Recognition of Images in Large Databases Using a Learning Framework*

Serge Belongie, Chad Carson, Hayit Greenspan, and Jitendra Malik
Computer Science Division, University of California at Berkeley
Berkeley, CA 94720
{sjb,carson,hayit,malik}@cs.berkeley.edu

Abstract

Retrieving images from very large collections using image content as a key is becoming an important problem. Classifying images into visual categories and finding objects in image databases are two major challenges in the field. This paper describes our approach toward the first of the two tasks, the generalization of which we believe will assist in the second task as well.

We define a blobworld *representation which provides a transition from the raw pixel data to a small set of localized coherent regions in color and texture space. Learning is then utilized to extract a probabilistic interpretation of the scene. Experimental results are presented for more than 1000 images from the Corel photo collection.*

1. Introduction

Very large collections of images are becoming common, and users have a clear preference for accessing images in these databases based on their content—be it the general image category (e.g., animal scenes, landscapes, urban scenes) or particular objects that are present in them. Creating indexes for these collections by hand is unlikely to be successful, because the databases can be prohibitively large. Furthermore, it can be very difficult to impose order on the collections. For example, the California Department of Water Resources (DWR) collection contains on the order of half a million images; a subset of this collection can be searched at `http://elib.cs.berkeley.edu`. Other examples include the Corel stock photo collection, which contains 60,000 images, as well as the collection of images available on the Internet, which is notoriously large and disorderly.

In this work our goal is to classify images in the Corel database into categories based on image content. Figure 1 indicates the variability of the images we are working with, even within a particular category.

*This work was supported by an NSF Digital Library Grant (IRI 94-11334) and NSF graduate fellowships for Serge Belongie and Chad Carson.

Classical object recognition techniques rely on clean segmentation of the object from the rest of the image and are designed for fixed, geometric objects such as machine parts. Neither constraint holds in our case; the shape, size, and color of objects like cheetahs and polar bears are quite variable, and segmentation is imperfect (even when their camouflage fails!). Clearly, classical object recognition does not apply. More recent techniques can identify specific objects drawn from a finite (on the order of 100) collection, but no present technique is effective at the general image analysis task, which requires both image segmentation and image classification.

We have found that interesting images can often be characterized by localized regions of coherent color and texture, defined in terms of their relative size and relative location in the image plane. Following are a few examples:

- In an *airplane* category, an important characterizing component is a large, uniform light blue region located at the top portion of the image, or all across the image, together with a small, dark region within it.

- In an *animal scene* category, an important recurring feature is a brown or green region at the bottom of the image, an animal-colored and -textured region in the center of the image, and a uniform light blue region at the top.

- In a *sunset scene* the sky colors (e.g., purple and orange) are significant, as is the fact that they are found in adjacent horizontally elongated patches; the existence of a distinct yellow or orange circle is also very informative.

It is evident that in order to analyze general image content, representing such localized coherent color-texture regions is key. In our system we shift from the raw pixel domain to such a representation, which we call the *blobworld*. We first perform feature extraction and then group the feature space into localized regions, or blobs. We believe that these first steps should be kept general and bottom-up, with no a priori

Figure 1. Sample images from the Corel database. Each row includes images from one category: air shows, black/brown bears, polar bears, elephants, tigers, cheetahs, bald eagles, mountains, fields, night scenes, deserts, and sunsets.

(top-down) model bias. We use learning to associate probabilistic interpretations of the blob ensembles in an image with categories.

In this paper we present an overview of our philosophy along with a detailed description of an implemented system.

2. Background

Existing content-based retrieval systems can handle queries about specific colors and textures but are far from achieving the higher-level content extraction that users prefer.

Object-oriented queries search for images that contain particular objects; such queries can be seen either as constructs on material queries [20], as essentially textual matters [21], or as the proper domain of object recognition. A third query mode looks for images that are iconic matches of a given image [10]. This matching strategy cannot find images based on the objects present, because it is sensitive to such details as the position of the objects in the image, the composition of the background, and the configuration of the objects—for example, it could not match a front and a side view of a horse.

As discussed in a review of model-based vision in the late 70s and early 80s [4], a number of researchers were addressing the task of scene labeling using hand-crafted image models. For example, the system developed by Ohta [18] seeks to label image regions as one of four object classes: sky, trees, buildings (with windows) and roads (with cars). The labeling process makes use of several top-down and bottom-up processing phases whereby segmentation steps involving color and texture are evaluated and re-evaluated to determine how well they satisfy a set of pre-programmed rules. Two example rules are that the sky meets the top border of the image and that cars appear dark relative to the road. As one might expect, the rules in this system are very image dependent and burdensome to itemize. Ohta's work nonetheless represents a significant step toward understanding scenes using properties of color and texture.

The best-known image database system is QBIC [16], which allows an operator to specify various properties of a desired image. The system then displays a selection of potential matches to those criteria, sorted by a score of the appropriateness of the match. Region segmentation is largely manual, but the most recent versions of QBIC [1] contain simple automated segmentation facilities. Photobook [19] largely shares QBIC's model of an image as a collage of flat, homogeneous frontally presented regions, but incorporates more sophisticated representations of texture and a degree of automatic segmentation. Further examples of systems that identify materials using low-level image properties include Virage [2], Candid [13], and Chabot [17]. None of these systems codes spatial organization in a way that supports object queries.

Variations on Photobook [15] use a form of supervised learning known in the information retrieval community as "relevance feedback" to adjust segmentation and classification parameters for various forms of textured region. When a user is available to tune queries, supervised learning algorithms can clearly improve performance, given appropriate object and image representations. Significantly, the representations used in these supervised learning algorithms do not code spatial relationships. Thus, these algorithms are unlikely to be able to construct a broad range of effective object queries.

Our approach uses similar kinds of low-level image features as a first step but also incorporates spatial information. The framework for representation and learning is flexible and can be extended to include shape information as well as a variety of relationships among meaningful regions in the image.

3. Feature extraction

At the lowest level of our system, grouping is based on coherent local image descriptors, such as color and texture. In this section we discuss the color and texture feature space. We investigate each separately, addressing its contribution to image segmentation, grouping, and description, as related to the general content-based retrieval task.

3.1. The color space

Color is an important cue in extracting information from images. Color histograms provide a global image color characterization and are commonly used in content-based retrieval systems [16, 17, 24]. They have proven to be very useful. Still, the global characterization is poor at, for example, distinguishing between a field of flowers and a single large flower, because it lacks information about how the color is distributed spatially. This example indicates the importance of grouping color in localized regions and of fusing color with textural properties.

Our color processing is based on partitioning the color space into perceptually meaningful channels in order to aid grouping and recognition. The perceptual channels we use loosely follow the color naming system of the Inter-Society Color Council and National Bureau of Standards [12], which uses six levels of detail to designate colors. These levels range from broad perceptual color names such as red, blue, and gray (13 colors) to about five million color designations defined by spectrophotometric measurements [26]. Only the first three levels correspond to human color names.

Our perceptual color categories are based on the first level of this system, slightly modified to better match our application.[1] The final list of colors includes red, orange, yellow, green, blue-green, light blue, blue, purple, pink, brown, white, gray, and black.

To determine the location and extent of each color in hue-saturation-value (HSV) space, the space is broken into $20 \times 10 \times 10$ grid points (taking into account that hue differences are more noticeable than saturation and value differences). For each grid point we presented a human observer with a patch of the corresponding synthesized color on a neutral gray background. For each perceptual color, the observer indicated how good an example of that color the patch was. (For any given patch, most perceptual colors had a matching score of zero.) In this way, we created a lookup table that allows us to divide any image into 13 color channels.

[1] We combined olive, yellow green, and green into one "green" category, since we found the distinctions among various greens detrimental in grouping and labeling vegetation. We added "light blue," which closely matches the sky in many images, and "blue-green," which matches ocean water.

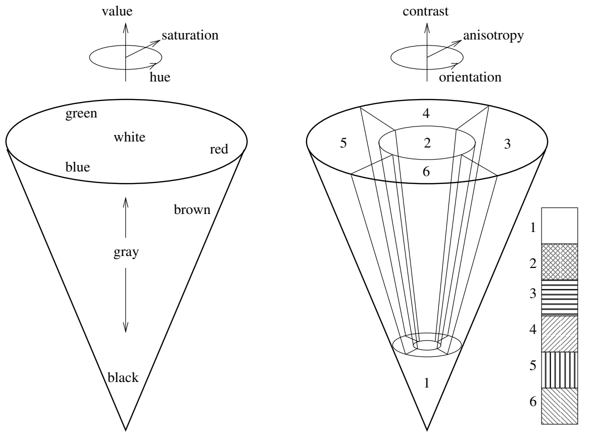

Figure 2. The color and texture cones, showing the texture bins and a few sample color locations.

Visualizing color data: the color cone

The standard red-green-blue (RGB) color space is not very useful for color processing, as distances in RGB space have little meaning and there is no simple (even approximate) mapping from RGB coordinates to human color names. A hue-based space such as HSV is superior to RGB in these respects [11].

In order to find distances in HSV space, we treat the space as a cone (see Fig. 2): for a given point (h, s, v), h and s are the angular and radial coordinates of the point on a disk of radius v at height v; all coordinates range from 0 to 1. Points with small v are black, regardless of their h and s values. This cone representation maps all such points to the apex of the cone, so they are close to one another. The Cartesian coordinates of points in the cone, $(sv\cos(2\pi h), sv\sin(2\pi h), v)$, can now be used to find color differences. This encoding allows us to operationalize the fact that hue differences are meaningless for very small saturations (those near the cone's axis). However, this scheme ignores the fact that for moderate and large values and saturations, hue is more perceptually relevant than saturation and value.

3.2. The texture space

Texture is a well-researched property of image regions, and many texture descriptors have been proposed, including multi-orientation filter banks [14, 9] and the second-moment matrix [7, 8]. We will not elaborate here on the classical approaches to texture segmentation and classification, both of which are challenging and well-studied tasks. Rather, we introduce a new perspective related to texture descriptors and texture grouping motivated by the content-based retrieval task.

In the framework of unconstrained image understanding, general categorizations become very useful and significant. Such is the case of identifying uniform intensity (non-textured) regions vs. textured regions. This often enables the extraction of foreground vs. background regions in the image, guiding the search for objects in the scene. In addition, distinguishing among texture patterns which are singly oriented (*1D texture*) or which are multiply oriented or stochastic in nature (*2D texture*) can allow for further categorization of the scene and for the extraction of higher-level features to aid the recognition process (e.g., singly-oriented flow is a strong characteristic of water waves; grass is stochastic; etc).

We extract texture features based on information obtained from the *windowed image second moment matrix*. Let us denote the image intensity (i.e., the v component) by $I = I(x, y)$. The first step is to compute the gradient, which we denote by ∇I, using the first difference approximation along each dimension. Then the windowed image second moment matrix $M(x, y)$ is computed via the expression

$$M(x, y) = G(x, y) * (\nabla I)(\nabla I)^T$$

where $G(x, y)$ is a 9×9 separable binomial approximation to a Gaussian smoothing kernel with variance 2. (Note that at each pixel location, $M(x, y)$ is a 2×2 symmetric positive semidefinite matrix.) The variance of G has been called the *integration scale* or *artificial scale* by various authors [7, 8] to distinguish it from the scale parameter used in linear smoothing of raw image intensities.

Consider a fixed scale and pixel location, let λ_1 and λ_2 denote the eigenvalues of M at that location ($\lambda_1 \geq \lambda_2$), and let ϕ denote the argument of the principal eigenvector. The relation between the eigenstructure of M and the local image structure it describes is well known [3, 7]. In particular, when λ_1 is large compared to λ_2, the local neighborhood possesses a dominant orientation (as specified by ϕ), and can be characterized as 1D-textured. When the eigenvalues are comparable, there is no preferred orientation. When both eigenvalues are negligible in magnitude, the local neighborhood is approximately a constant intensity and can be characterized as non-textured. For the case of two significant eigenvalues, we characterize the region as 2D-textured. In these cases, the value of ϕ is irrelevant.

We have found it beneficial to study the behavior of three expressions in particular which are functions of λ_1, λ_2 and ϕ.[2] The first is the *anisotropy*, $1 - \lambda_2/\lambda_1$. The second is the *texture contrast*, $\lambda_1 + \lambda_2$. The third is simply the doubled *orientation* angle, 2ϕ. These expressions are used to define texture categories: non-textured (featureless), 2D texture, and 1D texture. The latter is subdivided into 4 possible orientations: horizontally oriented texture, vertically oriented texture, and two diagonally oriented textures.

[2]These are related to derived quantities reported in [8].

Visualizing texture data: the texture cone

Since $M(x, y)$ possesses three values of interest (for a fixed scale) at each pixel, it is not immediately obvious how it should be visualized. We draw an analogy between the texture values and *hue, saturation, and value* as used in the HSV color-cone definition. In particular, we set $h = 2\phi$, $s = 1 - \lambda_2/\lambda_1$, and $v = \lambda_1 + \lambda_2$. In other words, the hue is set to be twice the orientation angle, the saturation is assigned the value of the anisotropy, and the value (or brightness) is associated with the texture contrast. (A similar color coding was suggested in [3].) In this manner, the "line of grays" associated with the zero-saturation axis of the HSV-cone corresponds to 2D textures of varying contrast. Textures possessing a preferred orientation correspond to saturated, colorful regions in color space. We refer to this representation as the "texture cone" to emphasize its relationship with the HSV cone. Just as the HSV cone tapers to a point when v is small, the texture cone tapers to a point when the contrast is small, so as to indicate that when the contrast is low, differences in orientation and anisotropy are irrelevant.

The six categories which we have chosen in our representation can be visualized as a partitioning of the texture cone as shown in Fig. 2.

4. From features to blobs

We model the input image as composed of an ensemble of regions, or *blobs*. Each blob represents a localized region of coherent color and texture (e.g., sky is a blue non-textured blob, usually located at the top of the image; grass is a green 2D-textured blob, usually located at the bottom of the image). In this stage of the system we move from the color and texture features to a blob representation of the image.

In the color domain we have found the need to adapt the decision boundaries in color space (as above) to the input image, as grouping based on color is very context dependent. The analysis begins with the distribution across the 13 color channels defined above (see Fig. 3(b)). The channels are ordered based on the number of pixels that belong to each and then collected in order of increasing non-overlapping contribution to image area until 90% of the image pixels are included. The image is thus reconstructed with the K most dominant colors which together cover at least 90% of the image area. K is typically 5 or less.

Upon discarding the weak channels, the means μ_i and covariance matrices Σ_i (computed in color cone coordinates) of each of the K dominant color channels are computed. The initial color labels of each pixel are subsequently dropped, and the μ_i's and Σ_i's are used to initialize a parameter search for a mixture of Gaussians using the Expectation Maximization (EM) algorithm [6, 22]. An iterative procedure forms the basis of the algorithm: the Expectation or E-step

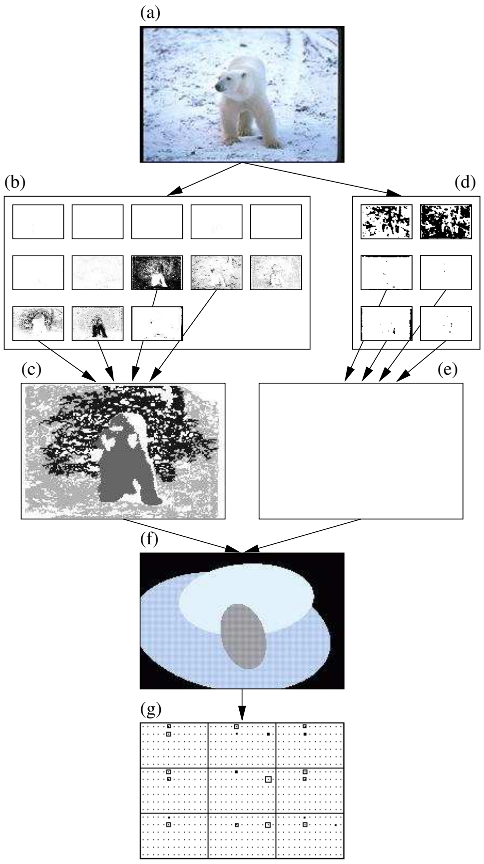

Figure 3. Finding blobs in an image. (a) Sample image. (b) The color channels; channels that initialize EM are indicated. (c) Support maps for the EM results; each gray level represents a different connected component. (d) The texture channels. The four 1D texture channels are allowed to contribute blobs to the ensemble. (e) Support maps for the 1D-texture blob. Here, no connected components of the combined 1D texture channel were large enough to contribute a blob. (f) Resulting ensemble of blobs; ellipses indicate the principal axes of each region's spatial extent. (g) Discretized tile representation of the image.

computes the expected data log likelihood, and the Maximization, or M-step, finds the parameters that maximize the likelihood. The output of the EM algorithm is a set of K support maps, together with their Gaussian means and covariances (see Fig. 3(c)).

In the process of iterating, the changes that occur in the μ_i's and Σ_i's correspond to a relabeling of regions in the color cone. To illustrate, consider a patch of sky in an image which initially falls mostly in the light blue bin except for a few small bits which go into the white bin, causing fragmentation. Viewed in the color cone, we would see that

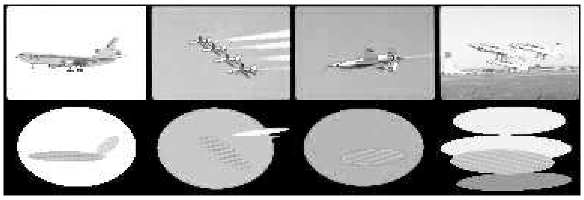

Figure 4. Sample blob ensembles for air show images. The ellipses indicate the principal axes of each region's spatial extent; the synthesized textures indicate the contrast, anisotropy, and orientation (if applicable) of the texture in the region.

the cluster corresponding to sky mostly falls inside the "light blue" decision region with a minority of points landing in the "white" decision region. After a modest number of iterations, the EM algorithm changes the color definitions so as to offer a "better" explanation of the data in terms of the mixture model. This explanation manifests itself in a set of "support maps," i.e., the K images corresponding to the most likely color-blob membership for each pixel in the image.

These support maps, along with the support maps for the 1D texture channels (see Fig. 3(d-e)), provide an initial segmentation of the image. In order to spatially localize the extracted regions, a connected-component algorithm is utilized. The final output consists of localized, coherent regions. This output provides us with the desired ensemble of blobs (see Fig. 4).

5. Learning in blobworld

We wish to recognize images, or image categories, from the blob ensembles. Of course, we would ideally like to have a perfect segmentation procedure; the interesting question is what we can do with a simple, imperfect segmenter.

5.1. Representing blob probabilities

As a discretized encoding of the blobworld representation, we encode the blob representation of each image as follows: The image is divided into 3×3 tiles (representing top, center, and bottom, and left, center, and right). The discrete map consists of 9 tiles, each subdivided into 78 points to represent the 13×6 color-texture bins (see Fig. 3(g)). Each blob contributes a vote in each tile based on the relative portion of the blob which exists in that part of the image. The relative votes are indicated via different size squares (representing different weights). This encoding scheme allows both robustness to fragmentation (a green field that splits into two blobs will contribute votes to the same tiles and bins that it would if it were not split) and size invariance (a small bird in the center tile will contribute the same weight as a large bird in that tile).

5.2. Classification

Consider an input image with n extracted blobs. We would like to find the class which best fits the image. The selection of a class is based on the evidence provided by the blobs present in the image as well as on "the dog that did nothing in the night-time"[3]: in order for an image to be classified into class C_j, each blob in the image should be consistent with C_j, and *the image should not lack any blobs which are defining features of the class.*

We first address the blobs which are present in the image. For each class we estimate the class-conditional blob probability density $\Pr(\text{blob}|C_j)$ using histogram averaging; each histogram bin specifies a particular color-texture bin and tile position.

To address those color-texture bins which do not match any blob in the image, we make use of the prior probability $\Pr(\text{bin not empty}|C_j)$ that a bin will contain *some* blob in an image of class C_j. For each color-texture bin, we take this prior probability to be the fraction of images in class C_j which have a blob in that color-texture bin, in any position in the image. Thirty-seven of the 78 bins were very rarely filled by blobs in the training data. To reduce noise, we reduced the fraction to zero in those cases; the result is that images are not penalized for not containing very rare blobs.

For each input image, we choose that class C_j which maximizes

$$\left[\prod_{k=1}^{n} \Pr(\text{blob}_k|C_j)\right] \left[\prod_{1\leq i\leq 78\,:\,\text{bin}_i\text{ empty}} \Pr(\text{bin}_i\text{ empty}|C_j)\right]$$

It is not uncommon that an image has several blobs which fall in the same color-texture bin and overlap in some spatial tile. In these cases, we make the approximation of adding the blob probabilities $\Pr(\text{blob}_k|C_j)$. For simplicity we have assumed that the blobs which support a given class are mutually independent. The above scheme represents a naïve Bayes classifier [23].

6. Experiments

6.1. The image collection

Our image collection consists of 28,000 of the Corel images online in a database [5]. The images are arranged in sets of 100, each with a category name; some of these categories are visually meaningful (e.g., "Cheetahs, Leopards, and Jaguars"), while others are not (e.g., "Tour through Europe"). We selected the following 12 categories[4]: air shows, brown and black bears, polar bears, elephants, tigers, cheetahs, bald eagles, mountains, fields, night scenes, deserts, and sunsets (see Fig. 1). We pruned a few images from each category, leaving approximately 90 images in most of the categories.[5] We also rotated images as necessary so that all were upright.

We divided each category into training and testing sets, containing 2/3 and 1/3 of the images, respectively. While exploring various approaches and evaluating their capabilities, we used four-fold cross-validation on the training set. We did not look at the testing set or perform any experiments on it until the complete procedure was fixed; the testing set was used only once, to find results using the final classification method.

6.2. Comparison with global color histograms

In order to obtain a baseline for the retrieval results, we performed an experiment using color histogram comparisons [25]. We first found the color histogram for each image using the 13 perceptual color channels. (Using a low-dimensional histogram is necessary since there is so much variability among the images in each category.)

We then found average histograms for each category using all combinations of the cross-validation partitions. Each incoming image's histogram was compared to the average histograms for the three folds not containing that image. (For example, the histogram for an image in fold 2 of the air shows category was compared to the average histogram for the other three folds for each category.) The image was classified in that category whose average histogram was closest to the image histogram.

6.3. Results

The results for each experiment are summarized using a standard class-confusion matrix and an additional measure of how often the correct class was selected as the first or second choice (see Tables 1 and 2). In each case the entries are percentages.

We can observe from the top-2 statistics that the overall performance of the two schemes is comparable; however, these are not the figures which are of most interest to us. It is most interesting to investigate the nature of the confusions in each scheme and, more generally, which scheme performs better on which images.

Consider for example the confusion that occurs in class a (air shows). The misclassified air show scenes primarily fall into class c (polar bears) in the color histogram scheme. This is not surprising, since the global color percentages in these two classes are quite similar. In our proposed method, the majority of the misclassified air show images fall into class g (bald eagle). This is arguably a preferable misclassification, since the air show and bald eagle images both typically exhibit gray/white/black textured blobs near the center of

[3] Sir Arthur Conan Doyle, "Silver Blaze," *Memoirs of Sherlock Holmes*.

[4] Categories were picked under the single constraint that they contain a significant number of visually consistent images.

[5] There are fewer images of polar bears and brown/black bears, because those two datasets come from one "Bears" category.

	a	b	c	d	e	f	g	h	i	j	k	l
a.	**80**	0	0	0	0	0	11	0	3	3	0	0
b.	0	**19**	0	14	14	9	14	0	14	0	9	4
c.	18	0	**54**	0	9	0	18	0	0	0	0	0
d.	6	12	6	**36**	9	9	0	3	12	0	6	0
e.	0	3	0	3	**54**	15	0	0	12	9	3	0
f.	0	0	0	6	21	**36**	0	0	12	15	9	0
g.	35	0	0	0	3	3	**48**	6	0	3	0	0
h.	16	0	3	0	0	0	13	**63**	3	0	0	0
i.	24	6	0	3	0	6	9	0	**48**	0	3	0
j.	0	3	0	0	3	0	0	0	3	**79**	3	6
k.	16	0	8	0	4	4	12	4	0	4	**48**	0
l.	0	0	0	3	0	0	3	0	0	3	0	**89**

a	b	c	d	e	f	g	h	i	j	k	l
96	38	63	51	69	63	77	73	63	86	72	89

Table 1. Class-confusion matrix for the first choice and statistics of a correct match occurring in the top 2, using the proposed scheme. The classes are (a) air shows, (b) brown/black bears, (c) polar bears, (d) elephants, (e) tigers, (f) cheetahs, (g) bald eagles, (h) mountains, (i) fields, (j) night scenes, (k) deserts, and (l) sunsets. Entries are percentages.

the image, while the rest of the image is essentially blue or light blue. The same general layout of colors is not found in the polar bear images.

We take the poor performance of class *b* (brown/black bears) in both schemes as evidence that class *b* does not constitute a strong visual category. The context of the bear in these images was highly variable in comparison to the other animal categories, though class *d* (elephant) suffered from a similar lack of consistency.

Class *l* (sunsets) represents a class for which our proposed scheme is well suited. The confusions we observe for sunsets in the color histogram case, namely with classes *j*, *f*, and *e* (night scenes, cheetahs, tigers), are due to global similarities in color which fail to capture the distinctive horizontally laminated strips of color and darkness which occur so frequently in sunset images.

Confusion of a similar nature can be observed in both schemes for those classes which are approximate subsets of other classes. Classes *i* and *k* (fields and desert) are examples of such confusion.

To the extent that a given Corel category constitutes a visual category, we feel that the results presented here indicate that there is something to be gained over simple color histogram matching in terms of making query results more meaningful.

	a	b	c	d	e	f	g	h	i	j	k	l
a.	**70**	5	23	0	0	0	0	0	0	0	0	0
b.	0	**29**	0	11	11	17	11	0	5	0	11	0
c.	27	0	**54**	0	0	0	9	0	0	0	9	0
d.	0	5	0	**64**	5	5	0	0	17	0	0	0
e.	0	11	0	11	**35**	5	0	0	23	0	5	5
f.	0	0	0	29	0	**47**	0	0	11	0	5	5
g.	35	0	0	0	0	0	**47**	5	0	0	11	0
h.	17	0	23	0	0	0	5	**52**	0	0	0	0
i.	0	5	0	0	0	35	5	0	**35**	0	17	0
j.	0	0	0	0	0	0	0	0	0	**76**	5	17
k.	17	5	11	0	5	5	5	0	11	0	**35**	0
l.	0	0	0	0	5	5	0	0	0	5	0	**82**

a	b	c	d	e	f	g	h	i	j	k	l
88	52	90	64	47	82	82	70	58	82	64	88

Table 2. Class-confusion matrix for the first choice and statistics of a correct match occurring in the top 2, using color histograms. The classes are the same as in Table 1. Entries are percentages.

7. Conclusion

Our goal in this work is to present a general framework for content-based image retrieval which allows for extensions on all levels, from the features to the classification scheme used.

This extensibility is a key advantage of the proposed scheme. While color histogram matching achieved results similar to those presented here, this is the best color histograms can do, whereas we have just taken the first step.

A variety of extensions may be made to the system within the current framework; much work remains to be done. In the feature extraction stage the next major step will be the addition of shape cues to the recognition process. Shape is very important for more accurate blob descriptors, all the way to full object recognition. We are currently looking at using the boundaries of the regions extracted in color and texture space as initializing closed contours in the image plane. Contour fragments or moments will then help enrich our feature space with shape features.

As the blob descriptors become cleaner and more representative of the image content (i.e. more consistent), a transition from the naïve Bayes approach will be justifiable. We can move toward more sophisticated learning schemes which look for higher-level dependencies between blobs as well as spatial relationships between blob pairs. Higher-order Bayesian nets (or Belief nets) are currently being investigated for learning conjunctions between variables.

The framework which we propose can be made hierarchical in the grouping and learning processes. We see this as providing for a new approach to object recognition, which will allow for recognition in general environments. In this approach, an object is modeled as a loosely coordinated collection of detection and grouping rules. The object is recognized if a suitable group can be built. Grouping rules incorporate both surface properties (color and texture) and shape information. This type of model gracefully handles objects whose precise geometry is extremely variable, where the identification of the object depends heavily on non-geometrical cues (e.g., color) and on the interrelationships between parts.

A great deal of work is required to fully elaborate and test this model of feature extraction, grouping, and recognition and ultimately to incorporate object modeling. Still, there is reason to believe that it will cover a wide range of categories and objects.

8. Acknowledgments

We would like to thank David Forsyth for useful discussions related to this work.

References

[1] J. Ashley et al. Automatic and semiautomatic methods for image annotation and retrieval in QBIC. In *SPIE Proc. Storage and Retrieval for Still Image and Video Databases III*, 1995.

[2] J. Bach et al. The Virage image search engine: An open framework for image management. In *SPIE Proc. Storage and Retrieval for Still Image and Video Databases IV*, 1996.

[3] J. Bigün. *Local symmetry features in image processing*. PhD thesis, Linköping University, 1988.

[4] T. Binford. Survey of model-based image analysis systems. *Int. J. Rob. Res.*, pages 18–64, 1982.

[5] C. Carson and V. Ogle. Storage and retrieval of feature data for a very large online image collection. *IEEE Data Engineering Bulletin*, to appear.

[6] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Soc., Ser. B*, 39(1):1–38, 1977.

[7] W. Förstner. A framework for low level feature extraction. In *Proc. European Conf. Comp. Vis.*, 1994.

[8] J. Gårding and T. Lindeberg. Direct computation of shape cues using scale-adapted spatial derivative operators. *Int. J. of Comp. Vis.*, 17, Feb 1996.

[9] H. Greenspan et al. Learning texture discrimination rules in a multiresolution system. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 16(9):894–901, 1994.

[10] C. Jacobs, A. Finkelstein, and D. Salesin. Fast multiresolution image querying. In *Proc. SIGGRAPH-95*, pages 277–285, 1995.

[11] A. Jain. *Fundamentals of digital image processing*. Prentice Hall, 1989.

[12] K. Kelly and D. Judd. *Color: Universal Language and Dictionary of Names*. U.S. National Bureau of Standards, Dec 1976. Spec. Publ. 440.

[13] P. Kelly, M. Cannon, and D. Hush. Query by image example: the comparison algorithm for navigating digital image databases (CANDID) approach. In *SPIE Proc. Storage and Retrieval for Image and Video Databases III*, pages 238–249, 1995.

[14] J. Malik and P. Perona. Preattentive texture discrimination with early vision mechanisms. *J. Opt. Soc. Am. A*, 7(5):923–932, 1990.

[15] T. Minka. An image database browser that learns from user interaction. Technical Report 365, MIT Media Lab, 1995.

[16] W. Niblack et al. The QBIC project: querying images by content using colour, texture and shape. In *IS and T/SPIE 1993 Int. Symp. Electr. Imaging: Science and Technology, Conf. 1908, Storage and Retrieval for Image and Video Databases*, 1993.

[17] V. Ogle and M. Stonebraker. Chabot: Retrieval from a relational database of images. *IEEE Computer*, 28(9), Sep 1995.

[18] Y. Ohta. *A region-oriented image-analysis system by computer*. PhD thesis, Kyoto University, 1980.

[19] A. Pentland, R. Picard, and S. Sclaroff. Photobook: content-based manipulation of image databases. Technical Report 255, MIT Media Lab Perceptual Computing, 1993.

[20] R. Picard and T. Minka. Vision texture for annotation. *J. Multimedia Sys.*, 3:3–14, 1995.

[21] R. Price, T.-S. Chua, and S. Al-Hawamdeh. Applying relevance feedback to a photo-archival system. *J. Information Sci.*, 18:203–215, 1992.

[22] R. Redner and H. Walker. Mixture densities, maximum-likelihood estimation and the EM algorithm (review). *SIAM Review*, 26(2):195–237, 1984.

[23] B. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[24] M. Swain and D. Ballard. Color indexing. *Int. J. Comp. Vis.*, 7(1):11–32, 1991.

[25] M. Swain and M. Stricker. The capacity and the sensitivity of color histogram indexing. Technical Report 94-05, University of Chicago, Mar 1994.

[26] G. Wyszecki and W. Stiles. *Color science: concepts and methods, quantitative data and formulae*. Wiley, second edition, 1982.