

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/221367650>

A Combination of Machine Learning and Image Processing Technologies for the Classification of Image Regions

Conference Paper · September 2003

DOI: 10.1007/978-3-540-25981-7_13 · Source: DBLP

CITATIONS
4

READS
371

3 authors:



Andreas D. Lattner
Otto Group
68 PUBLICATIONS 341 CITATIONS
[SEE PROFILE](#)





Andrea Miene
Universität Bremen
55 PUBLICATIONS 299 CITATIONS
[SEE PROFILE](#)



Otthein Herzog
Tongji University, Shanghai, Jacobs University and Bremen Univers...
272 PUBLICATIONS 1,615 CITATIONS
[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:

 Logistics Process Optimization [View project](#)

 Theoretical Foundations driving Urban Development [View project](#)

A Combination of Machine Learning and Image Processing Technologies for the Classification of Image Regions

Andreas D. Lattner, Andrea Miene, and Otthein Herzog

Center for Computing Technologies - TZI, University of Bremen,
PO Box 330 440, D-28334 Bremen, Germany
{adl|andrea|herzog}@tzi.de

Abstract If large amounts of images are available, as it is the case in image archives, image retrieval technologies are necessary to help the user to find needed information. In order to make such queries possible, images have to be enriched with content-based annotations. As manual annotation is very costly, system support is desired. This paper introduces an approach to learning image region classifiers from extracted color, texture, shape, and position features. In different experiments, three machine learning algorithms were applied for the classification of character shapes and regions in landscape images.

1 Introduction

Advances in information technologies during the last decades have made the storage of large amounts of data possible. Besides text this also includes images and videos. If many images have to be handled, as it is the case in videos, image archives or satellite images, image retrieval technologies have to be set up to help the user to find relevant information. In order to make such queries possible, images have to be enriched with content-based annotations. As manual annotation is very costly, system support is desired. In order to automate the annotation process, it is necessary to identify objects in images, to extract features, and to create classifiers, which later can be used for automated classification of objects.

In this work, an approach to applying machine learning (ML) technologies for the automated creation of classification rules for image regions is introduced. The next section gives an overview of which features were used as input for the ML algorithms and how they can be determined by image processing techniques. The following section shows our approach to learning classifiers and to applying them to unknown image regions. This section also presents an approach to enabling adaptivity by establishing multiple classification schemes. Section 4 presents experiments on two different test sets: character and landscape images. Following the section about related work, this paper ends with a conclusion.

Table 1. Color, texture, position, and shape features

Color Features	Texture Features	Shape Features
Mean of hue / lightness / saturation, standard deviation of hue / lightness / saturation, length of hue / lightness / saturation interval, minimal lightness / saturation value, maximal lightness / saturation value, color identifier	Shape of primitives, linelikeness, coarseness, regularity, directionality, contrast, softness	Number of lakes / vertices / edges / shifts in direction / bays / main lines, circularity, convexity, lake factor, circularity of convex hull, direction, eccentricity, horizontal / vertical position, relative size, horizontal / vertical balance, cog shift, main bay direction

2 Color, Texture, Position, and Shape Features

This section presents the image processing technologies which have been applied for extracting features from images. Some algorithms for image segmentation and feature extraction have been taken from the *PictureFinder* (*PF*) system [10]. Additionally, feature extraction algorithms for shape features and for statistical color features have been developed. An overview of the extracted features is shown in Table 1.

2.1 Image Analysis with the *PictureFinder* System

PictureFinder is an image retrieval system for analyzing and querying images with certain properties in image archives. During the analysis phase, an automated segmentation and annotation of color and texture features is performed. In order to find images with certain properties in the image archive, object recognition rules can be defined manually. The system also allows for annotating keywords to images. It is possible to search for images that contain objects with some color or texture features at different areas of the image. An overview of the *PF* system can be found in [10].

The color analysis of the *PF* system performs a color segmentation of the image. The segmentation algorithm is applied to an image representation in the HLS space (hue, lightness, saturation). For that reason, a transformation of RGB images has to be carried out.

The color segmentation groups pixels of similar colors together to regions. The assignment is determined by comparing the difference of hue, lightness, and saturation of two neighboring pixels. If the thresholds are exceeded, a new region is created. The color segmentation is performed by an extended blob-coloring algorithm [1] for color images. The result of the color analysis are the different color regions, represented by their bounding box, their centers of gravity, and their colors („blue“, „purpleblue“, „purple“, „redpurple“, „red“, „orangered“, „orange“, „yelloworange“, „yellow“, „greenyellow“, „green“, „bluegreen“, „black“, „darkgrey“, „grey“, „lightgrey“, „white“). This information about regions might be sufficient for image retrieval tasks in image databases. For the creation of classification rules, in this work additional color statistic features are provided:

- hue_{mean} , $lightness_{mean}$, $saturation_{mean}$: Mean of the hue / lightness / saturation value of all pixels of this region.
- hue_{stddev} , $lightness_{stddev}$, $saturation_{stddev}$: Standard deviation of the hue / lightness / saturation value of all pixels of this region.

- $hue_{range}, lightness_{range}, saturation_{range}$: Length of the interval where all hue / lightness / saturation values of this region are.
- $lightness_{min}, saturation_{min}$: Minimal lightness / saturation value of this region.
- $lightness_{max}, saturation_{max}$: Maximal lightness / saturation value of this region.

The hue values are represented by a cyclic value set (angle between 0° and 360°) where no minimum or maximum exists. In order to determine the mean of the hue values at first the smallest interval with all hue values of the region in the color circle is chosen. This interval is then used for the computation of the mean and interval length [13]. The different lightness and saturation values are represented by integers between 0 and 255.

Besides the color analysis, the *PF* system provides a texture analysis component. For the texture analysis, a segmentation into texture regions is performed by applying region-based and edge-based methods [8]. From the different regions samples are taken and used for the extraction of texture features [9]. The texture features are: shape of primitives (possible values: homogeneous, multi-areas, blob-like), linelikeness, coarseness, regularity, directionality, contrast, softness (possible values are {very low, low, medium, high, very high} represented by {0.0, 0.25, 0.5, 0.75, 1.0}).

2.2 Shape Analysis

This subsection presents a component for the extraction of shape features. The shape analysis assumes that an image has already been divided into regions and takes object images as input. An object image is a black and white image where all pixels of the object are identified by the value one, and all other pixels have the value zero. This region image is transformed in further steps into other representations: object contour, polyline, and convex hull. In this section we describe which features are extracted during which steps. A more detailed description of the shape analysis can be found in [13].

The **object image** can be used to compute the position of the region, its relative size, and the number of lakes (holes) in the object. The *position* of the object is computed by the scaled moments with order 0 ($m_{-s}(0,0)$) and order 1 ($m_{-s}(1,0)$ and $m_{-s}(0,1)$, Eq. 1, cf. [16]). The *relative size* is the ratio of the region size to the image size (Eq. 2).

$$F_{horPos} = \frac{m_{-s}(1,0)}{m_{-s}(0,0)} \text{ and } F_{verPos} = \frac{m_{-s}(0,1)}{m_{-s}(0,0)} \quad (1)$$

$$F_{relSize} = \frac{F_{area}}{F_{imageSize}} \quad (2)$$

For the computation of the *number of lakes* a region growing algorithm like the blob-coloring approach is applied [1]. All regions are counted which are not

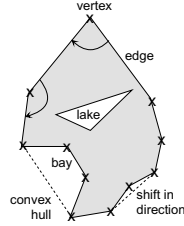


Figure 1. Edges, vertices, shifts in direction, bays and lakes of an object

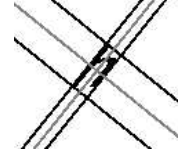


Figure 2. Bounding box and center lines of an object for the balance computations

part of the object, i.e., where the value is zero, and which are not connected to the border of the image. By this approach only regions are counted which are inside the object contour but not part of the object itself. This feature is called F_{lakes} with $F_{lakes} \in \mathbb{N}_0$.

The object image is taken as input for the creation of the **object contour**. It is processed line by line until the first pixel is found. From this starting point, the object contour is tracked. The chain code of the contour could easily be created and used for the computation of some features (e.g., perimeter) at this time. We just use the object contour pixels as input for the polyline computation in order to apply algorithms with better performance on that representation.

The list of contour pixels with their x and y positions is used to create a **polyline** with the “Iterative End-Points Fit” algorithm [6]. The polyline representation is used to extract many features. Vertices can be found between two neighboring edges. We define the occurrence of a vertex if the angle between two edges is less than a threshold $threshold_{corner}$ (e.g., 135°). For the computation of the angle, Pythagoras theorem and trigonometric functions can be used (cf. [13]). The *number of vertices* is called $F_{vertices}$. The *number of edges* F_{edges} is equivalent to the number of lines in the polyline representation. The *number of main lines* counts the number of lines which form the main part of the contour, $F_{mainLines}$. First of all the mean length of all lines is computed and multiplied by an adjustable factor $mainLineFactor$ (e.g., 0.5). All lines exceeding this value are main lines. Neighboring lines are merged to one line if their angle is less than threshold $mainLineAngleThreshold$ (e.g., 170°). A *shift in direction* appears if there is a shift from a right into a left curve (or the other way round). The directions of curves can be identified by computing the signed distance of the following vertex to the current line segment [13]. An example for edges, vertices, and shifts in direction is shown in Fig. 1.

The computation of *perimeter* $F_{polygonPerimeter}$, *area* $F_{polygonArea}$, and *circularity* $F_{circularity}$ (to be calculated using the two prior attributes) can be easily done from the polyline representation [1, 16].

The three moments of second order can be used to calculate the parameters of an ellipse with equivalent moments as the objects moments [7]. This information can be used to compute the lengths and directions of the principal axis and

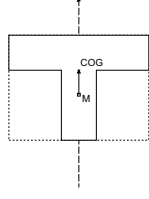


Figure 3. Shift of the center of gravity for a “T”-shape

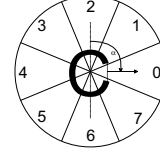


Figure 4. Main bay direction of the letter “C”

the minor axis of this ellipse. Steger presents an approach to compute arbitrary moments from the polyline representation [18]. $F_{direction}$ stands for the *direction*¹ of the principal axis. Having the lengths of the axes (λ_m and λ_n), the *eccentricity* is computed by Eq. 3.

$$F_{eccentricity} = 1 - F_{ellipseRatio} = 1 - \frac{\lambda_n}{\lambda_m} \quad (3)$$

$$F_{convexityFactor} = \frac{F_{polygonArea}}{F_{areaConvexHull}} \quad (4)$$

$$F_{lakeFactor} = \frac{F_{lakeArea}}{(F_{area} + F_{lakeArea})} \quad (5)$$

The *vertical and horizontal balance* describe the balance of the object areas of the left to the right side from the object center, and for the upper to the lower side, respectively. For the computation, the smallest bounding box parallel to the object direction is taken. Two lines g_1 and g_2 through the center are created where g_1 is parallel to the object direction and g_2 is orthogonal to g_1 . Counting the pixels to the left and right of g_1 and above and below g_2 , and correlating these values, the balances $F_{verBalance}$ and $F_{horBalance}$ can be calculated. $F_{verBalance}$ and $F_{horBalance}$ are in the range $[0, 1]$, where the value 0.5 stands for complete balance. Fig. 2 illustrates the bounding box and g_1 and g_2 for a distorted letter “h”. The *center of gravity shift* can be calculated by the features given so far. It is the direction of the vector from the bounding box center to the center of gravity (see Fig. 3).

The Quickhull algorithm is used for the **convex hull** computation from the polyline [2]. An object’s convex hull, lakes, and bays are shown in Fig. 1. The *convexity factor* correlates the areas of the object to the area of the convex hull. It describes how convex an object is (Eq. 4). The *circularity of the convex hull* $F_{chCircularity}$ is computed like the regular circularity of the object by using the area and perimeter of the convex hull representation.

The *lake factor* describes how holey an object is by calculating the ratio of the area of lakes to the area of the object if it had no lakes (Eq. 5). The *number*

¹ For our character recognition experiments the axes have been switched, if the principal axis was nearby 0° to compute the upstanding angle of a character.

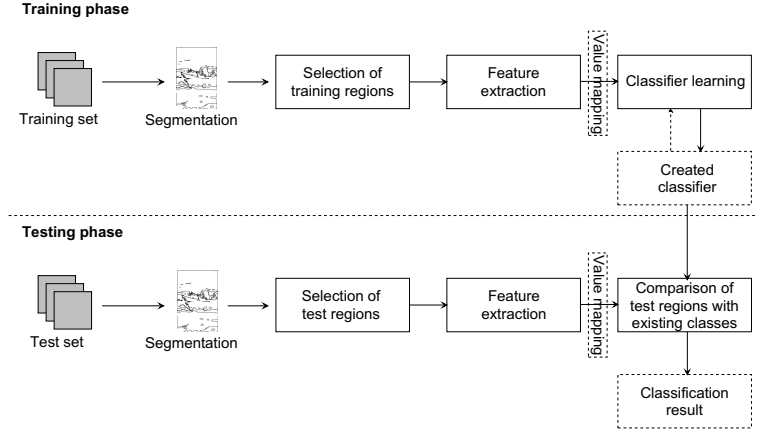


Figure 5. Training and test phases of the region classification system

of bays F_{bays} is determined by comparing the object's polyline to the polyline of the convex hull [13]. The *main bay direction* is the angle between the principal axis and the orthogonal line to the bay entrance. The direction of the main bay is discretized into eight regions as stated in Fig. 4. Instead of taking the direction of the bay with the biggest size, the sizes of the different discrete directions are accumulated. The direction with the biggest sum of bay sizes is the main bay direction $F_{mainBayDirection}$.

3 Machine Learning for Region Classification in Images

In the last section, a set of color, texture, position, and shape features was introduced. This section describes the approach to the automated creation of object classifiers. The following section shows evaluation results for three ML algorithms on two different image sets.

Fig. 5 shows the two main phases of the system. In the training phase, classifiers for different classes are created from example regions by ML algorithms. The user can select different regions of the training images as samples and assign classes to them. During the testing phase, selected regions are tested with the different classifiers. The class with the highest confidence value will be assigned automatically. Now the different steps in both phases are described briefly. The steps up to the value mapping in Fig. 5 are identical for the two phases.

3.1 Segmentation and Feature Extraction

For the segmentation of input images, the existing contour, color or texture segmentation algorithms of the *PF* system can be used. It is possible to apply just one algorithm or a combination of different algorithms, as demonstrated

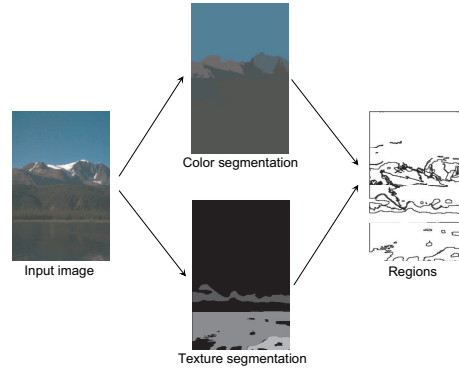


Figure 6. Segmentation of the input image

in Fig. 6. In some domains it might not be reasonable (or even affect results adversely) to use more than one segmentation method.

The features which can be used as input for the ML programs were introduced in section 2. It does not make sense to use all features in all domains, e.g., the color of a text is usually not important for character recognition. It is possible to deactivate features to avoid slowing down or irritating the ML algorithms by irrelevant features.

3.2 Value Mapping

The features span a n -dimensional space where each example can be seen as one point in this space. The size of the space is dependent on the number of attributes and the number of values of the attributes. The number of possible combinations is $\prod_{i=1}^n v_i$ if n features F_1, F_2, \dots, F_n with v_i possible values for F_i exist. If ten attributes with five possible values are used, the number of possible combinations is already 9765625. Value mappings can be applied to reduce the space for the ML algorithms. In our work we used three different kinds of mappings:

- **Discretization.** Mapping of values into a set of discrete values, e.g., mapping $[0, 1]$ to $\{\text{very low, low, medium, high, very high}\}$.
- **Mapping.** Mapping of continuous values to symbolic values, e.g., mapping $\{0.0, 0.25, 0.5, 0.75, 1.0\}$ to $\{\text{very low, low, medium, high, very high}\}$. This is necessary if a ML algorithm can just handle symbolic values.
- **Pruning.** This mapping type prunes an ordered value set after a defined value, e.g., IN_0 to $\{0, 1, 2, 3, 4\}$.

3.3 Learning and Classification

The learning input for the ML programs are the manually classified training samples. For each training sample, all active features are extracted and prepared for the ML programs. After the input is set up, the ML programs are applied.

Depending on the ML approaches, different classifiers are learned, e.g., concept descriptions, a decision tree or a neural net (NN). After classifiers have been learned, they can be applied to unknown regions. In the classification phase classes are assigned automatically to these regions.

Three different ML algorithms have been integrated for the creation of classifiers: C4.5 [17], AQ18 [12] and the backpropagation algorithm for training a neural net [19] using the the Stuttgarter Neuronale Netze Simulator (SNNS). Due to lack of space, these algorithms cannot be described in detail. A description of these algorithms and their parameters can be found in [12, 17, 19]. The next paragraphs just introduce the rough idea of these algorithms.

C4.5 creates decision trees from examples. The root node consists of all examples and at each node the examples are divided depending on their values for a certain attribute. The selection of the attribute is motivated by information theory: at each step the attribute which leads to the highest information gain will be selected until all nodes contain only samples of one class (or until another stop criterion has been satisfied).

AQ18 uses the AQ algorithm to create concept descriptions from samples. The algorithm covers the training data with rules until all examples are covered by at least one rule. In each step one example is taken from the set of (remaining) positive examples and generalized rules are created which do not cover any negative examples. Applying a criterion function, the most preferable rule is taken and all covered examples are removed from the positive example set.

The backpropagation algorithm is a standard learning algorithm for feed-forward networks. The initial weights are chosen randomly and are adjusted by the backpropagation algorithm. In each iteration an example is selected and the difference of the actual output to the real output is calculated. The weights are adjusted to minimize the error until a threshold has been reached. In our case the neural network has the features as input layer, one hidden layer with ten neurons and for each class an output neuron. The weights of the net are randomly initialized between $[0\ 1]$ and each experiment is repeated three times. All feature values are mapped to the range $[0\ 1]$. The learn factor and the number of learn cycles vary at the different experiments.

3.4 Multiple Classification Schemes

Different users often do not have the same idea of how to structure information. Depending on the background and the current task, the user has different needs while searching information in a multimedia retrieval system. Thus, it is desirable to allow the users to define their individual classification schemes. In some cases it might even be useful to define different views on the data for one person or a group of persons in order to support the users performing different tasks.

Diverse classification schemes can address different image regions of disjoint domains or look at the same information from varying perspectives. In the latter case it can be distinguished if different classification schemes just represent other levels of granularity or structure objects in a completely different way.

In order to satisfy these requirements, the system must be able to deal with different classification schemes. In combination with user profiles and possibly some basic organization management functionality like the management of groups or roles, the different classification schemes can be assigned to different users and groups. While working with the system, the user can select one of his available views on the information sources. By selecting a certain view, the available concepts for retrieval tasks and training procedures are adapted to the current view. Depending on the domain, a selection of attributes to use for learning and testing can be performed.

4 Experiments

In two experiments we investigated the quality of ML technologies for the automated classification of image regions. In the first experiment, classifiers are learned for character recognition from shape features. In the second experiment, color, texture, and position features are used as input for learning. In all experiments the sets of training and testing samples have been disjoint.

Different settings for the value set mappings and configuration of the ML algorithms have been tested. The used configurations for AQ18 [12] are:

- **Config. 1** Intersections between rules are allowed (mode="ic"), trim to short rules (trim="mini"), no noise.
- **Config. 2** Intersections between rules are not allowed (mode="dc"), trim to short rules (trim="mini"), no noise.
- **Config. 3** Intersections between rules are allowed (mode="ic"), specialized rules (trim="spec"), no noise.
- **Config. 4** Intersections between rules are allowed (mode="ic"), trim to short rules (trim="mini"), handle noise (q_weight="0.25", rule_probe="2", rtol="20").

In all four configurations these AQ parameters were the same: maximum size of the star (max_star="10"), handling of ambiguous examples (ambig="max"), preference criterion (default: LEF="maxnew, minsel"), and testing method (default: INLEN mode).

The configurations for C4.5 [17] are:

- **Config. 1** No pruning, at least 2 branches with 2 samples (m="2").
- **Config. 2** Pruning (cf="25%"), at least 2 branches with 2 samples (m="2").
- **Config. 3** Pruning (cf="10%"), at least 2 branches with 2 samples (m="2").
- **Config. 4** Pruning (cf="25%"), at least 2 branches with 4 samples (m="4").

4.1 Character Recognition

Our character recognition experiments do not try to compete with existing OCR algorithms on printed texts. They are rather suitable for displayed texts in videos



Figure 7. Sample image for character recognition

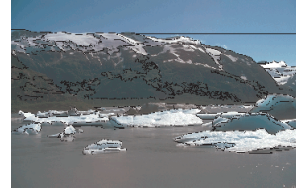


Figure 8. Sample image for landscape region classification

or of signs in photographs, where object borders are blurred. The character recognition experiments have two aims: studying the different learning algorithms and evaluating the shape analysis component. It has to be shown if the chosen features are sufficient to distinguish different classes of shapes. The used features in these experiments are: circularity, eccentricity, convexity, number of vertices, edges, shifts in direction, lakes, bays, main lines, circularity of the convex hull, horizontal and vertical balance, lake factor, cog shift, and main bay direction. For an experiment with separate classes of capital and lower case letters, an additional feature turned out to be helpful to overcome similarities between many letters which almost look the same in capital or lower case form (e.g., “O”). It is the relative size of a letter in relation to the biggest letter in the current image, where $F_{maxArea}$ is the area of the biggest letter:

$$F_{relSize2biggest} = \frac{F_{area}}{F_{maxArea}} \quad (6)$$

In these experiments three value mappings have been used:

- **Value mapping 1.** Continuous value sets are mapped to five intervals. The cog shift values are mapped to eight intervals. The number of vertices, edges, main lines are pruned after the 30th value, the number of shifts in direction and bays after the 20th, and the number of lakes after the 10th. The main bay direction stays the same in all experiments (eight values).
- **Value mapping 2.** Continuous value sets are mapped to 20 intervals. The cog shift values are mapped to 16 intervals. The other mappings are like value mapping 1.
- **Value mapping 3.** The continuous value sets in this setting stay untouched. The pruned values sets are mapped as in the first two settings.

In the first part of the experiment it is tested whether the shape features are sufficient to distinguish 26 (or 52) different letter shapes. Eight different images with 52 letters of the Arial font (normal, distorted to left or right, horizontally and vertically stretched, rotated to left or right, scaled down) were used as training and testing input. In the different runs with varied configuration seven images were used for training and one for testing (i.e., 12.5% of the samples). In the first run, capital and lower case letters were classified, in the second run only capital letters. Only the second value mapping was used here. C4.5 and AQ18

Table 2. Best accuracies of the ML algorithms in the character recognition domain

Run	AQ18	C4.5	NN
Single font			
Capital and lower case let.	79.6%	83.9%	84.1%
Capital let.	88.5%	94.3%	87.5%
Different fonts			
52 classes, mapping 1	53.5%	61.1%	56.2%
52 classes, mapping 2	62.5%	64.1%	66.0%
52 classes, mapping 3	-	67.1%	67.0%
26 classes, mapping 2	61.1%	69.8%	68.6%
Capital let., mapping 2	70.2%	75.2%	80.3%
Capital let., mapping 3	-	77.7%	76.7%
Disjoint fonts			
52 classes, mapping 1	47.5%	53.0%	51.3%
52 classes, mapping 2	51.6%	51.2%	58.0%
52 classes, mapping 3	-	55.4%	58.0%
26 classes, mapping 2	52.1%	60.7%	62.5%
Capital let., mapping 2	60.8%	65.4%	71.7%
Capital let., mapping 3	-	66.6%	69.5%

Table 3. Best accuracies of the ML algorithms in the landscape image domain

Run	AQ18	C4.5	NN
Unknown images			
Mapping 1	61.1%	62.1%	61.9%
Mapping 2	53.4%	61.8%	64.3%
Mapping 3	-	63.4%	63.0%
Mapping 3, less attr.	-	65.2%	63.6%
Unknown regions			
Mapping 2	68.7%	71.2%	67.9%
Mapping 3	-	75.1%	77.7%

were run with the different settings in all experiments; the NN was trained over 500 learning cycles with learn factor 0.8. Table 2 summarizes the best average accuracies of all character recognition experiments. At the runs with all letters, AQ18 has the best average accuracy at the setting with most specialized rules and C4.5 with pruning (config. 2). C4.5 with 83.9% outperforms AQ18 (79.6%). The NN has the best result of 84.1% accuracy. If only capital letters are used, the accuracies are higher. C4.5 has the best result with the unpruned decision tree (94.3%), followed by AQ18 (config. 3 as above) with 88.5%. The NN classified 87.5% of the test samples correctly.

In the second part of this experiment, ten different fonts were used as training and testing (10%, four different test sets) samples. Five of them are with serifs, the other five without. Besides the settings of 52 classes (separated classes for capital and lower case letters) and 26 classes (only capital letters), there was another setting with 26 classes, where capital and lower case letters form one class (e.g., “W” and “w” are in the same class). In these experiments the NN was trained for 1000 cycles. The learn factor was initially set to 1.0 and reduced by 0.1 every 100 learn cycles. In the runs with 52 classes, the best average result for AQ18 is 62.5% (short rules and mapping 2). C4.5 performed best with continuous values and pruning (config. 2) with the accuracy of 67.1%. The NN almost has the same accuracy here (67.0%). If the classes are put together into 26 classes, C4.5 has an accuracy of 69.8% with the same settings. The NN also performs better with 68.6%. AQ18 classifies 61.1% correctly with the configuration that handles noise. If just capital letters are regarded, all algorithms have higher accuracies: 80.3% with the NN (mapping 2), 77.7% with C4.5 (mapping 3, config. 3) and 70.2% with AQ18 (mapping 2, short rules).

In the third part of the character recognition experiment, the fonts of training and testing were completely disjoint, i.e., if some letters of a font are used as training, none of the remaining letters of this font is allowed to be in the test set (10% of samples, four different test sets). If 52 classes are distinguished, the best average result of the NN is with mapping 3 (58.0%). The best accuracy

<pre> B-outhypo # rule 1 [holes=2] U-outhypo # rule 1 [bays=1] [relSize2biggest=12..13] [cogShift=4..10] [mainBayDirection=2..3] Y-outhypo # rule 1 [edges=8..9] [bays=3] </pre>	<pre> holeFactor <= 0 : curves <= 2 : relSize2biggest > 9 : bays <= 1 : curves <= 1 : relSize2biggest <= 13 : mainBayDirection > 1 : relSize2biggest > 11 : U curves > 2 : relSize2biggest <= 10 : edges <= 9 : Y holeFactor > 0 : bays > 0 : convexity > 16 : holes > 1 : B </pre>	<pre> Sky-outhypo # rule 1 [uniformity=homogene] [color=blue,green,bluegreen,green,lightgrey] [hueMean=11..16] [lightnessStd=0] [saturationMean=1..8] 2 (...) SnowIce-outhypo # rule 1 [color=bluegreen,green,lightgrey] [hueMean=9..12] [hueStd=0..1] [lightnessMean=4..10] [lightnessStd=1..2] [verPosition=2..7] 2 (...) </pre>
--	--	---

Figure 9. AQ rules for letters

Figure 10. C4.5 tree for letters

Figure 11. AQ rules for landscape region classes

of C4.5 is 55.4% with mapping 3 and pruning (config. 3). AQ18 performs best with short rules and mapping 2 (51.6%). If capital and lower case letters are put together, higher accuracies can be achieved: 52.1% with AQ18 (handling noise), 60.7% with C4.5 (no pruning), and 62.5% with the NN. If just capital letters are used, the accuracies are even higher: 71.7% for the NN, 66.6% for C4.5 (mapping 3, no pruning), and 60.8% for AQ18 (mapping 2, handling noise). Fig. 7, 9, and 10 show created rules, a created decision tree, and one of the input images for the character recognition experiments.

4.2 Region Classification in Landscape Images

The second experiment analyzes how the ML methods perform using the color, texture, and position features for the classification of regions in landscape images. The different classes are: water (128 samples), sky (122), cloud (135), forest (133) and snow (125, including ice). In some cases bushes and green spaces have also been assigned to the class forest in order to get more sample regions. 643 regions from 41 images [5] were used altogether for training and testing (ca. 10% in all experiments). All texture features, the color identifier, vertical position, and the statistical color features² were used in these experiments. The NN was trained for 500 learning cycles with an initial learn factor of 1.0 (reduced every 100 cycles by 0.2). Three different value mappings were used:

- **Value mapping 1.** The statistical values were discretized into eight and the vertical position into five intervals. The values for the texture features and the color identifier were not changed in all three mappings.
- **Value mapping 2.** In the second mapping, the statistical values are divided into 16, the vertical position into ten intervals.
- **Value mapping 3.** In this setting the continuous values are not changed.

In the first run, completely unseen landscape images were used for testing. AQ18 has an accuracy of 61.1% with mapping 1 and handling noise. C4.5 per-

² In some experiments the ranges, minimal and maximal values were omitted experimentally.

forms a little better with 65.2% using mapping 3, no pruning and the reduced feature set. The NN classified 64.3% correctly with mapping 2.

In the second run, the regions of all images were divided randomly into training and testing regions, i.e., training and testing regions are allowed to be from the same image, but the training and testing sets are still disjoint. As the used landscape images were quite diverse, this experiment shows how advantageous it is if regions from the same image are in the training events. If the accuracies are much better in these experiments, the classifiers before did not learn general descriptions for the different concepts, but are specialized in the events from the training images. Indeed, here the results are better: the NN has the best accuracy (77.7% with mapping 3), followed by C4.5 with 75.1% (mapping 3, config. 3). AQ18 (handling noise) classified 68.7% correctly. Fig. 8 and 11 show created rules and an input image for the landscape experiments.

5 Related Work

An older version of AQ was applied to recognize textures [14]. The “Multi-Level Logical Template” (MLT) method by Michalski extracts local features by a sliding operator window, which are used to learn class descriptions for textures [15]. The MLT method has been enhanced in different ongoing works, e.g., by Chan- nic by applying filters and incremental learning on ultrasound images [4]. The MLT has been extended to the “Multi-Level Image Sampling and Transformation” (MIST) method and was used for the segmentation of landscape images and identification of blasting caps in x-ray images [15].

C4.5 was also used by Zrimec and Wyatt for object recognition under different lighting conditions [20]. The task here was to classify different regions in soccer scenes of the Sony legged robot league like goals, walls, and ground. The regions were characterized by different color (e.g., average red / green / blue values, hue and saturation), position, and shape (area, perimeter, “wiggleness”, scaled moments of inertia) features. By applying C4.5 more than 98% of the 631 training regions were correctly classified by the learned decision tree. The paper does not show evaluation results for regions which have not been used for training.

Campbell et al. address the automatic interpretation of outdoor scenes [3]. In their work they use texture (eight isotropic Gabor filters), color, contextual (surrounding areas), and shape (principal modes of variation of approximating polygons, size, position) features to train a neural network classifier. They report an accuracy of 82.9% at region classification on over 3000 unseen test regions. The correctly classified regions cover 91.1% of all image areas.

An empirical comparison between decision trees and NNs for the detection of defects in austenitic welds was performed by Jacobsen et al. [11]. In this work x-ray images are decomposed into regions of interests, where 36 features are extracted by different image processing procedures. In the experiments the NNs perform slightly better and use less attributes than the decision trees.

In our approach, different image segmentation algorithms can be applied and combined to create image regions. Out of these regions, different feature extrac-

tion methods are applied to extract texture, color, position, and shape features. The shape analysis combines well-known existing and some new extraction methods to create mainly easy to understand features. The system allows a flexible handling of the different feature extraction methods by enabling the user to select the attributes and their value mappings for training and classification.

6 Conclusion and Future Research

The experiments with the variations of the Arial font pointed out that the set of shape features is sufficient to distinguish at least 26 shape classes (94.3% in the best case). Many of the created rules of the symbolic learning approaches are comprehensible. In the more complex experiments the accuracies were worse. This can be explained by the problems with fonts with serifs and high similarity between some letters (e.g., “l”, “i” and “I”). The experiments on the landscape images with different regions are still satisfying (up to 77.7% in the best cases).

The NN performed best altogether, followed by C4.5 and AQ18. Regarding the readability it is the other way round. The rules created by AQ18 are quite compact and understandable. Decision trees can be hard to read if they have many nodes. The learned weights of a neural net give almost no information to the user. The choice of the ML algorithm depends on the needs of the user. If the rules have to be understandable, one of the symbolic learning approaches would be preferable. If highest accuracy is more important, the NN should be used.

Support Vector Machines (SVM) have recently become popular for classification tasks and outperform many other classifiers. Thus, it would be interesting to take SVM into account in future comparisons. As in this paper the shape analysis and classification was applied to a set of artificially created (and modified) characters, a real world application on video text captions should be performed. Future experiments can also investigate how context information (like neighborhood of a character) and a dictionary or statistical information about characters (e.g., in form of n-grams) could improve the classification results. Problems with fonts with serifs could be solved by adapting fonts, e.g., by removing serifs.

Acknowledgements

We wish to express our gratitude to Ryszard S. Michalski and Guido Cervone from the Machine Learning and Inference Laboratory at the George Mason University for fruitful discussions and support with the ML tool AQ18.

References

1. D. H. Ballard and C. Brown. *Computer Vision*. Prentice-Hall Inc., Englewoods Cliff, New Jersey, USA, 1982.
2. C. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The Quickhull algorithm for convex hulls. *ACM Transaction on Mathematical Software*, 22(4):469–483, 1996.

3. N. W. Campbell, W. P. J. Mackeown, B. T. Thomas, and T. Troschianko. Interpreting image databases by region classification. *Pattern Recognition (Special Edition on Image Databases)*, 30(4):555–563, April 1997.
4. T. Channic. Texpert: An application of machine learning to texture recognition. MLI Report, Machine Learning and Inference Laboratory, George Mason University, Fairfax, VA, USA, 1989.
5. Corel professional photos cd-rom, nature vol. 7, 1994.
6. R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, New York / London / Sydney / Toronto, 1973.
7. R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision*. Addison-Wesley Publishing Company, Reading, MA, USA, 1992.
8. T. Hermes, A. Miene, and P. Kreyenhop. On textures: A sketch of a texture-based image segmentation approach. In *Classification and Information Processing at the Turn of the Millenium*, pages 210 – 218, 2000.
9. T. Hermes, A. Miene, and O. Moehrke. Automatic texture classification by visual properties. In *Classification and Information Processing at the Turn of the Millenium*, pages 219 – 226, 2000.
10. O. Herzog, A. Miene, T. Hermes, and P. Alshuth. Integrated information mining for texts, images, and videos. *Comput. & Graphics*, 22(6), 1998.
11. C. Jacobsen, U. Zscherpel, and P. Perner. A comparison between neural networks and decision trees. In *Proceedings of the First International Workshop on Machine Learning and Data Mining in Pattern Recognition, MLDM'99, LNAI 1715*, pages 144–158, Leipzig, Germany, September 1999. Springer-Verlag Berlin Heidelberg.
12. K. Kaufman and R. S. Michalski. The AQ18 machine learning and data mining system: An implementation and user's guide. MLI Report, Machine Learning and Inference Laboratory, George Mason University, Fairfax, VA, USA, 1999.
13. A. D. Lattner. Automatische Generierung von Klassifikationsregeln aus Farb-, Textur- und Konturmerkmalen durch Induktives Lernen. Master's thesis, University of Bremen, Germany, 2000.
14. R. S. Michalski. A variable-valued logic system as applied to picture description and recognition. In F. Nake and A. Rosenfeld, editors, *Graphic Languages*, pages 21–47. North-Holland, Amsterdam, 1972.
15. R. S. Michalski, A. Rosenfeld, Z. Duric, M. Maloof, and Q. Zhang. Learning patterns in images. In *Machine Learning and Data Mining - Methods and Applications*, chapter 10, pages 241–268. John Wiley & Sons Ltd., New York, 1998.
16. W. K. Pratt. *Digital Image Processing, Second Edition*. John Wiley & Sons, Inc., New York / Chichester / Brisbane / Toronto / Singapore, 1991.
17. J. R. Quinlan. *C4.5 - Programs for Machine Learning*. Morgan Kaufmann Publishers, Inc., 1993.
18. C. Steger. On the calculation of arbitrary moments of polygons. Technical report, Technische Universität München, 1996.
19. A. Zell. *Simulation neuronaler Netze*. R. Oldenbourg Verlag, München / Wien, 3rd, unchanged reprint edition, 1994.
20. T. Zrimec and A. Wyatt. Learning to recognize objects- toward automatic calibration of color vision for Sony robots. In *ICML Workshop on Machine Learning in Computer Vision*, Sydney, Australia, July 9 2002.