EU-IST Project IST-2003-506826 SEKT
SEKT: Semantically Enabled Knowledge Technologies

D1.3.1 Dealing with different data types

Janez Brank
Dunja Mladenic, Marko Grobelnik
(Jožef Stefan Institute)

Abstract
Although the Semantic Web and related technologies are usually focused on textual or structured data, it is also important to consider techniques for handling other sources of data. Of particular interest are multimedia data such as images and sound and video clips. In this report we present an overview of techniques that can be used to represent images as a first step towards further processing, for example for clustering, categorization, or retrieval. We describe a software module that can be used to automatically extract several types of representations from images or sets of images. The representations currently supported include histograms, autocorrelograms, and banded autocorrelograms. As an example of use of these representations, we present results of experiments in which these representations were used as a basis for image categorization.

Keyword list: Semantic Web, multimedia, pictorial databases, image categorization, image retrieval, histogram, autocorrelogram, color space quantization

WP1
Prototype/Report PU
Contractual date of delivery: 31.12.2004
Actual date of delivery: 12.1.2005

SEKT Consortium

This document is part of a research project partially funded by the IST Programme of the Commission of the European Communities as project number IST-2003-506826.

British Telecommunications plc.
Orion 5/12, Adastral Park
Ipswich IP5 3RE
UK
Tel: +44 1473 609583, Fax: +44 1473 609832
Contact person: John Davies
E-mail: john.nj.davies@bt.com

Empolis GmbH
Europaallee 10
67657 Kaiserslautern
Germany
Tel: +49 631 303 5540
Fax: +49 631 303 5507
Contact person: Ralph Traphöner
E-mail: ralph.traphoener@empolis.com

Jozef Stefan Institute
Jamova 39
1000 Ljubljana
Slovenia
Tel: +386 1 4773 778, Fax: +386 1 4251 038
Contact person: Marko Grobelnik
E-mail: marko.grobelnik@ijs.si

University of Karlsruhe, Institute AIFB
Englerstr. 28
D-76128 Karlsruhe
Germany
Tel: +49 721 608 6592
Fax: +49 721 608 6580
Contact person: York Sure
E-mail: sure@aifb.uni-karlsruhe.de

University of Sheffield
Department of Computer Science
Regent Court, 211 Portobello St.
Sheffield S1 4DP
UK
Tel: +44 114 222 1891
Fax: +44 114 222 1810
Contact person: Hamish Cunningham
E-mail: hamish@dcs.shef.ac.uk

University of Innsbruck
Institute of Computer Science
Techikerstraße 13
6020 Innsbruck
Austria
Tel: +43 512 507 6475
Fax: +43 512 507 9872
Contact person: Jos de Bruijn
E-mail: jos.de-bruijn@deri.ie

Intelligent Software Components S.A.
Pedro de Valdivia, 10
28006
Madrid
Spain
Tel: +34 913 349 797
Fax: +49 34 913 349 799
Contact person: Richard Benjamins
E-mail: rbenjamins@isoco.com

Kea-pro GmbH
Tal
6464 Springen
Switzerland
Tel: +41 41 879 00
Fax: 41 41 879 00 13
Contact person: Tom Bösser
E-mail: tb@keapro.net

Ontoprise GmbH
Amalienbadstr. 36
76227 Karlsruhe
Germany
Tel: +49 721 50980912
Fax: +49 721 50980911
Contact person: Hans-Peter Schnurr
E-mail: schnurr@ontoprise.de

Sirma AI EAD, Ontotext Lab
135 Tsarigradsko Shose
Sofia 1784
Bulgaria
Tel: +359 2 9768 303, Fax: +359 2 9768 311
Contact person: Atanas Kiryakov
E-mail: naso@sirma.bg

Vrije Universiteit Amsterdam (VUA)
Department of Computer Sciences
De Boelelaan 1081a
1081 HV Amsterdam
The Netherlands
Tel: +31 20 444 7731, Fax: +31 84 221 4294
Contact person: Frank van Harmelen
E-mail: frank.van.harmelen@cs.vu.nl

Universitat Autonoma de Barcelona
Edifici B, Campus de la UAB
08193 Bellaterra (Cerdanyola del Vall` es)
Barcelona
Spain
Tel: +34 93 581 22 35, Fax: +34 93 581 29 88
Contact person: Pompeu Casanovas Romeu
E-mail: pompeu.casanovas@uab.es

Executive Summary

Although the Semantic Web and related technologies are usually focused on textual or structured data, it is also important to consider techniques for handling other sources of data. Of particular interest are multimedia data such as images and sound and video clips. In this report we present an overview of techniques that can be used to represent images as a first step towards further processing, for example for clustering, categorization, or retrieval. For instance, one can take the generated feature vector representation of images and apply some clustering or classification approach to construct/update a topic ontology of images.

Most of the image representation techniques focus on automatically extracting representations from the images themselves, rather than on using external sources of data such as textual descriptions or manual annotations provided by the user.

We describe a software module that can be used to automatically extract several types of representations from images or sets of images. The representations currently supported include histograms, autocorrelograms, and banded autocorrelograms. The RGB and YCC color spaces may be used, and the granularity of the color space quantization can also be customized. The module supports both textual XML-based output and binary output.

As an example of use of these representations, we present results of experiments in which these representations were used as a basis for image categorization. The experiments show that these approaches, in combination with machine learning techniques such as support vector machines, can achieve useful levels of classification accuracy on realistic collections of images. Autocorrelograms and banded autocorrelograms are found to perform better than histograms. Representations based on a finer-grained quantization of the color space are found to outperform those based on a coarser quantization.

The goal of this report is mainly to inform the other SEKT partners (including partners working on other technical issues, as well as the case study providers) on the possibilities of incorporating images into the process of semi-automatic ontology constructions. Nontechnical readers are recommended to focus on section 2, skipping the formulas and details if these are not of interest.

Contents

1. Introduction

This report deals with the problem of representing images in a way which enables efficient and effective manipulation of large pictorial databases, chiefly for the purposes of retrieval, categorization, and clustering. We describe various techniques for representing images and present a software module which can prepare some types of representations for a given set of images.

The report is arranged as follows. Section 2 presents an overview of related work in the fields of image retrieval and image categorization, showing the various approaches to image description, representation, matching, and querying that have been considered in the literature. Section 3 describes our software module for the automatic extraction of certain types of representations from collections of images. Section 4 presents some experiments which show how such representations can be used as a basis for image classification. We conclude with some suggestions for further work in this area in Section 5. User guide is provided in Appendix.

2. Related Work

The related work comes from the areas of image categorization and image retrieval, which have received a lot of attention in the recent years due to the proliferation of databases containing images and other sorts of multimedia data, and the resulting interest in approaches to managing and accessing such pictorial databases. Most work in these areas focuses on classifying images into a discrete set of categories (image categorization) and on finding, in a collection of images, those which correspond the most closely to the user's information need or query (image retrieval). Both problems share some of the underlying difficulties: the need to represent images on the one hand and queries and categories on the other, and the need for a matching function to determine the degree to which two representations (that of an image and that of a query or category) match.

2.1 Textual image representation

Various kinds of representations and matching functions have been considered in the literature. Sometimes textual representations are considered. They have the advantage that image retrieval and categorization can then be based on many well-known existing approaches from the field of information retrieval and text categorization. The downside of using textual descriptions of images is that preparing them manually is typically unacceptably time-consuming (particularly when dealing with large collections of images), and the resulting descriptions may be ambiguous as different people may describe an image in different ways (LEE *et al.*, 2004). Automatic extraction of textual descriptions from images is not yet possible, although some steps in that direction have been taken (WANG AND LI, 2002; JEON *et al.*, 2003; FAN *et al.*, 2004). On the other hand, in cases when reasonable textual descriptions can be obtained automatically from some existing source external to the images, a textual-based approach is often used in practice even though relatively less attention is paid to it in the literature. For example, the Google image search engine uses words from web pages and URLs to describe images which occur on those pages. Another example of using textual descriptions in image retrieval is the Chabot system (OGLE AND

STONEBAKER, 1995; CARSON AND OGLE, 1996); this is a hybrid approach combining textual descriptions and keywords with a few simple content-based features, which were obtained from the images automatically. Some authors have also considered augmenting unstructured textual descriptions into more structured representations, possibly involving some semantic; they have employed formalisms such as semantic frames, an is-a hierarchy of keywords to be used in the descriptions, etc. (QUINTANA, 1997, ASLANDOGAN *et al.*, 1996). Just like the plain textual representations, these approaches suffer from the need for large amounts of human attention when preparing the representations, often without corresponding improvements in e.g. retrieval effectiveness.

2.2 Content-based image representation

Therefore, the largest amount of attention has been devoted to the possibilities of representing images with descriptions induced automatically from the images themselves. The fact that these representations rely only on the contents of the image, rather than on some external source of data, has given rise to terms such as *content-based* image retrieval. In the remainder of this section we will consider various kinds of content-based image representations. For the purposes of our discussion, we will treat the image I as a matrix of H rows and W columns, with the pixel at the intersection of row y and column x denoted as $I(x, y)$.

2.2.1 Color space in image representation

Before we embark on the treatment of different approaches image representation, we should briefly consider the topic of *color spaces* and *color space quantization*. Typically, each color is described by a vector of three real values; the set of all colors then forms a *color space.* We can distinguish different color spaces based on the meaning of the three components of each color vector and their relationship to the color actually represented by the vector. Well-known color spaces include RGB (red, green, blue), HSB (hue, saturation, brightness), YCC (luminance and two chromaticity components), and so on. Not all color spaces are equally suitable for every purpose; in particular, for image categorization and retrieval it is often desirable if the color space is *perceptually uniform*, meaning that if we move by a certain distance in the color space, the change in color as perceived by the human eye should be roughly the same no matter in which part of the color space our starting point was located. Although perfect perceptual uniformity cannot be expected, some color spaces (e.g. YCC, Luv, and Lab) are closer to it than others. A very clear violation of this principle occurs in HSB, where a change in hue can cause the color to change into a completely different color if the brightness is large enough, but if the brightness is near 0 the color will be nearly black regardless of hue.

For further processing of an image, it is often desirable to *quantize* the color space, i.e. partition it into a discrete and usually fairly small set of regions such that all the colors from a particular region will be treated as undistinguishable. In effect this is the same as simplifying the image to employ only the colors from a fixed and limited palette. It is possible to select colors for the palette so as to represent a particular image or set of images particularly well, but this can make it more difficult to compare representations of different images if they rely on different color palettes. Thus a uniform quantization, which does not take the color characteristics of

individual images into account, is often preferred. A typical and practical form of quantization is to simply partition each axis of the color space into a set of intervals, often all equally wide. For example, consider the RGB color space, with each color represented by a triple (r, g, b), where r, g and b are real values from the range [0, 1). If we partition the r-, g- and b-axes into Q_r, Q_g and Q_b intervals, respectively, this quantization will result in a palette of $C = Q_r \cdot Q_g \cdot Q_b$ colors represented by the indices $0, \ldots, C-1$; the color (r, g, b) might be mapped into the palette index $\lfloor rQ_r \rfloor Q_g Q_b + \lfloor gQ_g \rfloor Q_b + \lfloor bQ_b \rfloor$.

2.2.2 Using histograms

One well-known representation of images is the *histogram*. Once the color space has been quantized, the histogram simply records for each color in the quantized color space, the proportion of the image that is covered by pixels of that color. Thus, the histogram is a C-dimensional vector $\mathbf{h} = (h_0, \ldots, h_{C-1})$, where $h_c = |\{(x, y) : I(x, y) = c\}| / (W \times H)$. Since each histogram is a simple C-dimensional vector, the degree to which two images are different can now be assessed by looking at the distance (e.g. Euclidean distance or Manhattan distance) between their histograms. Another possible measure is *histogram intersection* (SWAIN AND BALLARD, 1991) given two histograms $\mathbf{h}$ and $\mathbf{h}'$, the intersection is $\Sigma_{c=0..C-1} \min(h_i, h'_i)$. The downside of these distance measures is that they treat different colors c as orthogonal and independent of one another and ignore the fact that different colors from the palette can still look fairly similar. To remedy that, IBM's QBIC system (FALOUTSOS *et al.*, 1994) proposed taking the similarities between colors into account as well: their distance measure is $(\mathbf{h}-\mathbf{h}')^T A (\mathbf{h}-\mathbf{h}')$, where $A = (a_{c,c'})$ and $a_{c,c'}$ is the similarity between colors c and c' (Euclidean distance is a special case of this, obtained if A is the identity matrix). A still more sophisticated and dynamic distance measure has been proposed by GOH *et al.* (2002). DAVIDSON *et al.* (2001) describe an efficient indexing technique for querying a large collection of histograms. HE *et al.* (2004) describe how to map the images into a new space and define a distance measure there, taking into account an existing set of relevant images (e.g. supplied by the user during relevance feedback).

Apart from being very simple to compute, histograms have the desirable property of being fairly robust to many distortions, such as objects moving around the image or parts or small changes in the camera angle. They are more sensitive to the image becoming lighter or darker (especially if the quantization is rather fine-grained). Their main downside, however, is that they only record information about which colors are present on the image and to what extent, but say nothing about the distribution of colors within the image. For example, a hundred scattered red pixels have the same effect on the histogram as a 10×10 red square. To remedy this, various kinds of histogram refinement have been proposed, with the idea of distinguishing the pixels on basis of other characteristics besides color, and then computing the percentage of image covered by each group of pixels analogously to the original histogram. For example, one could count separately pixels near the center of the image and those near the edges; or one could count separately pixels which are part of a sufficiently large connected patch of that color, and separately those which aren't (the resulting representation is called *color coherence vectors* by PASS *et al.*, 1996; PASS AND ZADIH, 1999).

2.2.3 Using correlograms

Another potentially problematic aspect of histograms is that they regard each color separately and in isolation from the others; they give no information about local relations between different colors. To address this issue, *correlograms* have been proposed by HUANG *et al.* (1997). They defined a correlogram as a vector of probabilities

$$\gamma_{c,c',k} = P(I(x',y') = c' \mid I(x,y) = c \wedge \|(x,y) - (x',y')\|_\infty = k). \quad (1)$$

That is, $\gamma_{c,c',k}$ is the probability that a pixel, chosen randomly at distance k from a randomly chosen pixel of color c, will have the color c'. The above definition uses the max-norm $\|\cdot\|_\infty$ because of simpler implementation, although other distance measures could also be used. The problem with correlograms is that they consist of $C^2 \cdot K$ values if we have K different distances k and C colors in the quantized color space; even though the correlogram may be reasonably sparse, with many 0 values that need not be stored explicitly, it may still cause unacceptable time and space requirements when storing and matching the correlograms. Thus, an *autocorrelogram* is often preferable in practice. This is a subset of the correlogram containing only information about the correlations of each color with itself:

$$\alpha_{c,k} = \gamma_{c,c,k}.$$

The autocorrelogram is a vector of $C \cdot K$ values, meaning that it is only K times as time- and space-consuming as the histogram. To reduce these requirements still further, we can use the *banded autocorrelogram* (HUANG *et al.*, 1998), which is a "summary" of the autocorrelogram, obtained if we stop distinguishing between different distances k:

$$\beta_k = \Sigma_k \alpha_{c,k},$$

where the sum goes over all distances k used in the autocorrelogram. This results in a vector of the same dimensionality as the histogram.

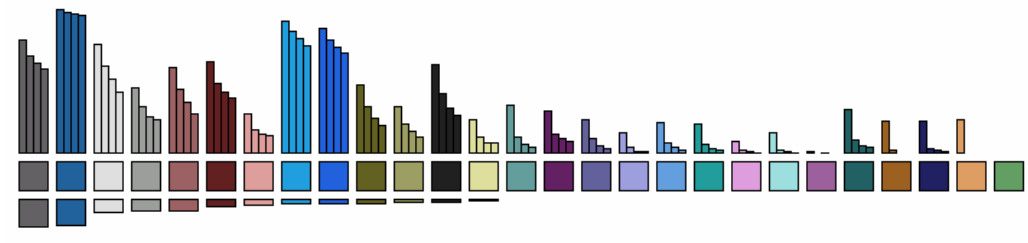

Figure 1. An image and its corresponding histogram and autocorrelogram, based on a 4×4×4 quantization of the RGB colorspace. Each of the boxes in the middle row shows the color on which the corresponding histogram and autocorrelogram entries are based. The bars in the top row are the autocorrelogram values, for the set of four distances $k \in \{1, 3, 5, 7\}$. The bars in the bottom row show the histogram (some bars are invisible because the color occurs on so few pixels). Note how the autocorrelogram enables us to distinguish between colors such as blue, which tend to occur in large patches (causing all entries of the autocorrelogram to be approximately equally large), and many other colors which occur in smaller isolated spots and consequently their autocorrelogram entry for distance k=1 is much higher than those for other distances.

2.2.4 Using color shape histograms

Another way of refining the simple histogram-based representation is to record separate histograms for different regions of the image. Our representation is now a matrix $H = (h_{i,c})$ where $h_{i,c}$ is the ratio of pixels of color c within region i to the total area of that region. The regions need not be disjoint; for example, STEHLING *et al.* (2000) suggested partitions the image into 3×3 equally large regions, as well as into 5×5 equally large regions, and recording the histograms of all 34 resulting regions, as well as of the image as a whole. If there are many regions, storage requirements may again be problematic here (similar to the case of (auto)correlograms), but the matrix H may be reasonably sparse, particularly if many colors c do not appear in the image at all. Each column of H gives the information about the distribution of a color around different parts of the image; Stehling *et al.* therefore refer to it as a *color shape histogram*.

Some authors have considered representing images by describing the shapes of entities that appear on the image, but for general-purpose collections of images it is difficult to reliably discover the meaningful and relevant entities and the resulting representation is not necessarily very useful for querying. PARK *et al.* (2000) proposed a histogram-like representation which basically records the percentage of edge pixels on various parts of the image, for several directions of edges. WANG AND MAKEDON (2003) used a histogram-like representation to describe the relative position of two objects on the image; this could be used to support querying based on the spatial relations between a reference object and other objects on the image. However, it is difficult to reliably and automatically identify the objects in an image.

2.2.5 Using texture

Image representation can also focus on *texture*, which is usually defined as a more or less repeating pattern on some part of the image. The problem of *texture segmentation*, i.e. how to partition the image into several regions such that the texture of each region is roughly homogeneous, has been attracting a lot of attention in the fields of pattern recognition and computer vision. Because of the supposed repetitiveness and periodicity of many textures, digital filtering techniques are sometimes used to detect them and segment the image accordingly; for example, a bank of Gábor filters may be used, each of which responds to a repetitive pattern with a particular frequency and orientation. Various *ad hoc* formulas are sometimes used instead to detect and describe textures (FALOUTSOS *et al.*, 1994).

Texture segmentation is sometimes based on clustering. For example, NATSEV *et al.* (1999) partitioned the entire image into small tiles of 4×4 pixels and represented each tile by a 12-dimensional vector based on the wavelet transform on that tile. The wavelet transform is appealing because it contains information about the average color and the frequency phenomena in the image on various scales, and such periodical phenomena often indicate the presence of a particular kind of texture. The vectors representing individual tiles can then be clustered and we can define one region for each cluster, containing the tiles whose vectors belong to that cluster. The centroid of the cluster can be used as a vector description of the corresponding region.

The number of regions found by texture segmentation typically varies considerably depending on the contents and complexity of the image. While it is in a way good that the representation can adapt to smaller or greater complexity of images, it also means that representations of different images can no longer be compared using such simple measures as the Euclidean or Manhattan distance, which can be used on vector-based representations such as histograms and autocorrelograms. Instead, various similarity functions have been proposed that work on segmented images. To determine how similar two images are, similarities between individual regions of the two images are usually computed first and then somehow combined into an overall similarity measure; the size and location of regions can also be taken into account, giving larger regions and regions closer to the centre of the image greater influence on the overall similarity measure. An example of this type of approach is the *integrated region matching* of WANG *et al.* (2000). Texture segmentation has also been used by CHEN *et al.* (2001), who represented each region by a feature vector and the image by a set of such feature vectors; they then replaced these sets by fuzzy sets and used a fuzzy similarity measure to measure similarity between images for the purposes of retrieval.

Many representations are based on keeping some of the coefficients of a wavelet transform of the image, or parts of the image. The wavelet transform can be computed efficiently and results in a frequency analysis of the image on a wide range of scales. Examples of using wavelets for image representation include JACOBS *et al.* (1995), WANG *et al.* (1997), LIU AND MANDAL (2001), HOI AND LYU (2004).
An approach that combines histograms and texture has been used by OBEID *et al.* (2001). Using a set of training images, they defined six types of texture and represented each of them by a histogram. When processing a new image, each pixel is considered to be a "representative" of that texture in which pixels of this color are the most likely to appear (based on the histograms obtained on the training set). If we record the number of representative pixels for each of the six textures, we have represented our new image by a 6-tuple which can be used for retrieval or classification. This approach is an example of an intermediate path, slightly above the purely low level features such as simple histograms, but below the high-level textual or semantics-oriented descriptions which often require too much manual involvement.

2.3 Query representation

Another interesting issue in image retrieval (and related to the question of how to represent images) is how the user can describe his or her query to the system. For systems which include or are based on textual representations of images, the query can simply be a set of keywords (this is e.g. the approach taken by the Google image search engine). Most authors in the field of image retrieval assume that the query is represented by a query image, which the user has supplied and is now interested in retrieving similar images from the database. Sometimes the user can draw a sketch instead of supplying a full query image (JACOBS *et al.*, 1995; DI SCIASCIO AND MONGIELLO, 1999). Sometimes it is difficult for the user to express his or her information needs in the form of a single query image; more interactive approaches may be used in such cases, often involving relevance feedback and active learning (DI SCIASCIO AND MONGIELLO, 1999; CHANG *et al.*, 2001; HE *et al.*, 2004). HOI AND LYU (2004) proposed an approach to relevance feedback which utilizes relevance judgments gathered during previous retrieval sessions. LONG AND LEOW (2001) used

relevance feedback to define a more "perceptually consistent" distance measure on images.

3. Architecture and Approach

We have developed an approach to representing image with a set of features that can be further used in image retrieval, classification, clustering, etc. Our system processes a set of image files, computes image descriptions of various kinds and outputs them in XML format or binary format (see Figure 2). In our approach, representation based on a quantized color space is used, more precisely histograms, autocorrelograms and banded autocorrelograms. The program is controlled via command-line parameters, which will be described in the following subsections.

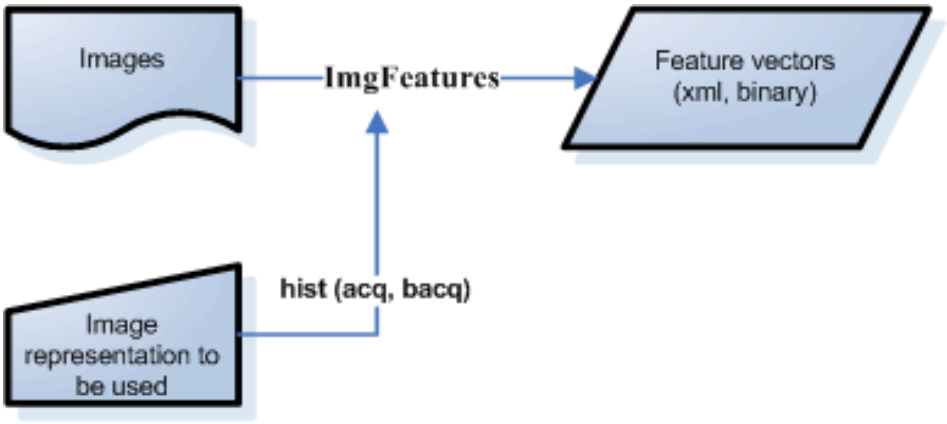

Figure 2. Architecture of the system for representing images as feature vectors based on a quantized color space using histograms (hist), autocorrelograms (acq), or banded autocorrelograms (bacq).

3.1 Input Files

The input files are specified via the "`-i:`⟨*file name*⟩" command-line parameter. This parameter may occur more than once if several image files need to be processed. The file name may also contain a path specification.

If the file name contains the wildcard * and ? characters (the asterisk matches zero or more characters, the question mark matches any single character), all file names in that directory will be examined and files whose names match the wildcard will be processed; if no directory is specified in the `-i` parameter, the current directory will be used. If the "`-recurse`" parameter is also given on the command line, the subdirectories will also be searched recursively for filenames matching the wildcard.

The "`-i`" parameter also allows the special form "`-i:@`⟨*file name*⟩", in which case the file name is assumed to refer to a plain-text file containing a list of image file names which should be processed (one name per line).

Currently the input files must be in BMP format with 24 bits per pixel.

3.2 Representation Types

Exactly one of the following command-line parameters must be present to specify what kind of image representation should be prepared: "`-hist`" (histograms), "`-acg`" (autocorrelograms), "`-bacg`" (banded autocorrelograms).

All these representation types are based on a quantized color space. By default, the underlying color space used is RGB, but the "`-ycc`" parameter can be used to convert the images to YCC before quantization and computation of the representations. The quantization granularity must be specified via the "`-gran:`Q_1`,`Q_2`,`Q_3" parameter. This causes the first axis to be divided into Q_1 equally long intervals, the second axis into Q_2 intervals and the third axis into Q_3 intervals. This quantization maps the color (x_1, x_2, x_3), where $0 \le x_1, x_2, x_3 < 1$, into the palette color index

$$\lfloor x_1 Q_1 \rfloor Q_2 Q_3 + \lfloor x_2 Q_2 \rfloor Q_3 + \lfloor x_3 Q_3 \rfloor$$

from the range $0, \ldots, (Q_1 \cdot Q_2 \cdot Q_3 - 1)$.

For autocorrelograms and banded autocorrelograms, the set of distances involved in computing the autocorrelograms (the values of k in equation (1)) must also be specified, using the "`-d:`k_1`,`k_2`,...,`k_K" parameter (where K is the number of different distances to be used in the autocorrelogram or banded autocorrelogram). Because the intention of the autocorrelogram is to capture information about local spatial co-occurrence of colors, the values of k to be used are usually small. For example, HUANG *et al.* (1997) used the values 1, 3, 5, and 7.

3.3 Output Types

Output files must be specified via the "-o" parameter. Two output formats are currently supported: "`-oxml:`⟨*file name*⟩" for XML-based output and "`-obin:`⟨*file name*⟩" for binary output. At least one of these parameters is required, but both may also be present to request that both types of output be produced.

The XML-based output has the following structure:

```
<ImageReprSet version="ImageReprSet Version 1">
  <ImageReprDesc version="1" type="histogram"
                 underlyingColorSpace="RGB"
                 gran1="16" gran2="8" gran3="8" />
  <ImageRepr fileName="foo.bmp"
             relPath="..\a\foo.bmp" absPath="c:\bar\a\foo.bmp">
     0.103448 0.00361867 ... 0.055056</ImageRepr>
  <ImageRepr ...>...</ImageRepr>
  <ImageRepr ...>...</ImageRepr>
<ImageReprSet/>
```

The `type` attribute can also have the values "`autocorrelogram`" and "`banded-autocorrelogram`". The `underlyingColorSpace` can also have the value "`YCC`". The `gran1`, `gran2`, and `gran3` attributes contain the quantization granularity values that were specified via the "`-gran`" command-line parameter. For autocorrelograms and banded autocorrelograms, the `ImageReprDesc` element has another attribute `ds="`k_1`,`k_2`,...,`k_K`"` listing the distances that were provided via

the "`-d`" command-line parameter and used to compute the autocorrelogram or banded autocorrelogram.

Each `ImageRepr` element contains the representation of one image. The attributes provide the file name of the original give the file name (without path) of the original image file and the relative and absolute path to the image file. The contents of the element consist of the components of the vector representation of that image. The components are listed in increasing order of color palette index. For a histogram or a banded autocorrelogram, there is one component per color. For a plain autocorrelogram, there are *K* components per color, one for each distance provided in the "`-d`" command-line parameter; these components appear in the same order as the distances in the "`-d`" parameter.

4. Evaluating the system on image categorization

In this section we present the results of some experiments in which the image representations described in the previous sections were used for the purpose of image categorization.

For the evaluation, we used a subset of the *misc* database which has already been used in the image retrieval literature, e.g. by WANG *et al.* (1997) and NATSEV *et al.* (1999). The full database (available from *http://www-db.stanford.edu/IMAGE/*) consists of 9907 small photographic images, mostly of 128×85 pixels. There is a wide variety in terms of content. Since no preexisting categorization scheme or relevance judgments were available for this dataset, we manually selected a subset of the images and classified them into 14 categories: flags, butterflies, sunset, autumn, flowers, planets, satellite images of Earth, cars, mountains, clouds, sea, surfing, sailboats, animals. This selection of classes is intended to contain some distinct and easily recognizable ones (such as sunsets) and some groups of classes which are harder to distinguish (e.g. mountains, clouds, and sea, all of which contain a large amount of blue and white color hues). Our resulting dataset consists of 1172 images, each of which belongs to one of the above-mentioned 14 classes.

We compared several kinds of image representations: histograms, autocorrelograms, and banded autocorrelograms. The latter two were based on the set of distances {1, 3, 5, 7}, as suggested in HUANG *et al.* (1997). We used the RGB color space, with several different quantizations: 4×4×4, 6×6×6, and 8×8×4 (the blue axis is sometimes divided into fewer ranges than the other two because the human eye is often less sensitive to small changes of color along the blue axis than along the other two axes). The resulting image representations were multidimensional vectors of varying length (from 64 to 1024 dimensions). We used them as input for several learning and classification methods: nearest neighbors based on Manhattan distance, nearest neighbors based on Euclidean distance, and support vector machines (SVMs) (CORTES AND VAPNIK, 1995; BURGES, 1998) with either linear, cubic, or RBF (radial basis function) kernels. We performed ten-fold cross-validation and report the average classification accuracy (i.e. the percentage of images from the test set that were classified into the correct class). Since SVMs were originally designed for binary classification problems while our problem is multiclass (but only one class may be

predicted), we trained one binary classifier for each pair of classes and used the predictions of the resulting model as votes for one or the other class.

The results of these experiments are shown on the charts on Figure 3, Figure 4 and Figure 5. Each of the three charts contains the results for one of the three different colorspace quantizations. The columns labeled "NN, L1" and "NN, L2" refer to the performance of the 1-nearest-neighbor based on the Manhattan and Euclidean distances, respectively. The other columns refer to the performance of support vector machines with the three different kinds of kernel (linear, cubic, and RBF).

Comparing the different classification algorithms, we observe that SVM significantly outperforms the nearest-neighbor classifier; for the latter, Manhattan distance works better than Euclidean distance, which confirms the observations of earlier studies (e.g. HUANG *et al.*, 1997). Of the different kernels, there is no significant difference in performance between cubic and RBF kernels, while both perform better than linear kernels.

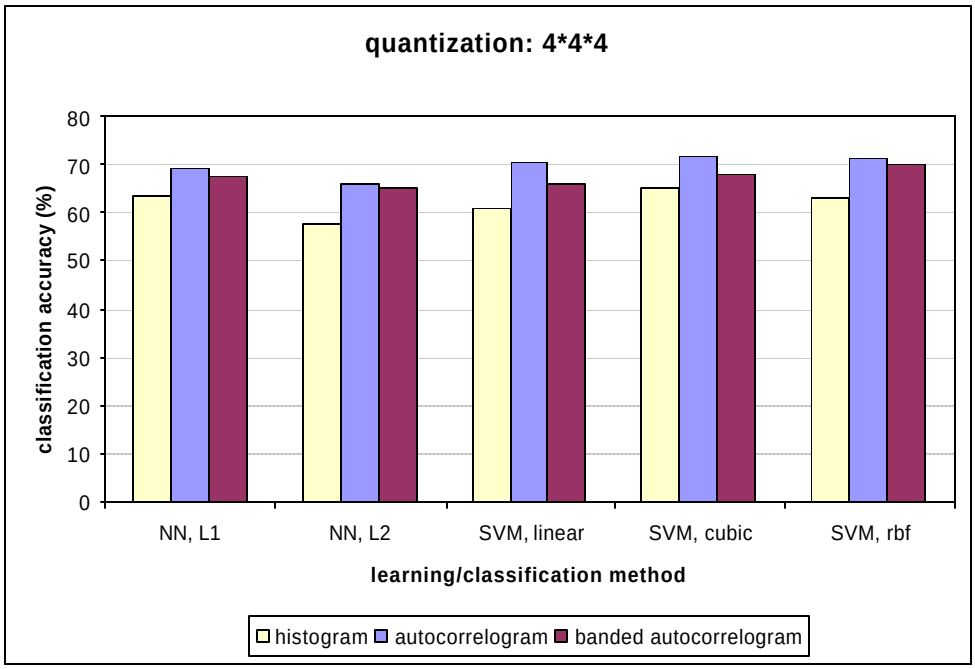

Figure 3. Results of the experiments comparing different combinations of image representation, and classification method using colorspace quantization: 4x4x4.

Regarding quantization, we observe that finer-grained quantizations performed better. There are no significant differences between 6×6×6 and 8×8×4, but both significantly outperform the coarser 4×4×4 quantization. The differences were particularly large for linear SVM, because with other kernels the greater flexibility of the kernel may make up for the smaller dimensionality caused by the coarser quantization.

Finally, if we compare different representation types, we notice that there are usually (except in the case of linear SVM) no significant differences in performance of autocorrelograms and banded autocorrelograms (but where there is a difference,

banded autocorrelograms perform worse than plain autocorrelograms); on the other hand, both of them significantly outperform the simple histograms. This makes banded autocorrelograms a particularly appealing representation type, because they combine the lower memory requirements of histograms with the better performance of autocorrelograms.

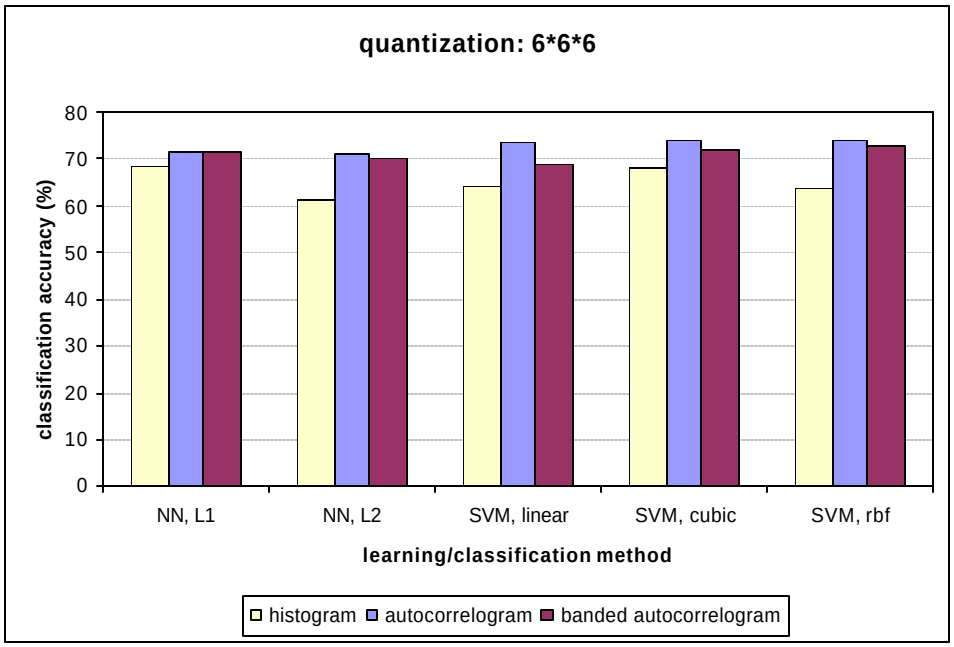

Figure 4. Results of the experiments comparing different combinations of image representation, and classification method using colorspace quantization: 6x6x6.

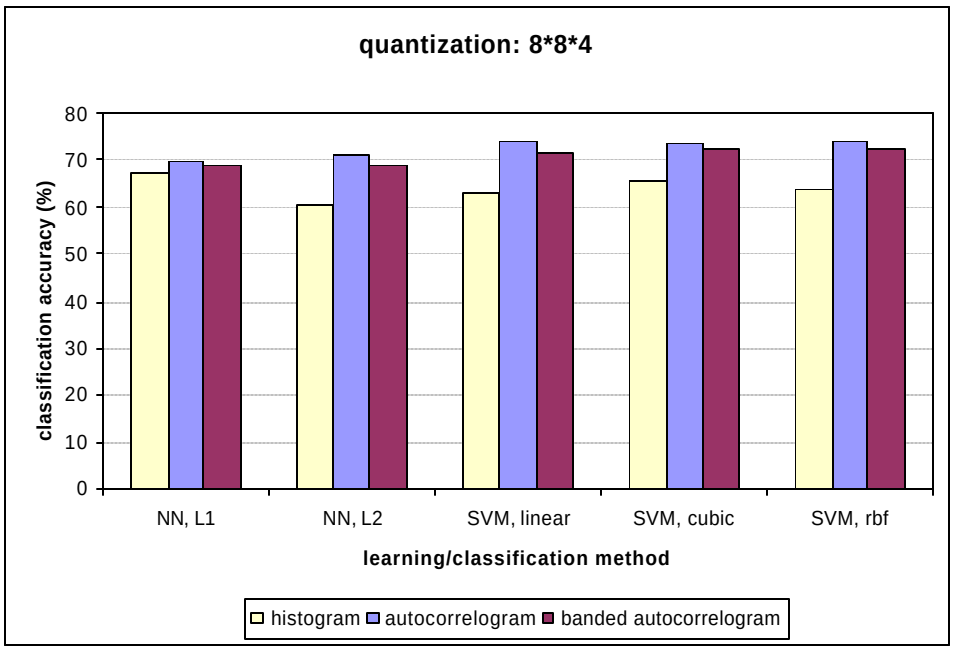

Figure 5. Results of the experiments comparing different combinations of image representation, and classification method using colorspace quantization: 8×8×4.

5. Future Work

The feature generation approach and software described in this report could be extended in many ways. Support for additional image file formats could be added, as well as for converting the images to different color spaces before quantization and generation of representations. Quantization is currently done by dividing each axis of the color space into several equally wide intervals; other types of quantization could be considered.

Many other approaches to image representation and extraction of interesting and useful features have also been described in the literature (see Section 2 for some examples). However, they are often oriented specifically at a particular task or a particular type of images and would therefore only be suitable for more narrowly specialized applications. More complex representations of images often rely on specific and complex matching functions and other algorithms that need to be employed to make full use of the representation. For example, instead of basing all histograms on a fixed quantization of the color space, a different quantization might be used for each image, adapted to the contents of that image; but this would make it impossible to compare histograms as vectors on a component by component basis, and a more complex matching function or distance measure would need to be introduced instead. Integrating richer representations into the general TextGarden framework (see SEKT Deliverable 1.5.1) could therefore be problematic and is left as the subject of future work.

Another interesting topic of further work is tighter integration of pictorial data with other kinds of data, particularly text but also traditional attribute-based or relational data. In some applications, images could be described by a combination of a textual

description and a content-derived representation such as a histogram or autocorrelogram (LU AND DREW, 2001). Various methods could be used to work with such combined representations; for example, categorization could be done using co-training, classifier combination methods such as stacking, or by using kernel methods and combining several kernels based on different representations of an image.

In the line of using the output of the system as input into other programs we have in mind the problem of constructing ontology of images based on their content. For that purpose, one would take the generated feature vectors and apply some clustering or classification approach to construct and update topic ontology of images. We are investigating possibilities of connecting the described utility with some of the other utilities that are developed in the same TextGarden library. This work will be prioritized and shaped based on the needs of the SEKT project case studies.

Appendix - User Guide

The utility extracts various groups of features from input images ("-i"). It can output the resulting image descriptions either in XML form ("-oxml") or in a binary format ("-obin"). In both cases the output also includes a description of the representation used.

One or more **input file** parameters can be specified. Each can have the following form:

- "-i:*FileName*" to process a specific image file.
  If the *FileName* includes the wildcard characters "?" and "*", all files matching the wildcard will be processed. *FileName* may also contain a path/drive specification, which should not include wildcard characters. The "-recurse" parameter may be used to look for files matching the wildcard in subdirectories as well.
- "-i:*@FileName*" to process all the image files whose names are listed, one per line, in the text file *FileName.*

The input files should be in BMP format, with 24 bits per pixel.

All image representations currently supported by the program are based on a quantization of the color space into a limited set of colors. The "-gran:Q_r,Q_g,Q_b" parameter should be used to specify the **granularity** of the quantization. Each axis of the color space is divided into Q_* sections. Thus, if a pixel originally had the RGB color

$$(r, g, b), \text{ with } 0 <= r, g, b < 1,$$

it will be represented by the color index

$$floor(r\ Q_r) * Q_g\ Q_b + floor(g\ Q_g) * Q_b + floor(b\ Q_b).$$

The resulting quantization has $n := Q_r * Q_g * Q_b$ colors. This directly affects the number of features in the resulting image representations, all of which are based on providing one or more numerical values for each color of the quantized color space.

As an alternative to the RGB color space in which the input images are originally stored, the "-ycc" parameter can be used to transform the images to the YCC color space before further processing. In this case the quantization will also be applied in

the YCC rather than in the RGB space; the three values listed with the "-gran" parameter refer to the quantization granularity for the Y (luminance), C_1 (chromaticity 1) and C_2 (chromaticity 2) axes, respectively. Using the YCC colorspace instead of RGB may be desirable because of YCC's greater perceptual uniformity (i.e. how well the distance between 3-D vectors representing colors corresponds to the human perception of the difference between the colors).

Three types of image representations are supported at the moment:

- A **histogram** ("-hist") is a vector of n values $h_0, h_1, \ldots, h_{n-1}$, where h_i is the proportion of the image covered by pixels belonging to color index i (in the quantized color space). All these values are in the range 0 to 1.
- An **autocorrelogram** ("-acg") is a vector of $n * k$ values $a_{i,j}$ for $i = 0, \ldots, n-1$ and $j = 0, \ldots, k-1$, arranged in increasing order of i and for each i in increasing order of j. The value $a_{i,j}$ is the probability, given a randomly chosen pixel with color i, that a pixel chosen randomly at a distance d_j from the first pixel also has color i. For the purposes of this definition, distance is measured using the max-norm. The distances d_j must be specified via the "-d:d_0,d_1,...,d_{k-1}" parameter.
- A **banded autocorrelogram** ("-bacg") is a vector of n values b_i, defined as $b_i = a_{i,0} + \ldots + a_{i,k-1}$. As with the autocorrelograms, the "-d:..." parameter must be provided to specify the distances d_j.

Two types of output are supported. At least one of the corresponding parameters must be present on the command line.

- **XML output** ("-oxml:*FileName*") has the following form:

```
<ImageReprSet version="ImageReprSet Version 1">
    <ImageReprDesc version="1" type="histogram"
        underlyingColorSpace="RGB"
        gran1="16" gran2="8" gran3="8" />
    <ImageRepr fileName="foo.bmp" relPath="..\a\foo.bmp"
        absPath="c:\bar\a\foo.bmp">0.103448 0.00361867 ...
        0.055056</ImageRepr>
    <ImageRepr ...>...</ImageRepr>
<ImageReprSet/>
```

  The `gran*` attributes report the granularity values Q_r, Q_g, Q_b, respectively. The `type` attribute can also have the values `autocorrelogram` and `banded-autocorrelogram`; in these cases, the `ImageReprDesc` element will also have a `ds="`$d_0 d_1 \ldots d_{k-1}$`"` attribute describing the distances used in the (banded) autocorrelogram.

  The following is a document type definition for the XML output of this program:

```
<!ELEMENT ImageReprSet (ImageReprDesc, ImageRepr*)>
<!ATTLIST ImageReprSet
      version CDATA #REQUIRED>
<!ELEMENT ImageReprDesc EMPTY>
<!ATTLIST ImageReprDesc
      version CDATA #FIXED "1"
```

```
        type (histogram | autocorrelogram |
              banded-autocorrelogram) #REQUIRED
        underlyingColorSpace (RGB | YCC) #REQUIRED
        gran1 CDATA #IMPLIED
        gran2 CDATA #IMPLIED
        gran3 CDATA #IMPLIED
        ds CDATA #IMPLIED>
<!ELEMENT ImageRepr (#PCDATA)>
<!ATTLIST ImageRepr
        fileName CDATA ""
        relPath CDATA ""
        absPath CDATA "">
```

- **Binary output** ("-obin:*FileName*") is based on the standard Text-Garden serialization mechanisms. The `TImageReprSet::Save` method is called to serialize the image representations into a binary form. See the source code for details.

Usage: ImgFeatures.exe
-i:Input-FileName
-i:Input-FileMask (containing * and/or ? characters)
-i:@File-Containing-List-Of-Input-FileNames
-oxml:Xml-Output-FileName
-obin:Binary-Output-FileName
-ycc (optional, to convert image to YCC colorspace)
-hist -acg -bacg
-gran:Q1,Q2,Q3
-ds:Comma-Separated-List-Of-Distances (only for -acg/-bacg)
-recurse (optional, to search subdirectories for files matching the Input-FileMask(s))

Example 1:
ImgFeatures.exe -i:foo.bmp -i:bar.bmp -oxml:baz.xml -hist -gran:8,8,4

This reads the files *foo.bmp* and *bar.bmp* and computes for each a histogram based on a quantization of the RGB color space into 8 * 8 * 4 = 256 colors (the red and green axes are divided into eight intervals each, and the blue axis into four). The two histograms are written in XML form into *baz.xml*.

Example 2:
ImgFeatures.exe -i:c:\data\foo*.bmp -obin:foo.bin -acg -gran:6,6,6 -ds:1,3,5,7

This searches the *c:\data* directory for all files whose name matches *foo*.bmp*. An autocorrelogram is computed for each of these images, based on four distances (1, 3, 5, 7) and a quantization of the RGB color space into 6 * 6 * 6 = 216 colors (each axis divided into six segments). Thus the autocorrelogram of each image consists of 864 components.. The autocorrelograms are stored in binary form into *foo.bin*.

Example 3:
ImgFeatures.exe -i:@bar.txt -i:c:\data\foo*.bmp -recurse -oxml:d:\foo.xml -obin:c:\foo2.bin -bacg -gran:6,6,6 -ds:1,3,5,7

This processes all files whose names are listed in *bar.txt* (one per line). It also

searches the *c:\data* directory and subdirectories recursively for all files whose name matches *foo*.bmp* . Banded autocorrelograms are computed for all these images and stored in *d:\foo.xml* and in *c:\foo2.bin* .

Bibliography and References

1. ASLANDOGAN, Y. A., THEIR, C., YU, C., A system for effective content-based image retrieval. Proc. 4th ACM Conf. on Multimedia, pp. 429–430 (1996).
2. BURGES, C. J. C., A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2):121–167 (1998).
3. CARSON, C., OGLE, V. E., Storage and retrieval of feature data for a very large online image collection. IEEE Bulletin of the Technical Committee on Data Engineering, 19(4):19–27 (1996).
4. CHANG, E., CHENG, K.-T., LAI, W.-C., WU, C.-T., CHANG, C., WU, Y.-L., PBIR: Perception-based image retrieval — a system that can quickly capture subjective image query concepts. Proc. ACM Multimedia Conf., pp. 611–4 (2001).
5. CHEN, Y., WANG, J. Z., LI, J., FIRM: Fuzzily integrated region matching for content-based image retrieval. Proc. ACM Multimedia Conf., pp. 543–5 (2001).
6. CORTES, C., VAPNIK, V. N., Support-vector networks. Machine Learning, 20(3):273–297 (1995).
7. DAVIDSON, A., ANVIK, J., NASCIMENTO, M. A., Parallel traversal of signature trees for fast CBIR. Proc. ACM Workshops on Multimedia Information Retrieval, pp. 6–9 (2001).
8. DI SCIASCIO, E., MONGIELLO, M., Query by sketch and relevance feedback for content-based image retrieval over the web. Journal of Visual Languages and Computing, 10(6):565-584 (1999).
9. FALOUTSOS, C., EQUITZ, W., FLICKNER, M., NIBLACK, N., PETKOVIC, D., BARBER, R., Efficient and effective querying by image content. Journal of Intelligent Information Systems, 3:231–262 (1994).
10. FAN, J., GAO, Y., LUO, H., XU, G., Automatic image annotation by using concept-sensitive salient objects for image content representation. Proc. ACM SIGIR Conf., pp. 361–368 (2004).
11. GOH, K.-S., LI, B., CHANG, E., DynDex: A dynamic and non-metric space indexer. Proc. ACM Multimedia Conf., pp. 466–475 (2002).
12. HE, J., LI, M., ZHANG, H.-J., TONG, H., ZHANG, C., Manifold-ranking based image retrieval. Proc. ACM Multimedia Conf., pp. 9–16 (2004).
13. HE, X., MA, W.-Y., ZHANG, H.-J., Learning an image manifold for retrieval. Proc. ACM Multimedia Conf., pp. 17–23 (2004).
14. HONG, C.-H., LYU, M. R., A novel log-based relevance feedback technique in content-based image retrieval. Proc. ACM Multimedia Conf., pp. 24–31 (2004).
15. HUANG, J., RAVI KUMAR, S., MITRA, M., Combining supervised learning with color correlograms for content-based image retrieval. Proc. 5th ACM Int. Conf. on Multimedia, pp. 325–334 (1997).
16. HUANG, J., RAVI KUMAR, S., ZABIH, R., An automatic hierarchical image classification scheme. Proc. 6th ACM Int. Conf. on Multimedia, pp. 219–228 (1998).
17. JACOBS, C. E., FINKELSTEIN, A., SALESIN, D. H., Fast multiresolution image querying. Proc. 22nd ACM SIGGRAPH Conf., pp. 277–286 (1995).

18. JEON, J., LAVRENKO, V., MANMATHA, R., Automatic image annotation and retrieval using cross-media relevance models. Proc. ACM SIGIR Conf., pp. 119–126 (2003).
19. LEE, C.-Y., SOO, V.-W., FU, Y.-T., How to annotate an image? The need of an image annotation agent. Proc. 4th Joint ACM/IEEE Conf. on Digital Libraries, p. 394 (2004).
20. LIU, C., MANDAL, M., Fast image indexing based on JPEG2000 packet header. Proc. ACM Conf. on Multimedia Information Retrieval, pp. 46–49 (2001).
21. LONG, H., LEOW, W. K., Perceptual consistency improves image retrieval performance. Proc. ACM SIGIR Conf., pp. 434–5 (2001).
22. LU, C., DREW, M. S., Construction of a hierarchical classifier schema using a combination of text-based and image-based approaches. Proc. ACM SIGIR Conf., pp. 438–9 (2001).
23. NATSEV, A., RASTOGI, R., SHIM, K., WALRUS: A similarity retrieval algorithm for image databases. Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 395–406 (1999).
24. OBEID, M., JEDYNAK, B., DAOUDI, M., Image indexing and retrieval based on intermediate features. Proc. ACM Workshops on Multimedia Information Retrieval, pp. 531–3 (2001).
25. OGLE, V. E., STONEBRAKER, M., Chabot: Retrieval from a relational database of images. IEEE Computer, 28(9):40–48 (1995).
26. PASS, G., ZABIH, R., Histogram refinement for content-based image retrieval. Proc. IEEE Workshop on Applications of Computer Vision, pp. 96–102 (1996).
27. PASS, G., ZABIH, R., Comparing images using joint histograms. Journal of Multimedia Systems, 7(3):234–240 (1999).
28. PARK, D. K., JEON, Y. S., WON, C. S., PARK, S.-J., Efficient use of local edge histogram descriptor. Proc. ACM Multimedia Workshop, pp. 51–54 (2000).
29. QUINTANA, Y., Organization and retrieval of a pictorial digital library. Proc. 2nd ACM Int. Conf. on Digital Libraries, pp. 13–20 (1997).
30. STEHLING, R. O., NASCIMENTO, M. A., FALCÃO, A. X., On "shapes" of colors for content-based image retrieval. Proc. ACM Multimedia Workshop, pp. 171–174 (2000).
31. SWAIN, M. J., BALLARD, D. H., Color indexing. Int. Journal of Computer Vision, 7(11):11–32 (1991).
32. WANG, J. Z., LI, J., Learning-based linguistic indexing of pictures with 2-D HMMs. Proc. ACM Multimedia Conf., pp. 436–445 (2002).
33. WANG, J. Z., LI, J., WIEDERHOLD, G., SIMPLIcity: Semantics-sensitive integrated matching for picture libraries. Proc. 4th Int. Conf. on Advances in Visual Information Systems, LNCS 1929, pp. 360–371 (2000).
34. WANG, Y., MAKEDON, F., R-histogram: Quantitative representation of spatial relations for similarity-based image retrieval Proc. ACM Multimedia Conf., pp. 323–326 (2003).
35. WANG, J. Z., WIEDERHOLD, G., FIRSCHEIN, O., WEI, S. X., Content-based image indexing and searching using Daubechies' wavelets. Int. Journal of Digital Libraries, 1(4):311–328 (1997).