts. Section 4 illustrates some experimental results concerning the application of ATRE to the problem of understanding a set of real-world multi-page documents. Finally, in Section 5 our conclusions are drawn.

2 THE SYSTEM WISDOM++

A distinguishing feature of WISDOM++ is the use of a knowledge base in order to support some document processing tasks. The knowledge base is automatically built from a set of training documents using machine learning tools and techniques, which make the system highly *adaptive*. WISDOM++ has been designed as a multi-user system, in the sense that each authorized user has his/her own rule base. Currently, two categories of users are defined: *Administrators* and *end users*. Administrators can train the system to classify and understand documents, while end users can only operate by using the learned rules. Finally, WISDOM++ has been designed to manage *multi-page* documents, each of which is a *sequence* of pages. The definition of the right sequence is responsibility of the user, since the optical scan function is able to work on a single page at a time. Pages of multi-page documents are processed independently of each other in all steps.

Initially, each page is scanned with a resolution of 300 dpi and thresholded into a binary image. The bitmap of an A4-sized page takes $2,496 \times 3,500 = 1,092,000$ bytes and is stored in TIFF format. The *document analysis* process includes:

1. *Preprocessing*, that is the evaluation of the skew angle, the rotation of the document, and the computation of a *spread factor*, which is greater than 1.0 in quite simple documents with few sparse regions, while it is lower than 1.0 in complex documents with closely written text regions. Details on preprocessing algorithms can be found in [2].
2. *Segmentation*, that is the identification of rectangular *blocks* enclosing content portions. WISDOM++ segments the reduced document image into rectangular blocks by means of a variant of the Run Length Smoothing Algorithm (RLSA) [25], which operates on a document image with a lower resolution (75 dpi is considered a reasonable trade-off between the accuracy and the speed of the segmentation process). The RLSA applies four operators to the document image: 1) horizontal smoothing with a threshold C_h; 2) vertical smoothing with a threshold C_v; 3) logical AND of the two smoothed images; 4) additional horizontal smoothing with another threshold C_a. The variant implemented by WISDOM++ scans the image only twice with no additional cost [23] instead of the four times required by the original algorithm. Another novelty is that the smoothing parameters C_v and C_a are adaptively defined on the basis of the spread factor computed during the preprocessing step.
3. *Blocks classification*, which aims at discriminating blocks enclosing text from blocks enclosing graphics (pictures, drawings and horizontal/vertical lines). In WISDOM++, the classification of blocks is performed by means of a decision tree automatically built from a set of training examples (blocks) of the five classes. The choice of a ”tree-based” method instead of the most common generalized linear models is due to its inherent flexibility, since decision trees can handle complicated interactions among features and give results that can be easily interpreted [1].
4. *Layout analysis*, that is the perceptual organization process that aims at detecting structures among blocks. The result is a hierarchy of abstract representations of the document image, that is the *layout structure* of the document. The leaves of the layout tree (lowest level of the abstraction hierarchy) are blocks returned by the segmentation algorithm, while the root represents the set of pages of the whole document. A page may include several layout components, called *frames*, which are rectangular areas corresponding to groups of blocks. WISDOM++ extracts the layout structure by means of a knowledge-based, bottom-up approach: Generic knowledge on typesetting conventions is used in order to group basic blocks together [9].

While the layout structure associates the content of a document with a hierarchy of layout objects, such as blocks, frames and pages, the *logical structure* of the document associates the content with a hierarchy of *logical objects*, such as sender/receiver of a business letter, title/authors of a scientific article, and so on. The problem of finding the logical structure of a document can be cast as the problem of defining a *mapping* from the layout structure into the logical one. In WISDOM++, this mapping is limited to the association of a page with a document class (*document classification*) and the association of page layout components with basic logical components (*document understanding*). The mapping is built by *matching* the document description against both *models* of classes of documents and models of the logical components of interest for that class. Models are rules expressed in a first-order logic language, which are automatically built by applying inductive learning algorithms. A detailed description on how models for document understanding of the logical components are represented and automatically built from some training examples is explained in next section.

WISDOM++ allows the user to set up the *text extraction* process by selecting the logical components to which an OCR has to be applied. Finally the system generates an *HTML/XML version* of the original document: It contains both text returned by the OCR and pictures extracted from the original bitmap and converted into the GIF format. Text and images are spatially arranged so that the HTML/XML reconstruction of the document is as faithful as possible to the original bitmap. Moreover the XML format maintains information extracted during the document understanding phase, since the Document Type Definition (DTD) is specialized for each class of

[2] WISDOM++ is a newer version of the system WISDOM (Windows Interface System for DOcument Management), originally written in C [15][8] and part of an intelligent digital library [10]. WISDOM++ has been designed according to an object-oriented analysis and design methodology and implemented in Microsoft Visual C++. WISDOM++ can be downloaded from the following site: www.di.uniba.it/ malerba/wisdom++.

documents in order to represent the specific logical structure.

3 LEARNING MULTIPLE DEPENDENT CONCEPTS

Experimental results of a previous study on multiple dependent concept learning confirmed that by taking into account concept dependencies it is possible to improve the predictive accuracy for the document understanding problem [16]. In that study, the learning system INDUBI/CSL had been extended in order to learn multiple dependent concepts provided that the user defines a graph of possible dependencies among logical components. As planned in a previous work [1], we have replaced INDUBI/CSL with the multiple concept learning system ATRE, which is able to autonomously discover such concept dependencies. A brief description of ATRE is reported below.

3.1 The learning problem

The learning problem solved by ATRE can be formulated as follows:
Given

- a set of concepts $C_1, \ldots, C_r$ to be learned,
- a set of observations O described in a language L_O,
- a background knowledge BK described in a language L_{BK},
- a language of hypotheses L_H,
- a generalization model Γ over the space of hypotheses,
- a user's preference criterion PC,

Find
a (possibly recursive) logical theory T for the concepts $C_1, \ldots, C_r$, such that T is complete and consistent with respect to O and satisfies the preference criterion PC.

ATRE mainly differs from INDUBI/CSL for the learning goal, that is the induction of recursive logical theories, which is common in the field of inductive logic programming (ILP) [3]. Further differences concern the representation languages L_O, L_{BK} and L_H, the generalization model Γ, the interpretation of the preference criterion and the search strategy in the space of hypotheses. Each of these issues will be addressed in the following.

As to the representation languages, the basic component is the *literal* in the two distinct forms:
$f(t_1, \ldots, t_n) = Value$ (simple literal)
$f(s_1, \ldots, s_n) \in Range$ (set literal)
where f and g are function symbols called descriptors, t_i's and s_i's are terms, and $Range$ is a closed interval of possible values taken by f. Some examples of literals are the following: $color(X1) = red$, $height(X1) \in$ [1.1..1.2], and $ontop(X, Y) = true$. The last example points out the lack of predicate symbols in the representation languages adopted by ATRE. Thus, the first-order literals $p(X, Y)$ and $\neg p(X, Y)$ will be represented as $f_p(X, Y) = true$ and $f_p(X, Y) = false$, respectively, where f_p is the function symbol associated to the predicate p. Therefore, ATRE can deal with *classical negation*, $\neg$, but not with *negation by failure*, not [12], which is common to most of ILP systems. Henceforth, for the sake of simplicity, we will adopt the usual notation $p(X, Y)$ and $\neg p(X, Y)$ instead of $f_p(X, Y) = true$ and $f_p(X, Y) = false$, respectively.

The *language of observations* L_O allow a more efficient and comprehensible *object-centered representation* of observations. Indeed, observations are represented by ground multiple-head clauses [11], called *objects*, which have a conjunction of simple literals in the head. An instance of object taken from the blocks-world is the following:

$$type(blk1) = lintel \wedge type(blk2) = column \leftarrow$$
$$pos(blk1) = hor, pos(blk2) = ver, ontop(blk1, blk2)$$

which is semantically equivalent to the definite program:

$$type(blk1) = lintel \leftarrow$$
$$pos(blk1) = hor, pos(blk2) = ver, ontop(blk1, blk2)$$

$$type(blk2) = column \leftarrow$$
$$pos(blk1) = hor, pos(blk2) = ver, ontop(blk1, blk2)$$

Examples are described as pairs $< L, OID >$ where L is a literal in the head of the object pointed by the object identifier OID. Examples can be considered as *positive* or *negative*, according to the concept to be learned. For instance $< type(blk1) = lintel, O_1 >$ is a positive example of the concept $type(X) = lintel$, a negative example of the concept $type(X) = column$, and it is neither a positive nor a negative example of the concept $stable(X) = true$.

The *language of hypotheses* L_H is that of *linked, range-restricted* definite clauses [6] with simple and set literals in the body and one simple literal in the head. An example of recursive theory expressed in L_H is the following:
$even(X) \leftarrow zero(X)$
$odd(X) \leftarrow succ(Y, X), even(Y)$
$even(X) \leftarrow succ(Y, X), odd(Y)$
It states conditions for integer numbers being even or odd, given the concepts of successor and zero. Here X and Y denote variables consistently to Prolog notation. ATRE is also able to deal with numeric descriptors. More precisely, given an n-ary function symbol, $f(X_1, \ldots, X_n)$, taking on values in a numerical domain, the system produces hypotheses with set literals $f(X_1, \ldots, X_n) \in [a..b]$, where $[a..b]$ is a numerical interval computed according to the same information theoretic criterion used in INDUBI/CSL [14].

The *language of background knowledge* L_{BK} has the same constraints as the language of hypotheses. The representation languages in ATRE seem not to fit very well the ILP framework, but it is easy to transform ATRE's theories into Datalog programs [4] with built-in predicates. In general, a simple literal $f(t_1, \ldots, t_n) = Value$ can be transformed into an (n+1)-ary predicate $f(t_1, \ldots, t_n, Value)$, while a set literal $f(t_1, \ldots, t_n) \in Range$, where $Range$ is an interval $[a..b]$, can be transformed into $f(t_1, \ldots, t_n, Z), Z \geq a, Z \leq b$. The relational operators $\geq$ and $\leq$ are built-in predicates. Thanks to this transformation it is possible to extend notions and properties of standard first-order logic to ATRE definite clauses.

Regardless of the chosen representation language, a key role of the induction process is the search through a space of hypotheses. A *generalization model* Γ provides a basis for organizing this search space, since it establishes when a hypothesis covers a positive/negative example and when an inductive hypothesis is more general/specific than another. The generalization model adopted in ATRE is a variant of Plotkin's *relative generalization* [19] [20], named *generalized implication* [13].

3.2 The learning strategy

The high-level learning algorithm in ATRE belongs to the family of *sequential covering* (or *separate-and-conquer*) algorithms [17] since it is based on the strategy of learning one clause at a time (*conquer* stage), removing the covered examples (*separate* stage) and iterating the process on the remaining examples.

Many FOIL-like algorithms adopt this separate-and-conquer strategy. The most relevant novelties of the learning strategy implemented in ATRE are embedded in the design of the conquer stage. Indeed, the conquer stage of our algorithm aims at generating a clause that covers a specific positive example, the *seed*, while FOIL does not. Thus ATRE implements a general-to-specific seed-driven search strategy in the space of definite clauses.

The search space is actually a forest of as many search-trees (called *specialization hierarchies*) as the number of chosen seeds, where at least one seed per incomplete concept definition is kept (see Figure 1). Each search-tree is rooted with a unit clause and ordered by generalized implication. The forest can be processed *in parallel* by as many concurrent tasks as the number of search-trees (*parallel-conquer search*). Each task traverses the specialization hierarchies top-down (or general-to-specific), but synchronizes traversal with the other tasks at each level. Initially, some clauses at depth one in the forest are examined concurrently. Each task is actually free to adopt its own search strategy, and to decide which clauses are worth to be tested. If none of the tested clauses is consistent, clauses at depth two are considered. Search proceeds towards deeper and deeper levels of the specialization hierarchies until at least one consistent clause is found. Task synchronization is performed after that all "relevant" clauses at the same depth have been examined. A supervisor task decides whether the search should carry on or not on the basis of the results returned by the concurrent tasks. When the search is stopped, the supervisor selects the "best" consistent clause according to the user's preference criterion PC. This strategy has the advantage that simpler consistent clauses are found first, independently of the concepts to be learned. Moreover, the synchronization allows tasks to save much computational effort when the distribution of consistent clauses in the levels of the different search-trees is uneven. In Figure 1 it is shown the parallel exploration of the specialization hierarchies for the concepts of *even* and *odd* numbers.

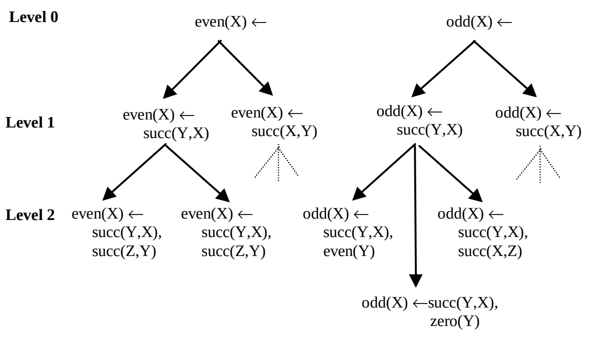

Level 0
even(X) ←
odd(X) ←
Level 1
even(X) ← succ(Y,X)
even(X) ← succ(X,Y)
odd(X) ← succ(Y,X)
odd(X) ← succ(X,Y)
Level 2
even(X) ← succ(Y,X), succ(Z,Y)
even(X) ← succ(Y,X), succ(Z,Y)
odd(X) ← succ(Y,X), even(Y)
odd(X) ← succ(Y,X), succ(X,Z)
odd(X) ←succ(Y,X), zero(Y)

Figure 1. Parallel search for the concepts *even* and *odd*

This *separate-and-parallel-conquer* search strategy provides us with a solution to the problem of *interleaving* the induction process for distinct concept definitions.

The main procedure of ATRE is shown in Figure 2. The input of the system is a set of objects, a background knowledge, a set of concepts to be learned, and a preference criterion that guides the heuristic search in the space of possible hypotheses.

The function implemented by the procedure *saturate_objects* is the saturation of a set of examples given a set of clauses. The first step towards the generation of inductive hypotheses is the saturation of all objects with respect to the given BK [22]. In this way, information that was implicit in the example, given the background knowledge, is made explicit.

Initially, all positive and negative examples (pairs $< L, OID >$) are generated for every concept to be learned, the learned theory is empty, while the set of concepts to be learned contains all C_i. The conquer stage performs a parallel general-to-specific beam search to generate a set of consistent, linked and range-restricted clauses for the concepts to be learned. A seed is associated with each specialization hierarchy. Seeds are chosen according to the textual order in which objects are provided to the system. If O_k is the first object with an untagged example of concept C_i then O_k is taken to generate seeds for C_i. In particular, all untagged examples of C_i in O_k will be selected as seeds, so it is possible to have several specialization hierarchies for each concept.

Ground literals in the body of seed objects are generalized. In particular, the generalization of a ground literal $f(t_1, \ldots, t_n) = Value$ is obtained by turning distinct constants into distinct variables, and replacing all occurrences of a constant t_i with the same variable X_i (*simple inverse substitution*). Clause specialization is performed either by adding a new generalized seed literal that preserves the property of linkedness of the clause or by restricting the interval of a set literal already in the body. When a consistent and range-restricted clause is found it is put aside: The search is stopped when at least M consistent, range-restricted clauses are determined. At this point, the best one is selected according to user's preference criterion. The default criterion is the maximization of the number of positive examples covered, and the minimization of the complexity of the clause (here represented by the number of literals in the body).

Since the addition of a consistent clause may lead to an augmented, inconsistent theory, the procedure *verify_global_consistence* applies a layering technique to recover the consistency. The selected clause is used to saturate again the object, so that recursive clauses could be generated in the next call of the procedure *parallel_conquer*. Finally, the procedure *update_examples* tags positive examples explained by the current learned theory, so that they will no longer be considered for the generation of new clauses.

4 EXPERIMENTAL RESULTS

The proposed approach to multiple concept learning has been applied to the problem of understanding multi-page printed documents.

```
procedure learn_recursive_theory(Objects, BK, {C_1,...,C_n }, PC)

SatObjects := saturate_objects(Objects, BK)
Examples := generate_pos_and_neg_examples(Objects, {C_1,...,C_n })
LearnedTheory := ∅
Concepts := {C_1,...,C_n }
repeat
  ConsistentClauses := parallel_conquer(Concepts, Examples, PC)
  Best := find_best_clause (ConsistentClauses, PC)
  ConsistentTheory:=
    verify_global_consistence(Best, LearnedTheory, Objects, Examples)
  LearnedTheory := ConsistentTheory ∪ {Clause}
  Objects := saturate_objects(SatObjects, LearnedTheory)
  Examples := update_examples(LearnedTheory,Examples)
  foreach C_i in Concepts do
    if pos_examples(C_i)= ∅ then
      Concepts := Concepts / {C_i} endif
  endforeach
until Concepts = ∅
return LearnedTheory
```

Figure 2. Main procedure of the learning algorithm implemented in ATRE

A user/trainer of WISDOM++ is asked to label some layout components of a set of training documents according to their logical meaning. Those layout components with no clear logical meaning are not labeled. Therefore, each document generates as many training examples as the number of layout components. Classes of training examples correspond to the distinct logical components to be recognized in a document. The unlabelled layout objects play the role of counterexamples for all the classes to be learned.

Figure 3. Layout of the first page of a multi-page document (left) and its partial description in a first-order logic language (right).

Each training example is represented as an *object* in ATRE, where different constants represent distinct layout components of a page layout. The description of a document page is reported in Figure 3. All descriptors used to represent a page layout of a multi-page document are listed in Table 1.

Table 1. Page layout descriptors for a multi-page document.

Descriptor name	**Definition**
$page(page)$	Nominal domain: $first, intermediate, last\_but\_one, last$
$width(block)$	Integer domain: (1..640)
$height(block)$	Integer domain: (1..875)
$x\_pos\_centre(block)$	Integer domain: (1..640)
$y\_pos\_centre(block)$	Integer domain: (1..875)
$type\_of(block)$	Nominal domain: $text, hor\_line, image, ver\_line, graphic, mixed$
$part\_of(block1, block2)$	Boolean domain: true if $block1$ contains $block2$
$on\_top(block1, block2)$	Boolean domain: true if $block1$ is above $block2$
$to\_right(block1, block2)$	Boolean domain: true if $block2$ is to the right of $block1$
$alignment(block1, block2)$	Nominal domain: $only\_left\_col, only\_right\_col, only\_middle\_col, both\_columns, only\_upper\_row, only\_lower\_row, only\_middle\_row, both\_rows$

The following clauses are used as background knowledge, in order to automatically associate information on page order to layout blocks.

$at\_page(X) = first \leftarrow part\_of(Y, X), page(Y) = first$
$at\_page(X) = intermediate \leftarrow$
  $part\_of(Y, X), page(Y) = intermediate$
$at\_page(X) = last\_but\_one \leftarrow$
  $part\_of(Y, X), page(Y) = last\_but\_one$
$at\_page(X) = last \leftarrow part\_of(Y, X), page(Y) = last$

Three long papers appeared in the January 1996 issue of the IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) have been considered. The papers contain thirty-seven pages, each of which has a variable number of layout components (about ten on average). Layout components can be associated with at most one of the following eleven logical labels: $abstract$, $affiliation$, $author$, $biography$, $caption$, $figure$, $index\_term$, $page\_number$, $references$, $running\_head$, $title$.

Learning rules for document understanding raises issues concerning the induction of recursive theories. Simple and mutual concept dependencies have to be handled, since the logical components refer to a part of the document rather than to the whole document and may be related to each other. For instance, in case of papers published in journals, the following dependent clauses:

$running\_head(X) \leftarrow$
  $top\_left(X), text(X), even\_page\_number(X)$
$running\_head(X) \leftarrow$
  $top\_right(X), text(X), odd\_page\_number(X)$
$paragraph(Y) \leftarrow ontop(X, Y), running\_head(X), text(Y)$

express the fact that a textual layout component at the top left (right) hand corner of an even (odd) page is a running head, while a textual layout component below a running-head is a paragraph of the paper. Moreover, the recursive clause
$paragraph(Y) \leftarrow ontop(X, Y), paragraph(X), text(Y)$
is useful to classify all textual layout components below the uppermost paragraph. Therefore, document understanding seems to be the kind of application that may benefit of learning strategies for multiple predicate learning. By running ATRE on the training set described above, the following theory is returned:

1. $logic\_type(X) = running\_head \leftarrow$
   $y\_pos\_centre(X) \in [18..39], width(X) \in [77..544]$
2. $logic\_type(X) = page\_number \leftarrow$
   $width(X) \in [2..8], y\_pos\_centre(X) \in [19..40]$
3. $logic\_type(X) = figure \leftarrow$
   $type\_of(X) = image, at\_page(X) = intermediate$
4. $logic\_type(X) = figure \leftarrow type\_of(X) = graphic$
5. $logic\_type(X) = abstract \leftarrow$
   $at\_page(X) = first, width(X) \in [487..488]$
6. $logic\_type(X) = affiliation \leftarrow$
   $at\_page(X) = first, y\_pos\_centre(X) \in [720..745]$
7. $logic\_type(X) = caption \leftarrow$
   $height(X) \in [9..75], alignment(Y, X) = only\_middle\_col,$
   $logic\_type(Y) = figure, type\_of(X) = text$
8. $logic\_type(X) = author \leftarrow$
   $at\_page(X) = first, y\_pos\_centre(X) \in [128..158]$
9. $logic\_type(X) = references \leftarrow$
   $height(X) \in [332..355], x\_pos\_centre(X) \in [153..435]$
10. $logic\_type(X) = title \leftarrow$
   $at\_page(X) = first, height(X) \in [18..53]$
11. $logic\_type(X) = biography \leftarrow$
   $at\_page(X) = last, height(X) \in [65..234]$
12. $logic\_type(X) = caption \leftarrow$
   $height(X) \in [9..75], on\_top(Y, X), logic\_type(Y) = figure,$

$type\_of(X) = text, to\_right(Z, Y)$

13. $logic\_type(X) = index\_term \leftarrow$
    $height(X) \in [8..8], y\_pos\_centre(X) \in [263..295]$
14. $logic\_type(X) = caption \leftarrow$
    $alignment(X, Y) = only\_lower\_row, height(X) \in [9..9]$
15. $logic\_type(X) = caption \leftarrow$
    $on\_top(Y, X), logic\_type(Y) = figure,$
    $type\_of(X) = text, alignment(Y, Z) = only\_right\_col$
16. $logic\_type(X) = caption \leftarrow$
    $height(X) \in [9..75], on\_top(X, Y),$
    $logic\_type(Y) = figure, type\_of(X) = text,$
    $type\_of(Y) = graphic$
17. $logic\_type(X) = caption \leftarrow$
    $height(X) \in [9..75], alignment(Y, X) = only\_left\_col,$
    $alignment(Z, Y) = only\_left\_col, logic\_type(Z) = caption,$
    $width(Z) \in [467..546]$

Clauses are reported in the order in which they are learned. The theory contains some concept dependencies (see clauses 7 and 12) as well as some kind of recursion (see clause 17). Surprisingly, some expected concept dependencies were not discovered by the system, such as that relating the running head to the page number:

$$\begin{aligned}&logic\_type(X) = page\_number \leftarrow\\&\quad to\_right(X, Y), logic\_type(Y) = running\_head\\&logic\_type(X) = page\_number \leftarrow\\&\quad to\_right(Y, X), logic\_type(Y) = running\_head\end{aligned}$$

The reason is due to the semantics of the descriptor to_right, which is generated by WISDOM++ only when two layout components are at a maximum distance of 100 points, which is not the case of articles published on the PAMI transactions. Same consideration applies to other possible concept dependencies (e.g., title-authors-abstract).

In order to test the predictive accuracy of the learned theory, we considered the fourth long article published in the same issue of the transactions used for training. WISDOM++ segmented the fourteen pages of the article into 169 layout components, sixteen of which (i.e., less than 10%) could not be properly labeled using by the learned theory (omission errors). No commission error was observed. This is important in this application domain, since commission errors can lead to totally erroneous storing of information. Finally, it is important to observe that many omission errors are due to near misses. For instance, the running head of the first page is not recognized simply because its centroid is located at point 40 along the vertical axis, while the range of y_pos_center values determined by ATRE in the training phase is $[18..39]$ (see clause 1). Significant recovery of omission errors can be obtained by relaxing the definition of flexible matching between definite clauses [7].

5 CONCLUSIONS

This paper illustrates the problem of document image understanding, which is just one of the problems met in paper document processing. To carry out the document understanding task, it is necessary to establish models, that is general descriptions of each logical component to be recognized. These descriptions are expressed in a first-order logic formalism, such that layout components correspond to variables, properties are expressed by means of either unary predicates or function symbols, while spatial relations among layout components are represented by either predicates or function symbols of arity $n < 1$.

Hand-coding models for document understanding has been the usual approach followed in many applications. Since this is a demanding task, we explored the possibility of automatically acquiring them by means of machine learning techniques. Models can be induced from a set of training documents for which the exact correspondence of layout components to logical labels is known a priori. The main issue in learning models for document understanding is concept dependence: mutual relations often occur between logical components and it would be sensible to learn rules that express such relations. Discovering concept dependencies is not easy so that in this work we have presented a solution based on a separate-and-parallel-conquer search strategy. The proposed strategy has been implemented in ATRE, a learning system that induces logical theories used by the document processing system WISDOM++ when the document understanding task is carried out.

The problem of learning multiple dependent concepts is not specific of the application to document understanding. It occurs every time a domain-specific knowledge-base used to solve the more general class of scene labeling problems is automatically built from a set of training examples (labeled scenes). As future work we plan to investigate the empirical and analytical effects of the computational strategy presented in this paper to other labeling problems.

REFERENCES

[1] O. Altamura, F. Esposito, F.A. Lisi, and D. Malerba, ‘Symbolic learning techniques in paper document processing’, in *Machine Learning and Data Mining in Pattern Recognition*, eds., P. Perner and M. Petrou, volume 1715 of *Lecture Notes in Artificial Intelligence*, Berlin, (1999). Springer-Verlag.

[2] O. Altamura, F. Esposito, and D. Malerba, ‘WISDOM++: An interactive and adaptive document analysis system’, in *Proceedings of the Fifth International Conference on Document Analysis and Recognition*, pp. 366–369, Los Vaqueros: CA, (1999). IEEE Computer Society Press.

[3] F. Bergadano and D. Gunetti, *Inductive logic programming: from machine learning to software engineering*, MIT Press, Cambridge: MA, 1997.

[4] S. Ceri, G. Gottlob, and L. Tanca, ‘What you always wanted to know about datalog (and never dared to ask)’, *IEEE Transactions on Knowledge and Data Engineering*, **1**, 146–166, (1989).

[5] J.H. Connell and M. Brady, ‘Generating and generalizing models of visual objects’, *Artificial Intelligence*, **31**, 159–183, (1987).

[6] L. De Raedt, *Interactive Theory Revision*, Academic Press, London, 1992.

[7] F. Esposito, S. Caggese, D. Malerba, and G. Semeraro, ‘Classification in noisy domains by flexible matching’, in *Proceedings of the European Symposium on Intelligent Techniques*, pp. 45–49, (1997).

[8] F. Esposito, D. Malerba, and F.A. Lisi, ‘Machine learning for intelligent document processing: The WISDOM system’, in *Foundations of Intelligent Systems*, eds., Z.W. Ras and A. Skowron, volume 1609 of *Lecture Notes in Artificial Intelligence*, Berlin, (1999). Springer-Verlag.

[9] F. Esposito, D. Malerba, and G. Semeraro, ‘A knowledge-based approach to the layout analysis’, in *Proceedings of the Third International Conference on Document Analysis and Recognition*, pp. 466–471, Los Alamitos: CA, (1995). IEEE Computer Society.

[10] F. Esposito, D. Malerba, G. Semeraro, N. Fanizzi, and S. Ferilli, ‘Adding machine learning and knowledge intensive techniques to a digital library service’, *International Journal of Digital Libraries*, **2**, 3–19, (1998).

[11] G. Levi and F. Sirovich, ‘Generalized and-or graphs’, *Artificial Intelligence*, **7**, 243–259, (1976).

[12] J.W. Lloyd, *Foundations of Logic Programming*, Springer-Verlag, Berlin, 2nd edn., 1987.

[13] D. Malerba, F. Esposito, and F.A. Lisi, ‘Learning recursive theories with ATRE’, in *Proceedings of the Thirteenth European Conference on Artificial Intelligence*, ed., H. Prade, pp. 435–439, Chichester: UK, (1998). John Wiley & Sons.

[14] D. Malerba, F. Esposito, G. Semeraro, and S. Caggese, ‘Handling continuous data in top-down induction of first-order rules’, in *AI*IA 97:*

Advances in Artificial Intelligence, ed., M. Lenzerini, volume 1321 of *Lecture Notes in Artificial Intelligence*, Berlin, (1997). Springer-Verlag.

[15] D. Malerba, F. Esposito, G. Semeraro, and L. De Filippis, ‘Processing paper documents with WISDOM’, in *AI*IA 97: Advances in Artificial Intelligence*, ed., M. Lenzerini, volume 1321 of *Lecture Notes in Artificial Intelligence*, Berlin, (1997). Springer-Verlag.

[16] D. Malerba, G. Semeraro, and F. Esposito, *A multistrategy approach to learning multiple dependent concepts*, Wiley, New York, 1997.

[17] T.M. Mitchell, *Machine Learning*, McGraw-Hill, New York, 3rd edn., 1997.

[18] G. Nagy, S. Seth, and M. Viswanathan, ‘A prototype document image analysis system for technical journals’, *IEEE Computer*, **25**, 10–22, (1992).

[19] G.D. Plotkin, *Automatic methods of inductive inference*, Edinburgh University, 1971. PhD thesis.

[20] G.D. Plotkin, ‘A further note on inductive generalization’, *Machine Intelligence*, **6**, 101–124, (1971).

[21] J.R. Quinlan, ‘Learning logical definitions from relations’, *Machine Learning*, **5**, 239–266, (1990).

[22] C. Rouveirol, ‘Flattening and saturation: Two representation changes for generalization’, *Machine Learning*, **14**, 219–232, (1994).

[23] F.Y. Shih and S.-S. Chen, ‘Adaptive document block segmentation and classification’, *IEEE Trans. on Systems, Man, and Cybernetics - Part B*, **26**, 797–802, (1996).

[24] G.R. Thoma, ‘Automating data entry for an online biomedical database: a document image analysis application’, in *Proceedings of the Fifth International Conference on Document Analysis and Recognition*, pp. 370–373, Los Vaqueros: CA, (1999). IEEE Computer Society Press.

[25] K.Y. Wong, R.G. Casey, and F.M. Wahl, ‘Document analysis system’, *IBM Journal of Research Development*, **26**, 647–656, (1982).

Improving the Accuracy of C4.5 by Feature Pre-Selection

Petra Perner [1] and **Chid Apte** [2]

Abstract.
Selecting the right set of features for classification is one of the most important problems in designing a good classifier. Decision tree induction algorithms such as C4.5 have incorporated in their learning phase an automatic feature selection strategy while some other statistical classification algorithm require the feature subset to be selected in a preprocessing phase. It is well know that correlated and irrelevant features may degrade the performance of the C4.5 algorithm. In our study, we evaluated the influence of feature pre-selection on the prediction accuracy of C4.5 using a real-world data set. We observed that the accuracy of the C4.5 classifier can be improved with an appropriate feature pre-selection phase for the learning algorithm.

1 Introduction

Selecting the right set of features for classification is one of the most important problems in designing a good classifier. Very often we don't know a-priori what the relevant features are for a particular classification task. One popular approach to address this issue is to collect as many features as we can prior to the learning and data modeling phase. However, irrelevant or correlated features, if present, may degrade the performance of the classifier. In addition, large feature spaces can sometimes result in overly complex classification models that may not be easy to interpret.

In the emerging area of data mining applications, users of data mining tools are faced with the problem of data sets that are comprised of large numbers of features and instances. Such kinds of data sets are not easy to handle for mining. The mining process can be made easier to perform by focussing on a subset of relevant features while ignoring the other ones. In the feature subset selection problem, a learning algorithm is faced with the problem of selecting some subset of features upon which to focus its attention.

In this paper, we present our study on features subset selection and classification with C4.5 algorithm. In Section 2, we briefly describe the criteria used for feature selection and the feature selection methods. Although, C4.5 has a feature selection strategy included in its learning performance it has been observed that this strategy is not optimal. Correlated and irrelevant attributes may degrade the performance of the induced classifier. Therefore, we use feature subset selection prior to the learning phase. The CM algorithm selects features based upon their rank ordered *contextual merits* [4]. The feature selection strategy used by C4.5 and the CM algorithm are reviewed in Section 2. For our experiments, we used a real data set that includes features extracted from x-ray images which describe defects in a welding seam. It is usually unclear in these applications what the right features are. Therefore, most analyses begin with as many features as one can extract from the images. This process as well as the images are described in Section 3.

In Section 4, we describe our results. We show that the prediction accuracy of the C4.5 classifier will improve when provided with a pre-selected feature subset. The results show that the feature subsets created by CM algorithm and the feature subset normally extracted by C4.5 have many features in common. However, the C4.5 selects some features that are never selected by the CM algorithm. We hypothesize that irrelevant features are weeded out by the CM feature selection algorithm while they get selected by the C4.5 algorithm. A comparison of the feature ranking done by the CM algorithm with the ranking of the features done by C4.5 for the first 10 features used by C4.5 shows that there is a big difference. Finally, our experiments also indicate that model complexity does not significantly change for the better or worse when pre-selecting features with CM.

2 Feature Subset Selection Algorithms

According to the quality criteria [8] for feature selection, the model for feature selection can be distinguished into the filter model and the wrapper model [1, 7]. The wrapper model attempts to identify the best feature subset for use with a particular algorithm, while the filter approach attempts to assess the merits of features from the data alone. Although the wrapper model can potentially produce the best resulting classifier, it does so by doing an exhaustive search over the entire feature space. Various search strategies have been developed in order to reduce the computation time [9] for wrapper algorithms. The filter approach on the other hand is a greedy search based approach that is computationally not as expensive. The feature selection in C4.5 may be viewed as a filter approach, while the CM algorithm may be viewed as a wrapper approach.

[1] Institute of Computer Vision and Applied Computer Sciences, Arno-Nitzsche-Str. 45, 04277 Leipzig, Germany. Ibaiperner@aol.com
[2] IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA. apte@us.ibm.com

2.1 Feature Selection done by Decision Tree Induction

Determining the relative importance of a feature is one of the basic tasks during decision tree generation. The most often used criteria for feature selection is information theoretic based, such as the Shannon entropy measure I for a data set. If we subdivide a data set using values of an attribute as separators, we obtain a number of subsets. For each of these subsets we can compute the information value. If the the information value of a subset n is i_n, then the new information value is given by $I_i = \sum q_n i_n$, where q_n is the subset of data points with attribute value n. I_i will be smaller than I, and the difference $(I - I_i)$ is a measure of how well the attribute has discriminated between different classes. The attribute that maximizes this difference is selected.

The measure can also be viewed as a class separability measure. The main drawback of the entropy measure is its sensitivity to the number of attributes values [11]. Therefore C4.5 uses the gain ratio. However, this measure suffers the drawback that it may choose attributes with very low information content of the attribute itself [2].

C4.5 [10] uses a univariate feature selection strategy. At each level of the tree building process only one attribute, the attribute with the highest values for the selection criteria, is picked out of the set of all attributes. Afterwards the sample set is split into sub-sample sets according to the values of this attribute and the whole procedure is recursively repeated until only samples from one class are in the remaining sample set or until the remaining sample set has no discrimination power anymore and the tree building process stops.

As we can see feature selection is only done at the root node over the entire decision space. After this level, the sample set is split into sub-samples and only the most important feature in the remaining sub-sample set is selected. Geometrically it means, the search for good features is only done in orthogonal decision subspaces ,which might not represent the real distributions, beginning after the root node. Thus, unlike statistical feature search strategies [3] this approach is not driven by the evaluation measure for the combinatorial feature subset; it is only driven by the best single feature. This might not lead to an optimal feature subset in terms of classification accuracy.

Decision trees users and researchers have recognized the impact of applying a full set of features to a decision tree building process versus applying only a judiciously chosen subset. It is often the case that the latter produces decision trees with lower classification errors, particularly when the subset has been chosen by a domain expert. Our experiments were intended to evaluate the effect of using multivariate feature selection methods as pre-selection steps to a decision tree building process.

2.2 Contextual Merit Algorithm

For our experiment, we used the contextual merit (CM) algorithm [4]. This algorithm employs a merit function based upon weighted distances between examples which takes into account complete feature correlations to the instance class. The motivation underlying this approach was to weight features based upon how well they discriminate instances that are close to each other in the Euclidean space and yet belong to different classes. By focusing upon these nearest instances, the context of other attributes is automatically taken into account.

To compute contextual merit, the distance d^k_{rs} between values z_{kr} and z_{ks} taken by feature k for examples r and s is used as a basis. For symbolic features, the inter-example distance is 0 if $z_{kr} = z_{ks}$, and 1 otherwise. For numerical features, the inter-example distance is $\min\left(\frac{z_{kr}-z_{ks}}{t_k}, 1\right)$, where t_k is a threshold for feature k (usually 1/2 of the magnitude of range of the feature). The total distance between examples r and s is $D_{rs} = \sum_{k=1}^{N_f} d^k_{rs}$, and the contextual merit for a feature f is $M_f = \sum_{r=1}^{N} \sum w^f_{rs} d^f_{rs}$, where N is the total number of examples, C_r is the set of examples not in the same class as examples r, and w^f_{rs} is a weight function chosen so that examples that are close together are given greater influence in determining each features merit. In practice, it has been observed that $\frac{1}{D^2_{rs}}$ if s is one of k nearest neighbors to r, and 0 otherwise, provides robust behavior as a weight function. Additionally, using $\ln \#\bar{C}(r)$ as the value for k has also exhibited robust behavior. This approach to computing and ordering features by their merits has been observed to be very robust, across a wide range of examples.

3 Our Data Set

A detailed description of the data set can be found in [6]. Here we try to briefly sketch out how the data set was created and what kind of features were used.

The subject of this investigation is the in-service inspection of welds in pipes of austenitic steel. The flaws to be looked for in the austenitic welds are longitudinal cracks due to intergranular stress corrosion cracking starting from the inner side of the tube.

The radio-graphs are digitized with a spatial resolution of 70 mm and a gray level resolution of 16 bit per pixel. Afterwards they are stored and decomposed into various Regions of Interest (ROI) of 50 x 50 pixel size. The essential information in the ROIs is described by a set of features which are calculated from various image-processing methods.

Images of flaws in welds are radio-graphed by local grey level discontinuities. Subsequently, the morphological edge finding operator, the derivative of Gaussian operator and the Gaussian weighted image moment vector operators are used for feature extraction.

The morphological edge detection operator consists of a combination of morphological operators (i.e. dilation and erosion) which move the gray value edges in an image in different directions. The difference in images $dilation(g(p)) - g(p)$ and $g(p) - erosion(g(p))$ result in respectively shifted edge-images (where $g(p)$ is the original image). After a final minimum operation on both images, the steepest edges remain in the resulting image as a maximum.

The derivative of Gaussian filter is based on a combination of a Gaussian smoothing followed by a partial derivation of the image in the $x-$ and $y-$ directions. The result of the filter is chosen as the maximum of the derivatives.

Another filter is designed specially for flaw detection in radio-graphs of welds. This method uses the vector representation of the image and calculates the image moment in an analogous fashion to the model known from mechanics.

A one-dimensional FFT-filter for crack detection problem is also employed. This filter is based on the assumption that the preferential direction of the crack is positioned in the image in the horizontal direction. The second assumption is based upon the empirical observation that the half power width of a crack indication is smaller than $300mm$. The filter consists of a column wise FFT high-pass Bessel operation that works with a cutoff frequency of $2LP/mm$. Normally the half-power width of under-cuts is greater so that this filter suppresses them. This means that it is possible to distinguish between under-cuts and cracks with this FFT-filter. A row-oriented low-pass that is applied to the output of this filter helps to eliminate noise and to point out the cracks more clearly.

Furthermore, a Wavelet filter is also used. The scale representation of the image after the Wavelet transform makes it possible to suppress the noise in the image with a simple threshold operation without losing significant parts of the content of the image. The noise in the image is an interference of film and scanner noise and irregularities caused by the material of the weld.

The features which describe the content of the ROI are extracted from profile plots which run through the ROI perpendicular to the weld. In a single profile plot, the position of a local minimum is detected which is surrounded by two maxima that are as large as possible. This definition varies a little depending on the respective image processing routine. A template which is adapted to the current profile of the signal allows us to calculate various features. Additionally, the half-power width and the respective gradients between the local extrema are calculated. To avoid statistical calculation errors, the calculation of the template features is averaged over all of the columns along an ROI.

The methods outlined here lead to 36 parameters being collected for every ROI. The data set used in this experiment contains features for ROIs from background, crack and undercut regions. The data set consists of altogether 1924 ROIs with 1024 extracted from regions of no disturbance, 465 from regions with cracks and 435 from regions with under-cuts.

4 Results

Table 1 illustrates the error rate for the C4.5 classifier when using all features as well as error rates for different feature subsets. The error rate was estimated using cross-validation. The improvement in accuracy is two percent for the pruned case. To interpret this improvement, we use a classification analysis conducted earlier [5], where performance actually peaked and then deteriorated as the number of features was increased. We observe similar behavior in our experiments. Classification error is at its minimum when the feature subset size is 20. This is in contrast to the feature subset size of 28 that C4.5 selects when presented with the entire feature set, with no pre-selection.

It may be argued that it is not worth doing feature subset selection before tree induction since the improvement in prediction accuracy is not so dramatic. However, the importance of an improvement, however small, clearly depends on the requirements of the application for which the classifier is being trained. We further observed (Table 4) that about 67% of the total features are used similarly by CM and C4.5, while about 33% of the features are exclusively selected by CM, and 16% are exclusively selected by C4.5.

Parameters	Test=Design		Crossvalidation	
	Unpruned	Pruned	Unpruned	Pruned
All	0.9356	1.6112	24.961	24.545
10	1.5073	3.7942	29.4332	28.7051
15	1.4033	3.0146	26.365	26.4171
20	1.5073	2.5988	23.7649	22.7769
24	0.9356	1.7152	24.493	23.5049
28	0.9875	1.7152	25.117	24.077

Table 1. Error Rate for Different Feature Subsets

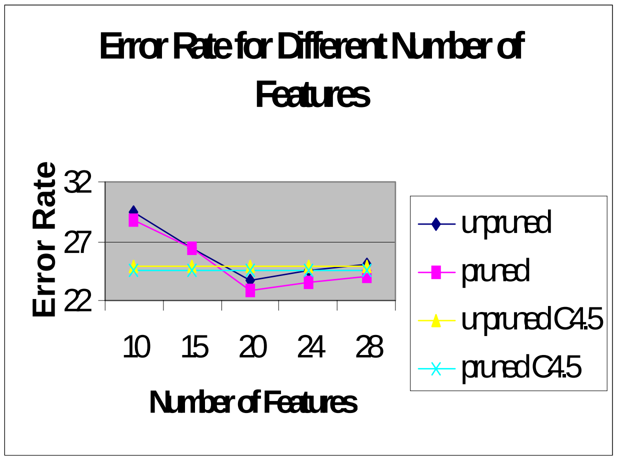


Table 2. Error Rates for Different Sizes Feature Sets

Table 3 shows that the tree does not necessarily become more compact even if a reduced set of features is used. The tree actually becomes even larger in the case with the best error rate. We therefore cannot draw any useful conclusion about feature set size and its relation to model complexity.

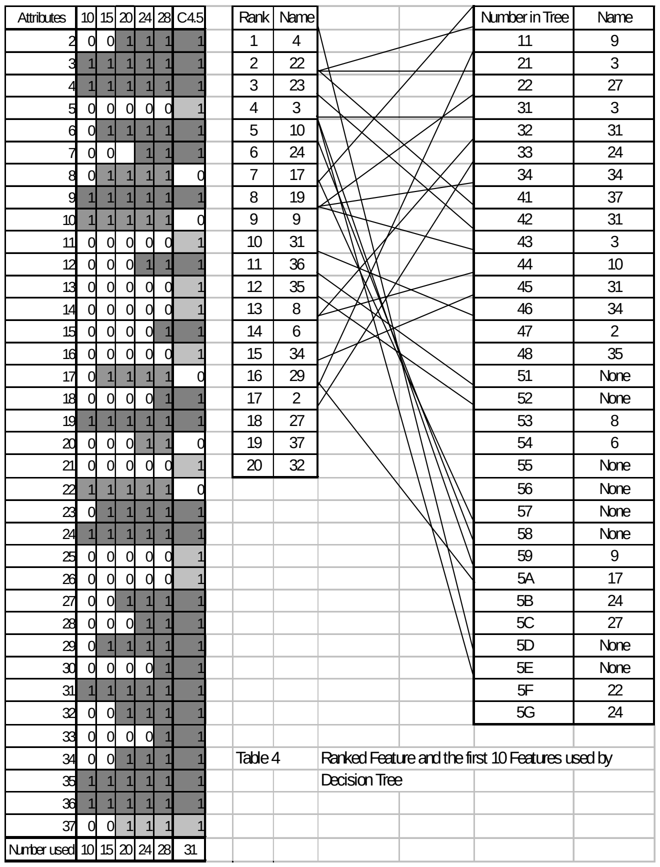
Number of Features	10	15	20	24	28	37
Nodes	236	204	178	166	164	161
Edges	237	206	176	137	161	159

Table 3. Number of Nodes and Edges

We also observe (Table 4) that in comparing the two trees generated by C4.5 with and without CM's pre-selection, the feature used for splitting at the root node changes.

5 Conclusion

We have studied the influence of feature subset selection based on a filter and wrapper approach to the performance of C4.5. Our experiment was motivated by the fact that C4.5 uses a non-optimal feature search strategy. We used the CM algorithm for feature subset selection which measures importance of a feature based on a contextual merit function. Our results



Attributes	10	15	20	24	28	C4.5
2	0	0	1	1	1	1
3	1	1	1	1	1	1
4	1	1	1	1	1	1
5	0	0	0	0	0	1
6	0	1	1	1	1	1
7	0	0		1	1	1
8	0	1	1	1	1	0
9	1	1	1	1	1	1
10	1	1	1	1	1	0
11	0	0	0	0	0	1
12	0	0	0	1	1	1
13	0	0	0	0	0	1
14	0	0	0	0	0	1
15	0	0	0	0	1	1
16	0	0	0	0	0	1
17	0	1	1	1	1	0
18	0	0	0	0	1	1
19	1	1	1	1	1	1
20	0	0	0	1	1	0
21	0	0	0	0	0	1
22	1	1	1	1	1	0
23	0	1	1	1	1	1
24	1	1	1	1	1	1
25	0	0	0	0	0	1
26	0	0	0	0	0	1
27	0	0	1	1	1	1
28	0	0	0	1	1	1
29	0	1	1	1	1	1
30	0	0	0	0	1	1
31	1	1	1	1	1	1
32	0	0	1	1	1	1
33	0	0	0	0	1	1
34	0	0	1	1	1	1
35	1	1	1	1	1	1
36	1	1	1	1	1	1
37	0	0	1	1	1	1
Number used	10	15	20	24	28	31

Rank	Name
1	4
2	22
3	23
4	3
5	10
6	24
7	17
8	19
9	9
10	31
11	36
12	35
13	8
14	6
15	34
16	29
17	2
18	27
19	37
20	32

Number in Tree	Name
11	9
21	3
22	27
31	3
32	31
33	24
34	34
41	37
42	31
43	3
44	10
45	31
46	34
47	2
48	35
51	None
52	None
53	8
54	6
55	None
56	None
57	None
58	None
59	9
5A	17
5B	24
5C	27
5D	None
5E	None
5F	22
5G	24

Table 4 Ranked Feature and the first 10 Features used by Decision Tree

show that feature subset selection can help to improve the prediction accuracy of the induced classifier. However, it may not lead to more compact trees and the prediction accuracy may not increase dramatically.

The main advantage may be that fewer features required for classification can be important for applications such as image interpretation where computational costs for extracting the features may be high and require special purpose hardware. For such domains, feature pre-selection to prune down the feature set size may be a beneficial analysis phase.

ACKNOWLEDGEMENTS

We would like to thank the referees for their comments which helped improve this paper.

REFERENCES

[1] T.M Cover, 'On the possible ordering in the measurement selection problem', *IEEE Transactions*, **SMC-7**(9), 657–661, (1977).

[2] R. Lopez de Mantaras, 'A distance-based attribute selection measure for decision tree induction', *Machine Learning*, **6**(1991), 81–92.

[3] Keinosuke Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, 1990.

[4] Se June Hong, 'Use of contextual information for feature ranking and discretization', *IEEE Trans. on Knowledge Discovery and Data Engineering*, (1996).

[5] G. F. Hughes, 'On the mean accuarcy of statistical pattern recognizers', *IEEE Transactions*, **IT-14**(1), 55–63, (1968).

[6] C. Jacobsen, U. Zscherpel, and P. Perner, *A Comparision between Neural Networks and Decision Trees*, 144–158, Machine Learning and Data Mining in Pattern Recognition, Springer Verlag, 1999.

[7] R. Kohavi and G.H. John, *The Wrapper Approach*, 33–50, Feature Extraction Construction and Selection, Kluwer Academic Publishers, 1998.

[8] M. Nadler and Eric P. Smith, *Pattern Recognition Engineering*, John Wiley&Sons Inc., 1993.

[9] P. Pudil, J. Navovicova, and J. Kittler, 'Floating search methods in feature selection', *Pattern Recognition Letters*, **15**(1994), 1119–1125.

[10] J.R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.

[11] A.P. White and W.Z. Lui, 'Bias in the information-based measure in decision tree induction', *Machine Learning*, **15**(1994), 321–329.