At present a great deal of research is being done in different aspects of Content-Based Image Retrieval (CBIR). Image classification is one of the most important tasks that must be dealt with in image DB as an intermediate stage prior to further image retrieval. The issue we address is an evolution from the simplest to more complicated classifiers. Firstly, there is the most intuitive one based on a comparison of the features of a classified object with a class pattern. We propose a solution to the problem of finding the adequate weights, especially in the case of comparing complex values of some features. Secondly, the paper presents decision trees as another option in a great number of classifying methods. Thirdly, to assign the most ambiguous objects we have built fuzzy rule-based classifiers. We propose how to find the ranges of membership functions for linguistic values for fuzzy rule-based classifiers according to crisp attributes. In this paper, we present the promising results of the three above-mentioned classifications. Experiments demonstrate the precision of each classifier for the crisp image data in our CBIR. Furthermore, these results are used to construct a search engine, taking into account data mining. If the classification precision appears insufficient for the search engine requirements, in the next step fuzzy decision trees will be introduced.

1. INTRODUCTION

In recent years, the availability of image resources and large image datasets has increased enormously. This has created a demand for effective and flexible techniques for automatic image classification and retrieval. Although attempts to construct the Content-Based Image Retrieval (CBIR) in an efficient way have been made before, a major problem in this area, which is the extraction of semantically rich metadata from computationally accessible low-level features, still poses a tremendous scientific challenge. Images and graphical data are complex in terms of visual and semantic contents. Depending on the application, images are modelled using their

* Tatiana JAWORSKA*

Systems Research Institute, Polish Academy of Sciences, 01-447, 6 Newelska Street, Warsaw, Poland, e-mail: Tatiana.Jaworska@ibspan.waw.pl
• visual properties (or a set of relevant visual features) [3],
• semantic properties [2], [14],
• spatial or temporal relationships of graphical objects [4].

The classification problem is crucial for multimedia information retrieval in general, and for image retrieval in particular. There are a number of standard classification methods in use such as: k-NN [5], SVM [7], naïve Bayes classifier [18], neural network [20], and others [1]. Having surveyed these methods, we started our classification from the simplest algorithm, namely, the similarity to the pattern which compares the features of a classified object with the set of pattern features which define classes.

Object classification is so important in the context of CBIR because it is used for several purposes, for example [10]:
1. to compare whole images. Specifically, an algorithm which describes a spatial object location needs classified objects.
2. to help the user form a query in the GUI. The user forms a query choosing graphical objects semantically collected in groups.
3. to compare image objects coming from the same class as a stage in the image retrieval process. Details are presented in sec. 5.

Generally, the classification problem can be defined as follows. Let $\Omega$ be a complete set of objects which we want to automatically recognize, hence we want to define a division into $k$ separate classes $c_1, \ldots, c_k$. It means that there must be a division function $\Theta$, such as:

$$\Theta: \Omega \rightarrow L = \{1, \ldots, k\}$$

which assigns to each object of the set $\Omega$ a particular class. We do not know the assignment rules, but only know the $\Omega$ subset that we call the learning or training subset.

1.1. CBIR CONCEPT OVERVIEW

In general, our system consists of five main blocks (see Fig. 1):
1. the image preprocessing block (responsible for image segmentation), implemented in Matlab, (cf. [12]);
2. the database, which is implemented in the Oracle Database (DB), stores information about whole images, their segments (here referred to as graphical objects), segment attributes, object location, pattern types and object identification, (cf. [11]);
3. The classification module, which is used by the search engine and the GUI, is implemented in Matlab. The algorithms applied in this module will be described in the following sections.

![Block diagram of our content-based image retrieval system.](image)

4. The search engine responsible for the searching procedure and retrieval process based on feature vectors of objects and spatial relationship of these objects in an image, implemented in Matlab;

5. The graphical user's interface (GUI), which allows users to compose their own image, consisting of separate graphical objects as a query. Classification helps in the transition from rough graphical objects to human semantic elements.

We have had to create a user-friendly semantic system, also implemented in Matlab.

1.2. REPRESENTATION OF GRAPHICAL DATA

In our system, a new image is segmented, yielding as a result a collection of objects. Both the image and the extracted objects are stored in the database. Each object, selected according to the algorithm presented in detail in [12], is described by some low-level features. The features describing each object include: average colour $k_{av}$, texture parameters $T_p$, area $A$, convex area $A_c$, filled area $A_f$, centroid $\{x_c, y_c\}$, eccentricity $e$, orientation $\alpha$, moments of inertia $m_{11}$, $m_{12}$, $m_{21}$, $m_{22}$, bounding box $\{bb_1(x,y),...,bb_4(x,y)\}$, major axis length $m_{long}$, minor axis length $m_{short}$, solidity $s$, Euler number $E$, Zernike moments $Z_{00},...,Z_{33}$ [19], and some others.

Let $F_O$ be a set of features where:

$$F_O = \{k_{av}, T_p, A, A_c, ..., E\}$$

Hence, for an object, we construct a feature vector: $x = [x_1, x_2, ..., x_r]$, where $n$ is the number of the above-mentioned features, in our system $r = 45$. 
2. SIMILARITY TO PATTERN

The simplest approach to the classification is the comparison of an object feature vector \( x \) to the previously prepared patterns \( P_k \) for each class. Patterns can be created in different ways. The simplest method is the calculation of the average value of each vector component. The subsets of objects used to define particular patterns are also used as learning subsets. In order to compare the object vector with a pattern we apply the Euclidean metrics:

\[
d(x, P_k) = \sqrt{\sum_{i=1}^{r} \xi_k (x_i) [x_i - P_k(x_i)]^2}
\]

where: \( k \) – pattern number, \( 1 \leq i \leq r \). All pattern vectors are normalized. A new object is classified to a class for which \( d \) is the minimum [3], [10].

We also assume weights \( \xi_k (i) \) for all pattern features where: \( i \) is the number of feature, \( 1 \leq i \leq r \). Weights for real features are the coefficients of variation

\[
\xi_{P_k} = \frac{\sigma(i)}{\bar{x}(i)} \in [0,1]
\]

in order to reflect the dispersion of each feature in the subset selected as a pattern (where \( \sigma \) – standard deviation and \( \bar{x} \) – mean value for each feature). However, Zernike’s moments are complex features, hence to obtain the real weight we apply the formula [6]:

\[
\xi(i) = \sqrt{\frac{\sigma_{\text{Re}}^2 + \sigma_{\text{Im}}^2}{\bar{x}_{\text{Re}}^2 + \bar{x}_{\text{Im}}^2}}
\]

where standard deviations and means are calculated separately for real and imaginary parts of complex moments.

For all these classes we have created the pattern library (also stored in the DB) which contains information about pattern types, weights and objects belonging to learning subsets [11].

We decided to classify separately objects with and without texture to reduce the misclassification between these two groups. This division diminishes the number of classification errors resulting from the fact that the pattern for non-textured objects gave a smaller \( d \) in spite of introducing weights. All results are presented in sec. 5.
3. DECISION TREES

A decision tree represents a function that takes as an input a vector of attribute values and returns a single output value as a “decision”. We consider a list of attributes of our objects \( \{x_1, x_2, ..., x_r \} \) and classes \( C = \{c_1, ..., c_k \} \). A learning subset contains examples associated with both values of the attributes and a class [15].

Each attribute \( x_i \) can be either symbolic, numerical, or fuzzy. In our case, attributes are numerical: real and complex. The aim of the inductive process is to find a general rule to point out the relation between values of attributes and classes in \( C \). The inductive method is based here on a decision tree from the learning subset.

In the construction of decision trees, a measure of discrimination is used in order to rank attributes and select the best one. Each vertex of a binary tree is associated with an attribute [16]. We construct our trees using the Matlab function \texttt{ClassificationTree.fit} (\texttt{training_set,classes}).

In order to avoid high error rates resulting from as many as 24 classes we use the hierarchical method. The more general division is created by dividing the whole data set into four clusters applying \( k \)-means clustering. The most numerous classes of each cluster constituting a meta-class are assigned to four decision trees, which results in 6 classes for each one.

The second stage of the method, after constructing the trees, is the classification of a new object on the basis of its values of the feature vector. This stage is also realized by the Matlab function \texttt{predict(tree,X_new)}.

4. FUZZY CLASSIFICATION

The results presented in sec. 5 indicate that there are objects difficult for classification. Some difficulties arise from the fact that there are imbalanced classes and mistakes in object segmentation. All this motivated us to use the fuzzy rule-based classifiers.

4.1. FUZZY RULE-BASED CLASSIFIERS

Let us consider an \( M \)-class classification problem in an \( n \)-dimensional normalized hyper-cube \([0, 1]^n\). For this problem, we use fuzzy rules of the following type [8]:

\[
\text{Rule } R_q : \text{If } x_1 \text{ is } A_{q1} \text{ and } ... \text{ and } x_n \text{ is } A_{qn} \text{ then Class } C_q \text{ with } CF_q, \quad (6)
\]

where \( R_q \) is the label of the \( q \)-th fuzzy rule, \( x = (x_1, ..., x_n) \) is an \( n \)-dimensional feature vector (2), \( A_{qi} \) is an antecedent fuzzy set \((i = 1, ..., n)\), \( C_q \) is a class label, \( CF_q \) is a real number in the unit interval \([0,1]\) which represents a rule weight. The rule weight can
be specified in a heuristic manner or it can be adjusted, e.g. by a learning algorithm introduced by Ishibuchi et al. [17], [9].

We use the \(n\)-dimensional vector \(A_q = (A_{q1}, ..., A_{qn})\) to represent the antecedent part of the fuzzy rule \(R_q\) in (6) in a concise manner.

A set of fuzzy rules \(S\) of the type shown in (6) forms a fuzzy rule-based classifier. When an \(n\)-dimensional vector \(x_p = (x_{p1}, ..., x_{pn})\) is presented to \(S\), first the compatibility grade of \(x_p\) with the antecedent part \(A_q\) of each fuzzy rule \(R_q\) in \(S\) is calculated as the product operator

\[
\mu_{A_q}(x_p) = \mu_{A_{q1}}(x_{p1}) \times ... \times \mu_{A_{qn}}(x_{pn}) \quad \text{for} \quad R_q \in S,
\]

where \(\mu_{A_{qi}}(\cdot)\) is the membership function of \(A_{qi}\). Then a single winner rule \(R_{w(x_p)}\) is identified for \(x_p\) as follows:

\[
w(x_p) = \arg \max_q \{CF_q \times \mu_{A_q}(x_p) | R_q \in S\},
\]

where \(w(x_p)\) denotes the rule index of the winner rule for \(x_p\).

The vector \(x_p\) is classified by the single winner rule \(R_{w(x_p)}\) belonging to the respective class. If there is no fuzzy rule with a positive compatibility grade of \(x_p\) (i.e., if \(x_p\) is not covered by any fuzzy rules in \(S\)), the classification of \(x_p\) is rejected. The classification of \(x_p\) is also rejected if multiple fuzzy rules with different consequent classes have the same maximum value on the right-hand side of (8). In this case, \(x_p\) is on the classification boundary between different classes. We use the single winner-based fuzzy reasoning method in (8) for pattern classification.

An ideal theoretical example of a simple three-class, two-dimensional pattern classification problem with 20 patterns from each class is considered by Ishibuchi and Nojima [8] (Fig. 2 a)). There are three linguistic values (small, medium and large) used as antecedent fuzzy sets for each of the two attributes, and 3x3 fuzzy rules were generated. \(S_1\) was the fuzzy rule-based classifier with nine fuzzy rules shown below:

\(S_1\): fuzzy rule-based classifier with nine fuzzy rules

\(R_1\): If \(x_1\) is small and \(x_2\) is small then Class2 with 1.0,
\(R_2\): If \(x_1\) is small and \(x_2\) is medium then Class2 with 1.0,
\(R_3\): If \(x_1\) is small and \(x_2\) is large then Class1 with 1.0,
\(R_4\): If \(x_1\) is medium and \(x_2\) is small then Class2 with 1.0,
\(R_5\): If \(x_1\) is medium and \(x_2\) is medium then Class2 with 1.0,
\(R_6\): If \(x_1\) is medium and \(x_2\) is large then Class1 with 1.0,
\(R_7\): If \(x_1\) is large and \(x_2\) is small then Class3 with 1.0,
\(R_8\): If \(x_1\) is large and \(x_2\) is medium then Class3 with 1.0,
\(R_9\): If \(x_1\) is large and \(x_2\) is large then Class3 with 1.0.

For simplicity, the rule weight is 1.0 in \(S_1\). The location of each rule is shown in (Fig. 2 b)).
4.2. CONSTRUCTION OF MEMBERSHIP FUNCTIONS

The theoretical method presented by Ishibuchi does not answer the question how to construct membership functions, especially those corresponding to the linguistic values. We solved this problem calculating the mean value $\bar{x}$ and standard deviation $\sigma$ for the elements of each of the three classes. The membership function of each class is constructed as a trapezoidal function (see Fig. 3), where points $b$ and $c$ are in the $\pm \sigma/2$ distance from the mean value $\bar{x}$, and the basis points $a$ and $d$ are $\pm \sigma$ distant from the mean value.

Then, we divide the ranges of features $x_1$ and $x_2$ into three equal intervals. Next, we assign the mean value of a particular class to correspondent intervals. The effect is visible in Fig. 4 for the horizontal and vertical axes.

In each case, the fuzzy rule-based classifier is constructed automatically by matching the membership function related to the proper linguistic value, resulting in the right class for each rule. The classifier $S_2$ corresponds to the example seen in Fig. 4:

**S_2**: fuzzy rule-based classifier with nine fuzzy rules
- $R_1$: If $x_1$ is small and $x_2$ is small then non-defined with 1.0,
- $R_2$: If $x_1$ is small and $x_2$ is medium then balkon with 1.0,
- $R_3$: If $x_1$ is small and $x_2$ is large then arc with 1.0,
Fig. 4. Classification example. The new element marked by the full green square is recognized as an arc. Membership functions are represented by solid colour lines and linguistic intervals are drawn in dashed lines.

- $R_4$: If $x_1$ is medium and $x_2$ is small then non-defined with 1.0,
- $R_5$: If $x_1$ is medium and $x_2$ is medium then balkon with 1.0,
- $R_6$: If $x_1$ is medium and $x_2$ is large then non-defined with 1.0,
- $R_7$: If $x_1$ is large and $x_2$ is small then filar with 1.0,
- $R_8$: If $x_1$ is large and $x_2$ is medium then non-defined with 1.0,
- $R_9$: If $x_1$ is large and $x_2$ is large then non-defined with 1.0.

The winner is the rule for which the product operator is maximum (cf. (7)), as follows:

$$\mu_{R_3}(x_p) = \mu_{\text{small}}(x_1) \times \mu_{\text{large}}(x_2) = \mu_{\text{small}}(8.6383) \times \mu_{\text{large}}(0.1506) = 1 \times 1 = 1$$

The fuzzy rule-based classifier is stable, irrespective of attribute selection. We treat it as a "decisive voice" in the case of differences between Euclidean and decision tree classifications.
5. RESULTS

Our learning set consists of 472 objects, which gives about 20 objects per each of the 24 classes. Based on it we classified 532 new objects of all classes and we obtained the total precision of 21.5% for the similarity to pattern algorithm, 68.6% for decision trees and 88% for the fuzzy rule-based classifier FRBC (see Tab. 1).

<table>
<thead>
<tr>
<th>Precision</th>
<th>Similarity to pattern</th>
<th>Decision trees</th>
<th>FRBC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total (for 24 classes)</strong></td>
<td>21.5%</td>
<td>68.6%</td>
<td>88%</td>
</tr>
<tr>
<td>Window-pane</td>
<td>16.1%</td>
<td>72%</td>
<td>89.7%</td>
</tr>
<tr>
<td>Window</td>
<td>46.7%</td>
<td>61%</td>
<td>57.6%</td>
</tr>
<tr>
<td>Brick wall</td>
<td>9%</td>
<td>45.5%</td>
<td>90.9%</td>
</tr>
<tr>
<td>Arc</td>
<td>63.6%</td>
<td>68.2%</td>
<td>58%</td>
</tr>
<tr>
<td>Roof edge</td>
<td>8.4%</td>
<td>86.7%</td>
<td>93.9%</td>
</tr>
</tbody>
</table>

The high rate of false classification in the similarity to pattern algorithm results from extensive aggregation of information. Although the weights are used, all the features are involved in eventual class assignment, whereas, in the case of trees, only the most informative features are selected. Our classification process is divided into four trees due to the number of meta-classes.

The FRBC is in ‘the best situation’ because it is used to distinguish only among tree classes.

An additional problem, which we avoided in the learning set construction, arises from imbalanced classes. In the proper classification, however, it is inevitable.

6. CONCLUSIONS

The results presented here seem to be encouraging to move forward to the next stages of the CBIR system preparation, namely, to the description of spatial object location, the GUI and the search engine. The methods already implemented will be also evaluated in terms of the addition of new classes to the system. GUI development will also enforce introducing subclasses to some of the most numerous classes.

If classification precision turns out insufficient, we will have to apply fuzzy decision trees [13] or other more sophisticated methods.
REFERENCES


