Using Image Segmentation as a Basis for Categorization

Janez Brank
Department of Intelligent Systems
Jožef Stefan Institute, Ljubljana, Slovenia
janez.brank@ijs.si

Keywords: image categorization, segmentation, generalized kernels

Received: June 26, 2002

Image categorization is the problem of classifying images into one or more of several possible categories or classes, which are defined in advance. Classifiers can be trained using machine learning algorithms, but existing machine learning algorithms cannot work with images directly. This leads to the need for a suitable way of representing or describing images such that learning algorithms can work with them. We consider a representation based on texture segmentation and a similarity measure between segmented images which has been used successfully in the related area of image retrieval. A generalized kernel for use with the support vector machine (SVM) algorithm can be built from such a similarity measure. We compare this approach with a more straightforward representation based on autocorrelograms, and we show that these two representations can be combined to obtain classifiers with higher categorization accuracy.

1 Introduction

Besides textual and relational data, people increasingly have to deal with pictorial data, or data in the form of images. Large *pictorial databases* are being produced as archives digitize their collections, and additionally the World Wide Web contains a huge number of images. Apart from purely technical problems of storing and processing such large amounts of data, the emergence of large collections of images opens the problems of enabling the users to make sense of this data and find what they need. *Image categorization* deals with one aspect of this problem: given a set of images and a set of predefined categories or classes, we assume that each image should belong to one or possibly several of these categories. For a large collection it would be impractical to have a human observer categorize all the images, so we want to be able to classify the images automatically after a small number of images has been classified manually to be used for training the automatical classifiers.

However, this view of image categorization as a machine learning task immediately opens up a new problem: existing machine learning algorithms generally cannot work with images directly. Instead, they often assume they will be dealing with instances described by vectors or tuples. We need to be able to represent images using structures of this kind to make use of existing machine learning algorithms.

1.1 Related work in image retrieval

We can build on existing work in *image retrieval*, which is a related area where the problem of representation has already been encountered. In image retrieval, the user poses a query to the system and the system should find images that are somehow relevant to the query. Thus a way of representing the query, a way of representing images, and a way of comparing a query and an image (to determine if the image is relevant with regard to this query) are needed. One approach that is both technically feasible and useful enough to be commonly used in practice (e.g. in web image search engines such as Google) is to describe each image using a few keywords, and the user's query can then request images whose description includes or excludes particular keywords. However, this approach is only feasible if textual descriptions of images can be obtained automatically (e.g. from the HTML file that linked to an image); it is usually too costly to have a human maintainer prepare such descriptions manually for a larger database. In addition, this textual approach suffers from problems of polysemy: different people would use different words to describe an image, and the same words may mean different things to different people. Therefore it is often desirable to rely solely on what can be automatically extracted from the images themselves. The user's query is then often simply a request to look for images similar to a given query image or sketch (this approach is known as "querying by content", or "content-based image retrieval").

There are several close parallels between image retrieval and image categorization. In categorization, if a new image is similar to training images from a particular category, it should probably itself belong to that category; in content-based image retrieval, if an image from the database is similar to the query image, it should probably be shown to the user. Thus we see that both areas need a way of representing images and assessing similarity between them. Many image representations and similarity measures have been proposed in image

retrieval, and we would like to examine some of them from the point of view of image categorization as well.

One popular class of image representations is based on simplifying the image by approximating the color of each pixel by the nearest color from a predefined and fixed color palette; this can also be seen as partitioning (or *quantizing*) the space of all possible colors. Some information is then recorded about the presence of each color on the image. When simply the proportion of the image covered by (the pixels of) that color is stored, the resulting description is called a *histogram* [11]. However, this disregards all spatial information (how the color is distributed around the image): for example, a large patch of red would affect the histogram of an image in the same way as a large number of red pixels scattered all over the image, which is surely undesirable.

Several improved histogram-like representations of images have been proposed. For example, an *autocorrelogram* [4] records, for each color c and for a few small integers d, the probability that a pixel, chosen randomly at distance d from a randomly chosen pixel of color c, will itself be of the color c. This retains information about the amount of a color present on the image, but also records something about the spatial arrangement of each color. Still, all "global" representations of this type can be seen as somewhat rigid as they record a strictly fixed amount of data for each image. They cannot take into account the fact that some images are more complex than others, that an image may contain several objects, or that it may be helpful to distinguish between an (interesting) object and (uninteresting) background.

1.2 Image segmentation

Another, more sophisticated, class of image representations is based on *segmentation*, or dividing an image into a set of *regions* such that each region is roughly homogeneous in color and/or texture. Each image is then represented by a set of regions; each region is typically described by a short vector that is a by-product of the segmentation procedure (containing e.g. the average color of the region, information about texture, and so on). Additionally, the location of each region on the image (i.e. which parts of the image are covered by that region) is often recorded as well. In general, regions might overlap, and each region might itself be composed of several disjoint parts; this is not necessarily problematic as they need not be shown to the user, and image similarity measures usually permit the regions to be disconnected, and sometimes work with overlapping regions as well. Representations based on segmentation can adapt well to differences in complexity between images, and have been used successfully in image retrieval [NRS99, WLW00].

Various segmentation algorithms have been proposed in the context of image retrieval [NRS99, WLW00]. These approaches are usually based on dividing the image into a grid of small "windows" (e.g. 4×4 pixels); each window is described by a short vector (containing e.g. the average color and possibly a few coefficients from the higher-frequency bands of a wavelet transform, in order to capture the presence of edges or texture), and these vectors are then clustered. Each of the resulting clusters contains vectors that lie close together in their vector space, and such vectors hopefully correspond to windows that are similar in appearance; therefore it makes sense to form a region from such a group of windows. The region thus obtained can be described by the centroid of the cluster, i.e. by the average of the vectors that describe the windows from which the region was formed.

To use segmentation for image retrieval, it is also necessary to introduce a measure of similarity between segmented images. Such measures usually examine pairs of individual regions (one region from each image) and combine the measures of similarity or difference between regions into a single similarity measure between entire images. For example, the *integrated region matching* (IRM) measure [8] defines the distance between two images as a weighted sum of distances between regions, in which the weights are chosen so as to allow larger regions to have a larger influence on the similarity between images.

To use the representations described above for image categorization, one could use global representations (e.g. autocorrelograms) in combination with any of several machine learning algorithms (such as support vector machines, SVM); or use a segmentation-based similarity measure with an algorithm that allows an arbitrary similarity measure to be plugged into it (e.g. the nearest-neighbor method). However, our earlier work [1] has shown that the nearest neighbor method, in combination with segmentation-based image similarity measures, results in rather unimpressive performance in comparison to SVM and global representations. It is therefore our goal to try using segmentation together with support vector machines. The main challenge here is that the SVM in its original formulation assumes all training and test examples to be described by vectors with the same number of components, while in the case of segmentation the description of each image has more structure than that, and the number of regions can also vary from image to image.

2 Support vector machines

Support Vector Machines (SVMs) [3] are a relatively recent family of machine learning algorithms that have been used successfully in many application domains. In the most elementary form of this method, we assume that each training example is a vector from some d-dimensional real space, and that there are exactly two classes, called positive and negative. Several extensions to multiclass problems are possible [5], usually by converting one multiclass learning problem into several two-class problems (e.g. training one classifier for each pair of classes to separate members of one class from those of the other class).

In SVM learning, we want to separate the positive vectors from the negative ones using a hyperplane such that the positive training vectors lie on one side of the plane and the negative ones lie on the other side.

Additionally, to help make the classifier more robust and more reliable for use on unseen test vectors, we want the training vectors to lie as far from the separating hyperplane as possible. Maximizing this distance (known as the *margin*) from the plane to the nearest training example can be cast as an optimization problem in the following way.

Let x_i be the ith training vector, and y_i its label (which equals +1 for positive examples and–1 for negative training examples. A hyperplane can be described by the equation $w^T x + b = 0$, where w is the "normal", i.e. a vector perpendicular to the plane, and b is a threshold that determines the actual location of the plane in space. $w^T x$ denotes the dot product of the vectors w and x. Given a particular vector x, we can determine what side of the plane it lies on by examining whether $w^T x + b$ is positive or negative. However, to ensure that the training examples do not lie too close to the plane, we must also insist that $w^T x + b$ has a large enough absolute value. We can describe this using the following conditions:

$$y_i = 1 \Rightarrow w^T x_i + b \geq 1 \quad \text{and} \quad y_i = -1 \Rightarrow w^T x_i + b \leq -1,$$

or, more concisely: $y_i(w^T x_i + b) \geq 1$ for all training instances i. If all training examples satisfy these conditions, the space between the hyperplanes $w^T x + b = 1$ and $w^T x + b = -1$ is empty; to maximize the breadth of this margin space, we need to maximize the distance between these two planes, which equals $2/\|w\|$. Maximizing the margin is thus equivalent to minimizing $\|w\|^2$ subject to the above conditions.

This optimization problem is usually also extended to allow some training instances to be misclassified (or at least lie within the margin, though perhaps on the correct side of the separating plane) if this leads to a wider margin on the other training instances (the *soft margin* formulation of SVM).

Solving the optimization problem gives us the values of w and b, and the resulting classifier simply works according to the formula $prediction(x) = \text{sgn}[w^T x + b]$.

Using standard techniques from optimization theory, this optimization problem can be transformed into a "dual" form. It turns out that the dual form, as well as the resulting classification rule, can be expressed so that the training vectors need never be accessed directly, as long as we are able to compute the dot product of any two vectors. In particular, the normal w can be written as $w = \Sigma_i \alpha_i y_i x_i$, where the α_i coefficients are obtained by solving the dual optimization problem. The classifier can then be described as $prediction(x) = \text{sgn}[b + \Sigma_i \alpha_i y_i x_i^T x]$.

Now suppose we used some mapping φ to map our original instances x_i into some other (possibly higher-dimensional) vector space F. Let $K(x_i, x_j) := \langle \varphi(x_i), \varphi(x_j) \rangle_F$ be a function that, given two instances x_i and x_j, computes the dot product $\langle \cdot, \cdot \rangle_F$ (in the new space F) of their images $\varphi(x_i)$ and $\varphi(x_j)$ under the mapping φ. It follows from the above that we could train a hyperplane in F without ever working with the mapped vectors $\varphi(x_i)$ explicitly, as long as we are able to compute $K(x_i, x_j)$ for any two vectors x_i and x_j. The function K defined in this way is known as a *kernel*. The importance of kernels arises from the fact that the mapping φ need not be linear, and for a nonlinear φ a hyperplane in F could correspond to some highly nonlinear separation surface in the original space. In this way, kernels allow the SVM algorithm to induce nonlinear models while preserving the optimization framework essentially intact. The appeal of kernels stems from the fact that a wisely chosen function K can be simple to compute and yet correspond to a complex nonlinear mapping into some very high-dimensional space F.

A kernel corresponds to a dot product in some vector space and can therefore in some sense be seen as a sort of similarity measure: the dot product of two vectors (if their length is fixed) is greatest when they point in the same direction, and then decreases as the angle between them increases, eventually becoming 0 (for orthogonal vectors) and even negative, reaching the minimum if the two vectors point in exactly the opposite direction.

However, the converse is not true: that is, not every similarity measure corresponds to a scalar product in some vector space. If we used a non-kernel similarity measure as if it were an actual kernel, we would no longer have the mathematical guarantees that the SVM training algorithm would converge, and even if it converged there would be no theoretical grounds to expect the resulting classifier to have good performance.

3 Generalized kernels

Generalized SVMs have been proposed by Mangasarian [9] to allow an arbitrary similarity function to be used in a way analogous to a kernel. In the previous section we have seen that SVM can learn nonlinear models of the form

$$prediction(x) = \text{sgn}[b + \Sigma_i \alpha_i y_i K(x_i, x)]$$

where $K(x_i, x) = \langle \varphi(x_i), \varphi(x) \rangle_F$ for some mapping φ to some space F and some dot product $\langle \cdot, \cdot \rangle_F$ in F.

Now if some arbitrary function K were used instead of a proper kernel function, again giving us a classifier of the form $\text{sgn}[b + \Sigma_i \alpha_i y_i K(x_i, x)]$, this might still be a perfectly reasonable and useful classifier, but it wouldn't necessarily correspond to some hyperplane in some vector space F to which the instances x_i and x might have been mapped. Thus we couldn't obtain the α_i values using the criterion of maximizing the margin, because there wouldn't even be a hyperplane whose margin to maximize. Instead, [9] proposes to minimize the value $\alpha^T H \alpha$ (subject to the same constraints as before, i.e. that our training instances should lie on the correct side of the separation surface) for some positive definite matrix H. (This problem has a very similar structure to the dual form of the original SVM optimization problem, and is in fact equivalent to it if K really corresponds to a dot product and a suitable matrix H is chosen.)

In the simplest case of the generalized SVM, we would take $H = I$ (the identity matrix) and thus minimize $\Sigma_i \alpha_i^2$. This can be interpreted intuitively as looking for a separation surface that can be expressed in the simplest

possible way, possibly with many α_i equal to 0 (i.e. without really using the training example x_i in the description of the separating surface).

It can be shown that the formulation for $H = I$ is equivalent to mapping each instance x into the vector $(K(x, x_1), \ldots, K(x, x_n))$ of its "similarities" (as measured by K) to all the training instances $x_1, \ldots, x_n$, and then using an ordinary linear support vector machine over this new representation. For the problem of image categorization, this amounts to the intuitively appealing suggestion that two images should be treated as similar if they exhibit a similar pattern of similarities to known training images.

4 Region clustering

In this section we consider another approach to using segmentation-based representations for image categorization. Each image has its own set of regions and regions belonging to different sets are in a sense quite independent of each other. This leads to the need for special similarity measures that compare two images by considering all pairs of regions and aggregating the similarities of regions into a measure of similarity between the images.

As an alternative, we propose to bring the region-based representations of images to a "common denominator" by clustering the descriptions of all the regions of all the training images. The hope here is that each cluster would correspond to a group of similar regions from several images, while regions from separate clusters would be quite different in appearance. Thus, when comparing two images, if a region of one image belongs to a different cluster than some region of the other image, there would be no need to compare these two regions in any particular way, because knowing that they belong to different clusters already indicates that they are different in appearance and cannot really contribute towards the similarity of the two images under consideration.

Therefore, an image would then be described by recording, for each cluster of regions, what proportion of the area of this image is covered by regions of this cluster. If there are d region clusters, each image would now be represented by a d-dimensional real vector (with possibly many zero-value components, as there would probably be much more clusters than an average image has regions). With all images represented in this same d-dimensional space, we can then use the ordinary linear support vector machine to train classifiers.

5 Experimental evaluation

To compare the approaches described in the previous sections, we conducted experiments on the *misc* database, which is publicly available (http://www-db.stanford.edu/IMAGE/) and has already been used in image retrieval literature [13, 10], as well as in our earlier work on image categorization [1]. This database contains approximately 10000 small photographic images (of sizes around 128 by 96 pixels). It is thematically very diverse.

We selected 1172 images from the database and manually assigned each of them to one of 14 categories (butterflies, US flag, sunsets, autumn, flowers, planets, satellite images of Earth, cars, mountains, clouds, sea, surfboards, sailboats, prairie animals). The intention of this selection was to have categories of varying size and difficulty. The smallest category (flags) contains 32 images, and the largest (sunsets) contains 224 images. Some of the categories, such as sunsets or flowers, have characteristic and easily recognizable color distributions, while some categories are quite similar in this respect and would therefore be more difficult to distinguish (e.g. sea and clouds, both of which have a lot of blue and white pixels).

To train the SVM classifiers, we used the LibSvm [2] program, which has the advantage of natively supporting multiclass problems. It uses the all-pairs approach to convert a multiclass problem to several two-class problems: for each pair of classes, a classifier is trained to distinguish members of one class from members of the other class. To classify a new example, it is shown to all the classifiers, each of which then votes for either one or the other of the two classes which it has been trained to separate. The class with the greatest number of votes is then adopted as the final prediction.

We compared the following approaches to image categorization:

1. Images are represented in the HSV (hue, saturation, value) color space, which is quantized into 256 colors (the H axis is split into 16 equal ranges and the S and V axes into 4 equal ranges). Each image is then described by an autocorrelogram in the resulting quantized color space. The autocorrelograms are 1024-dimensional vectors and are used as input for linear SVM.

2. Images are segmented into regions using the segmentation algorithm from WALRUS [10]. The IRM similarity metric [8] is then used to construct a generalized kernel as described in Section 3 above. In other words, each image is represented by a vector of its IRM similarities to all the training images; these vectors are then used as input for linear SVM.

3. Images are segmented as in the previous paragraph. Each region is described by a short (12-dimensional) vector, which is a by-product of the segmentation algorithm. The vectors resulting from all the regions of all the training images are then clustered (here we use the same algorithm, BIRCH [14], that is also used by WALRUS during segmentation). An image is then described by a sparse vector specifying what proportion of the area of the image is covered by regions from each region cluster. Depending on the parameters of the segmentation, the average number of regions per image might vary from less than ten to more than a hundred; then, depending on the parameters of the clustering, the number of region clusters (and hence the dimensionality of the space in which our images are now represented) is usually on the order of a few hundred.

Once images are represented in this way, linear SVM can be used to train classifiers for them.

For the sake of comparison, we also report the performance of the nearest neighbor method with the IRM similarity metric (that is, each image is predicted to belong to the same class as the most similar training image). All performance values reported here are averages (and standard errors) based on tenfold stratified cross-validation.

Method	Classification accuracy
Autocorrelograms	80.2 % ± 1.3 %
Generalized kernels	79.0 % ± 1.3 %
Region clustering	70.0 % ± 1.6 %
Nearest neighbors + IRM	69.1 % ± 1.3 %

As expected, the nearest-neighbor method is in general less successful than the approaches based on SVM. However, it turns out that the two segmentation-based approaches do not outperform the representation based on autocorrelograms. The performance of the generalized kernel method is not significantly different (using a paired t-test) from that of autocorrelograms, and the generalized kernel method has the additional disadvantage of much greater computational cost.

In addition, the performance of the region clustering approach is remarkably poor. A closer examination suggests that the partitioning of regions into region clusters is problematic and unstable. For example, if the centroid of each cluster is recorded and then all regions are distributed to the cluster with the nearest centroid, most of the regions will tend to move to a different cluster than they were originally attached to. This means that two otherwise similar regions might fall into different clusters by pure chance, and the similarity between their images would thus go unnoticed. The authors of the BIRCH clustering algorithm were aware of the possibility of such problems, and proposed several redistribution passes where the regions are redistributed to the nearest centroids, but in our experiments this did not lead to a really stable partition even after five or ten such passes.

An alternative way of making use of the region clustering approach might be to include the test images in the region clustering phase. This really amounts to a form of transduction, i.e. using test data as if it was simply additional unlabeled training data. It ensures that both the training images and the test images are really being represented in a space that treats both groups of images equally. In this setting, the performance of the region clustering increases considerably, and it achieves an accuracy of 86.4 % ± 1.0 %. However, for the comparison with other methods to be fair, transduction should also be included in the SVM learning process. Since LibSvm does not support tranduction, we used the SvmLight program [6] for these experiments; it implements Joachims' transductive SVM algorithm [7]. With transductive SVM, region clustering achieves an average accuracy of 91.9 % ± 1.0 %, while autocorrelograms achieve an accuracy of 90.7 % ± 1.1 %. Although this difference is not really significant from a practical point of view (a t-test shows that it is statistically significant at a confidence level of 0.945, slightly below the usual 0.95), it suggests that the region clustering approach does have at least some potential to be useful.

Finally, we also considered combining several representations. An analysis of classification errors shows that classifiers based on different representations often make mistakes on different test images; that is, a it often happens that a test image is classified correctly by one classifier but incorrectly by another. For example, consider the classifiers based on autocorrelograms and on generalized kernels (with the IRM measure). Of the 1172 images, 828 are classified correctly by both; 120 only by the former; 100 only by the latter; and 128 are misclassified by both. (To obtain these numbers, each image was classified by a model obtained from that 90% of the dataset which does not contain the image under consideration.)

Thus it seems that some advantage could be gained by combining the features of both of these representations. Many approaches exist for combining several classifiers, but with SVM, this can be done in a particularly simple way. If we have two representations, ϕ_1: $X \to F_1$ and ϕ_2,: $X \to F_2$, combining their features (or attributes) would be equivalent to a new representation ϕ: $X \to F_1 \times F_2$ defined by the formula $\phi(x) = (\phi_1(x), \phi_2(x))$. Now if the kernels $K_1(x_i, x_j)$ and $K_2(x_i, x_j)$ correspond to some dot product on F_1 and F_2, respectively, the function $K(x_i, x_j) := K_1(x_i, x_j) + K_2(x_i, x_j)$ is a dot product on $F_1 \times F_2$. Thus we can obtain the equivalent of a combined representation simply by computing the sum of two kernels.

In our experiments, the combination of the autocorrelogram representation and the generalized kernel using the IRM similarity measure achieved a categorization accuracy of 83.7 % ± 1.4 %. A t-test shows that this performance level is significantly better than that of either of these two representations individually.

6 Conclusions and future work

Our experiments show that it is difficult to use segmentation-based image representation methods in image categorization. Relatively complex ways of using information obtained from segmentation, such as the generalized kernel approach and (to a lesser extent) the region clustering approach, have been found able to compete with a simpler and more straightforward approach such as autocorrelograms but not to significantly outperform it. In the presence of unlabeled test images, the region clustering approach performs really well (relative to other representations) if a transductive SVM learner is not available. We have shown that it is possible to use segmentation-based representation in combination with another representation to achieve a small but significant increase of categorization accuracy.

We nonetheless believe that there must be ways of using segmentation more profitably for image catego-

rization, just as it is used in image retrieval, and that this is still an interesting topic for future work. In particular, it would be interesting to further explore the influence of the clustering algorithm used in the region clustering approach, and to look for more stable clustering algorithms that would allow the region clustering approach to perform better in the inductive in additional to the transductive setting.

In addition, as segmentation is a relatively complex task, and segmentation algorithms usually depend on several parameters, it would be interesting to explore the influence of these various parameters on the segmentation (and consequently on image categorization) in a more systematic way.

The region clustering approach could also be augmented by taking the similarity between different clusters into account. Currently, regions that belong to different clusters contribute to different components of the sparse vectors that describe our images, and therefore whatever similarity might exist between two regions from different clusters cannot contribute anything towards our algorithm's perceived similarity between their two images. Acknowledging that regions can be at least somewhat similar even if they belong to different clusters might lead to an improved representation, but would (if taken to the extreme case) again require us to do the equivalent of comparing every region of one image with every region of the other image, which is what the region clustering approach was designed to avoid in the first place. Perhaps one could determine (from the region clustering process itself), for each region cluster, just a few most similar clusters and then compare pairs of regions from the closely similar clusters but ignore pairs of regions from entirely unrelated clusters.

Region clustering could also be integrated with segmentation. Currently, segmentation is being performed separately on each image, by clustering the descriptions of its 4×4 pixel windows; then, the region descriptions of all the images in the training set are clustered to form region clusters. These two steps could be merged by considering the descriptions of all windows from all the images as a single large set and performing clustering on this. Each image would then be represented by a vector of values showing what proportion of the image is covered by windows belonging to a particular cluster.

Combination of several kernels could also be pursued further, particularly in the direction of combining more than two classifiers and using weighted sums of kernels.

Additionally, the methods considered here should be tested on other datasets, as (given that widely different methods achieve highly similar categorization accuracy values on the present dataset) it is perhaps simply unrealistic to expect better performance on the current dataset, as the categories have an essentially "semantic" motivation that the current image representation methods simply cannot capture.

References

[1] J. Brank: *Machine learning on images* (in Slovenian). Proc. IS 2001, Ljubljana, 2001, pp. 152–55.

[2] C.-C. Chang, C.-J. Lin: *LibSVM: a library for support vector machines* (version 2.3). Dept. of Comp. Sci. and Inf. Eng., Nat'l. Taiwan University, April 2001.

[3] C. Cortes, V. Vapnik: *Support-vector networks.* Machine Learning, 20(3):273–297, September 1995.

[4] J. Huang, S. R. Kumar, M. Mitra: *Combining supervised learning with color correlograms for content-based image retrieval.* Proc. 5th ACM Multimedia Conf., Seattle, USA, 1997, pp. 325–334.

[5] C.-W. Hsu, C.-J. Lin: *A comparison of methods for multi-class support vector machines.* Dept. of Comp. Sci. and Inf. Eng., Nat'l Taiwan University, April 2001.

[6] T. Joachims: *Making large-scale SVM learning practical.* In: B. Schölkopf et al. (eds.), Advances in Kernel Methods. MIT Press, 1999, pp. 169–184.

[7] T. Joachims: *Transductive inference for text classification using support vector machines.* Proc. 16th ICML, Bled, Slovenia, 1999, pp. 200–209.

[8] J. Li, J. Z. Wang, G. Wiederhold: *IRM: Integrated region matching for image retrieval.* Proc. 8th ACM Multimedia Conf., Los Angeles, 2000, pp. 147–156.

[9] O. L. Mangasarian: *Generalized support vector machines.* In: A. J. Smola et al. (eds.), Advances in Large Margin Classifiers, MIT Press, 2000, pp. 135–146.

[10] A. Natsev, R. Rastogi, K. Shim: *WALRUS: a similarity retrieval algorithm for large databases.* Proc. ACM SIGMOD, 1999, pp. 395–406.

[11] M. J. Swain, D. H. Ballard: *Color indexing.* Int. Journal of Computer Vision, 7(1):11–32, Nov. 1991

[12] J. Z. Wang, J. Li, G. Wiederhold: *SIMPLIcity: Semantics-sensitive integrated matching for picture libraries.* Advances in Visual Inf. Systems, 4th Int. Conf., 2000, pp. 360–371.

[13] J. Z. Wang, G. Wiederhold, O. Firschein, S. X. Wei: *Content-based image indexing using Daubechies' wavelets.* Int. Journal of Dig. Lib., 1(4):311–328, December 1997.

[14] T. Zhang, R. Ramakrishnan, M. Livny: *BIRCH: An efficient data clustering method for very large databases.* Proc. ACM SIGMOD, 1996. pp. 103–114.