Color- and Texture-Based Image Segmentation Using EM and Its Application to Content-Based Image Retrieval

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 large and varied collections using image content as a key is a challenging and important problem. In this paper we present a new image representation which provides a transformation from the raw pixel data to a small set of image regions which are coherent in color and texture space. This so-called "blobworld" representation is based on segmentation using the Expectation-Maximization algorithm on combined color and texture features. The texture features we use for the segmentation arise from a new approach to texture description and scale selection.

We describe a system that uses the blobworld representation to retrieve images. An important and unique aspect of the system is that, in the context of similarity-based querying, the user is allowed to view the internal representation of the submitted image and the query results. Similar systems do not offer the user this view into the workings of the system; consequently, the outcome of many queries on these systems can be quite inexplicable, despite the availability of knobs for adjusting the similarity metric.

1 Introduction

Very large collections of images are growing ever more common. From stock photo collections to proprietary databases to the Web, these collections are diverse and often poorly indexed; unfortunately, image retrieval systems have not kept pace with the collections they are searching. The shortcomings of these systems are due both to the image representations they use and to their methods of accessing those representations to find images:

- While users generally want to find images containing particular objects ("things") [4, 6], most existing image retrieval systems represent images based only on their low-level features ("stuff"), with little regard for the spatial organization of those features.
- Systems based on user querying are often unintuitive and offer little help in understanding why certain images were returned and how to refine the query. Often the user knows only that he has submitted a query for, say, a bear and retrieved very few pictures of bears in return.
- For general image collections, there are currently no systems that automatically classify images or recognize the objects they contain.

In this paper we present a new image representation, "blobworld," and a retrieval system based on this representation. While blobworld does not exist completely in the "thing" domain, it recognizes the nature of images as combinations of objects, and both querying and learning in blobworld are more meaningful than they are with simple "stuff" representations.

We use the Expectation-Maximization (EM) algorithm to perform automatic segmentation based on image features. EM iteratively models the joint distribution of color and texture with a mixture of Gaussians; the resulting pixel-cluster memberships provide a segmentation of the image. After the image is segmented into regions, a description of each region's color, texture, and spatial characteristics is produced. In a querying task, the user can access the regions directly, in order to see the segmentation of the query image and specify which aspects of the image are important to the query. When query results are returned, the user sees the blobworld representation of the returned images; this assists greatly in refining the query.

We begin this paper by briefly discussing the current state of image retrieval. In Section 2 we describe the blobworld representation, from features through segmentation to region description. In Section 3 we present a query system based on blobworld, as well as results from queries in a collection of highly varied natural images.

1.1 Background

Current image database systems include IBM's Query by Image Content (QBIC) [18], Photobook [20], Virage [10], Candid [14], and Chabot [19]. These systems primarily use low-level image properties; several of them include some degree of automatic segmentation. None of the systems codes spatial organization in a way that supports object queries.

Classical object recognition techniques usually rely on clean segmentation of the object from the rest of the image or 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. Clearly, classical object recognition does not apply. More recent techniques [21] 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.

Promising work by Lipson et al. [16] retrieves images based on spatial and photometric relationships within and across image regions. Little or no segmentation is done; the regions are derived from low-resolution images.

Earlier work has used the EM algorithm and/or the Minimum Description Length (MDL) principle to perform segmentation based on motion [1, 25] or scaled intensities [26], but EM has not previously been used on joint color and texture. Related approaches such as deterministic annealing [11] and classical clustering [12] have been applied to texture segmentation without color.

2 The blobworld image representation

The blobworld representation is related to the notion of photographic or artistic scene composition. In the sense discussed in [23], the blobworld descriptors constitute an example of a *summary representation* because they are concise and relatively easy to process in a querying framework.

Blobworld is distinct from color-layout matching as in QBIC in that it is designed to find objects or parts of objects. Each image may be visualized by an ensemble of 2-D ellipses, or "blobs," each of which possesses a number of attributes. The number of blobs in an image is typically less than ten. Each blob represents a region of the image which is roughly homogeneous with respect to color or texture. A blob is described by its dominant colors, mean texture descriptors, and spatial centroid and scatter matrix. (See Figs. 3–4 for a visualization of blobworld.)

2.1 Extracting color and texture features

Our goal is to assign the pixels in the original image to a relatively small number of groups, where each group represents a set of pixels that are coherent in their color and local texture properties; the motivation is to reduce the amount of raw data presented by the image while preserving the information needed for the image understanding task. Given the unconstrained nature of the images in our database, it is important that the tools we employ to meet this goal be as general as possible without sacrificing an undue amount of descriptive power.

2.1.1 Color

Color is a very important cue in extracting information from images. Color histograms are commonly used in content-based retrieval systems [18, 19, 24] and have proven to be very useful; however, the global characterization is poor at, for example, distinguishing between a field of orange flowers and a tiger, because it lacks information about how the color is distributed spatially. It is important to group color in localized regions and to fuse color with textural properties.

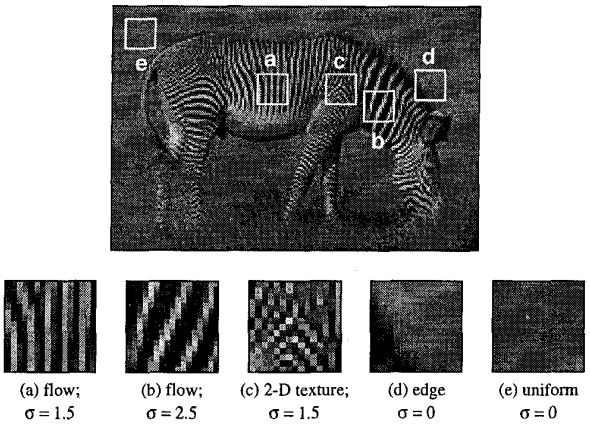
We treat the hue-saturation-value (HSV) color space as a cone: for a given point (h, s, v), h and sv are the angular and radial coordinates of the point on a disk of radius v at height v; all coordinates range from 0 to 1. Points with small v are black, regardless of their h and s values. The cone representation 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, it ignores the fact that for large values and saturations, hue differences are perceptually more relevant than saturation and value differences.

2.1.2 Texture

Texture is a well-researched property of image regions, and many texture descriptors have been proposed, including multi-orientation filter banks [17] and the second-moment matrix [5, 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.

While color is a point property, texture is a local-neighborhood property. It does not make sense to talk about the texture of zebra stripes at a particular pixel without specifying a neighborhood around that pixel. In order for a texture descriptor to be useful, it must provide an adequate description of the underlying texture parameters and it must be computed in a neighborhood which is appropriate to the local structure being described.

The first requirement could be met to an arbitrary degree of satisfaction by using multi-orientation filter banks such as steerable filters; we chose a simpler method that is sufficient



(a) flow; $\sigma = 1.5$ (b) flow; $\sigma = 2.5$ (c) 2-D texture; $\sigma = 1.5$ (d) edge $\sigma = 0$ (e) uniform $\sigma = 0$

Figure 1. Five sample patches from a zebra image. (a) and (b) have stripes (1-D flow) of different scales and orientations, (c) is a region of 2-D texture, (d) contains an edge, and (e) is a uniform region.

for our purposes. The second requirement, which may be thought of as the problem of *scale selection*, does not enjoy the same level of attention in the literature. This is unfortunate, since texture descriptors computed at the wrong scale only confuse the issue.

In this work, we introduce a novel method of scale selection which works in tandem with a fairly simple but informative set of texture descriptors. The scale selection method is based on edge/bar polarity stabilization, and the texture descriptors arise from the windowed second moment matrix. Both are derived from the gradient of the image intensity, which we denote by ∇I. We compute ∇I using the first difference approximation along each dimension. This operation is often accompanied by smoothing, but we have found this preprocessing operation unnecessary for the images in our collection.

To make the notion of scale concrete, we define the scale to be the width of the Gaussian window within which the gradient vectors of the image are pooled. The second moment matrix for the vectors within this window, computed about each pixel in the image, can be approximated using

$$M_\sigma(x,y) = G_\sigma(x,y) * (\nabla I)(\nabla I)^T \qquad (1)$$

where $G_\sigma(x,y)$ is a separable binomial approximation to a Gaussian smoothing kernel with variance σ^2.

At each pixel location, $M_\sigma(x,y)$ is a 2×2 symmetric positive semidefinite matrix; thus it provides us with three pieces of information about each pixel. Rather than work with the raw entries in M_σ, it is more common to deal with its eigenstructure [2, 5]. Consider a fixed scale and pixel location, let λ_1 and λ_2 ($\lambda_1 \geq \lambda_2$) denote the eigenvalues of M_σ at that location, and let ϕ denote the argument of the principal eigenvector of M_σ. When λ_1 is large compared to λ_2, the local neighborhood possesses a dominant orientation, as specified by ϕ. When the eigenvalues are comparable, there is no preferred orientation, and when both eigenvalues are negligible, the local neighborhood is approximately constant.

Scale selection

We may think of σ as controlling the size of the integration window around each pixel within which the outer product of the gradient vectors is averaged. σ has been called the *integration scale* or *artificial scale* by various authors [5, 8] to distinguish it from the *natural scale* used in linear smoothing of raw image intensities. Note that $\sigma = \sigma(x,y)$; the scale varies across the image.[1]

In order to select the scale at which M_σ is computed, i.e. to determine the function $\sigma(x,y)$, we make use of a local image property known as *polarity*.[2] The polarity is a measure of the extent to which the gradient vectors in a certain neighborhood all point in the same direction. (In the computation of second moments, this information is lost in the outer product operation; i.e., gradient vector directions differing by 180° are indistinguishable.) The polarity at a given pixel is computed with respect to the dominant orientation ϕ in the neighborhood of that pixel. For ease of notation, let us consider a fixed scale and pixel location. We define polarity as

$$p = \frac{|E_+ - E_-|}{E_+ + E_-}$$

The definitions of E_+ and E_- are

$$E_+ = \sum_{(x,y)\in\Omega} G_\sigma(x,y)[\nabla I \cdot \hat{n}]_+$$

and

$$E_- = \sum_{(x,y)\in\Omega} G_\sigma(x,y)[\nabla I \cdot \hat{n}]_-$$

where $[q]_+$ and $[q]_-$ are the rectified positive and negative parts of their argument, $\hat{n}$ is a unit vector perpendicular to ϕ, and Ω represents the neighborhood under consideration. We can think of E_+ and E_- as measures of how many gradient vectors in Ω are on the "positive side" and "negative side" of the dominant orientation, respectively. Note that p ranges from 0 to 1. A similar measure is used in [15] to distinguish a flow pattern from an edge.

[1] Strictly speaking, eqn. (1) is a sliding inner product, not a convolution, since $\sigma(x,y)$ is spatially variant.

[2] Polarity is related to the *quadrature phase* as discussed in [7, 9].

The polarity p_σ varies as the scale σ changes; its behavior in typical image regions can be summarized as follows:

Edge: The presence of an edge is signaled by p_σ holding values close to 1 for all σ.

Texture: In regions with 2-D texture or 1-D flow, p_σ decays with σ due to the presence of multiple orientations.

Uniform: In a constant-intensity neighborhood, p_σ takes on arbitrary values since the gradient vectors have negligible magnitudes and therefore arbitrary angles.

The process of selecting a scale is based on the derivative of the polarity with respect to scale. First, we compute the polarity at every pixel in the image for $\sigma_k = k/2, k = 0, 1, \ldots, 7$, thus producing a "stack" of polarity images across scale. Then, for each k, the polarity image computed at scale σ_k is convolved with a Gaussian with standard deviation $2\sigma_k$ to yield a smoothed polarity image $\tilde{p}_{\sigma_k}$. For each pixel, we select the scale as the first value of σ_k for which the difference between successive values of polarity $(\tilde{p}_{\sigma_k} - \tilde{p}_{\sigma_{k-1}})$ is less than 2%. In this manner, we are performing a soft version of local spatial frequency estimation, since the smoothed polarity tends to stabilize once the scale window encompasses one approximate period. Since we stop at $\sigma_k = 3.5$, the largest period we can detect is approximately 10 pixels. Note that when the period is undefined, as is the case in uniform regions, the selected scale is not meaningful and is set to zero. We declare a pixel to be uniform if its mean contrast across scale is less than 0.1.

Another method of scale selection that has been proposed [8] is based on localizing extrema across scale of an invariant of M_σ, such as the trace or determinant. In this algorithm, which is applied to the problem of estimating the slant and tilt of surfaces with tangential texture, it is necessary to perform natural smoothing at a scale tied to the artificial scale. We found that this extra smoothing compromised the spatial localization ability of our scale selection method.

Texture features

Once a scale σ^* is selected for each pixel, that pixel is assigned three texture descriptors. The first is the polarity, p_{σ^*}. The other two, which are taken from M_{σ^*}, are the anisotropy, defined as $a = 1 - \lambda_2/\lambda_1$, and the normalized texture contrast, defined as $c = 2\sqrt{\lambda_1 + \lambda_2}$.[3] These are related to derived quantities reported in [8].

2.1.3 Combining color and texture features

The color/texture descriptor for a given pixel consists of six values: three for color and three for texture. The three color components are the color-cone coordinates found after spatial averaging using a Gaussian at the selected scale. The three texture components are ac, pc, and c, computed at the selected scale; the anisotropy and polarity are each modulated by the contrast in analogy to the construction of the color-cone coordinates. (Recall that anisotropy and polarity are meaningless in regions of low contrast.) In effect, a given textured patch in an image first has its texture properties extracted and is then replaced by a smooth patch of averaged color. In this manner, the color and texture properties in a given region are decoupled; for example, a zebra is a gray horse plus stripes.

Note that in this formulation of the color/texture descriptor, orientation and selected scale do not appear in the feature vector; as a result, grouping can occur across variations in scale and orientation.

[3] If we use a centered first difference kernel in the gradient computation, the factor of 2 makes c range from 0 to 1.

2.2 Grouping with the EM Algorithm

Once an image has been processed using the above feature extraction scheme, the result is a large set of 6-D feature vectors, which we may regard as points in a 6-D feature space. In order to divide these points into groups, we make use of the Expectation-Maximization (EM) algorithm [3] to determine the maximum likelihood parameters of a mixture of K Gaussians inside the 6-D feature space.

The EM algorithm is used for finding maximum likelihood parameter estimates when there is missing or incomplete data. In our case, the missing data is the region to which the points in the feature space belong. We estimate values to fill in for the incomplete data (the "E-Step"), compute the maximum likelihood parameter estimates using this data (the "M-Step"), and repeat until a suitable stopping criterion is reached.

The first step in applying the EM algorithm is to initialize a mean vector and covariance matrix to represent each of the K groups. We initialize the means to random values and the covariances to identity matrices. (In earlier work we chose the initialization for EM carefully, but we have found that the initialization has little effect on the quality of the resulting segmentation.) The update scheme allows for full covariance matrices; variants include restricting the covariance to be diagonal or a constant times the identity matrix. Full covariance matrices suit our problem, since many plausible feature clusters require extruded covariance shapes, e.g. the shades of gray along the color cone axis.

Upon convergence, the Gaussian mixture parameters can be inspected to determine what color/texture properties are represented by each component of the mixture. Some examples of groups that can form include the following:

- bright, bluish, and textureless regions (e.g., sky)
- anisotropic and non-polar regions (e.g., zebra hide)
- green weak-isotropic texture (e.g., grass)

We have thus far not discussed how to choose K, the number of mixture components. Ideally we would like to choose that value of K that best suits the natural number of groups present in the image. One readily available notion of goodness of fit is the log-likelihood. Given this indicator, we can apply the Minimum Description Length (MDL) principle [22] to select among values of K. As a consequence of this principle, when models using two values of K fit the data equally well, the simpler model will be chosen. For our experiments, K ranges from 2 to 5.

Once a model is selected, the next step is to perform spatial grouping of those pixels belonging to the same color/texture cluster. We first produce a K-level image which encodes pixel-cluster memberships by replacing each pixel with the label of the cluster for which it attains the highest likelihood (see Fig. 2(d)). To enforce a small amount of spatial smoothness in this representation, we apply a 3×3 maximum-vote filter to the raw cluster-membership image. Finally, we run the resulting image through a connected-components algorithm to produce a set of labeled image regions (see Fig. 2(e)). (Alternatively, one could enforce spatial constraints by appending the pixel coordinates to the feature vectors, though we observed that this method too often yields unsatisfactory segmentations.)

2.3 Describing the regions

We store a simple description of each region's color, texture, and spatial characteristics. (See Fig. 2(f) for a visualization of the stored representation.)

Color and texture descriptors

The two dominant colors within a connected component are chosen by using the EM algorithm to fit a mixture of two Gaussians in the HSV cone. The details are as before except that in this case we restrict the covariances to be a constant times the identity matrix. Upon convergence, the two mean vectors are recorded as the dominant colors in the region. When the color distribution inside the HSV cone is in fact unimodal, both means become nearly coincident; we have not found it necessary to apply model selection between $K = 1$ and $K = 2$.

For each image region (blob) we store the mean texture descriptors (i.e., anisotropy, orientation, contrast) and the top two colors. We do not store the selected scale, since we want to be invariant to scales in the range $\sigma_k = 0, \ldots, 3.5$. Although polarity is used for scale selection, we discard it here, since in any textured or uniform region it is approximately zero by virtue of the scale selection process.

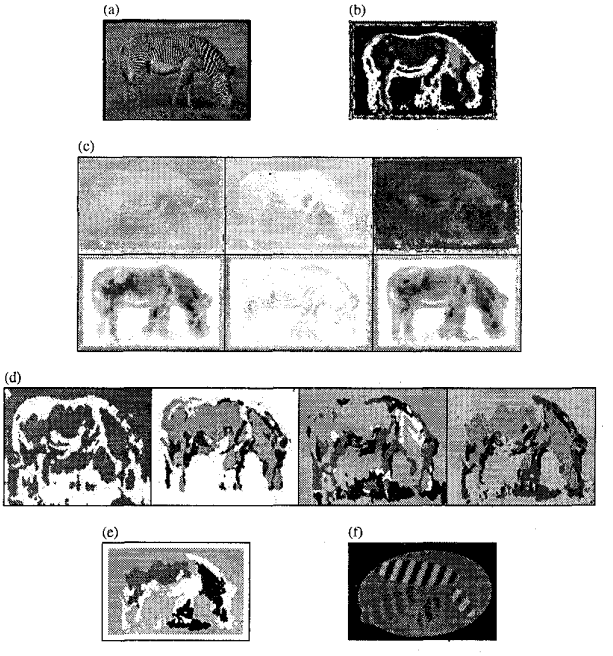


Figure 2. Creating the blobworld representation.
(a) Original image.
(b) Scale estimated using polarity. The values range from $\sigma = 0$ (black) to $\sigma = 3.5$ (white).
(c) The six components of the color/texture feature vectors, each bounded between 0 (white) and 1 (black). Top: the locally smoothed color-cone coordinates. Bottom: the texture coordinates; from left to right, ac, pc, and c. The zebra hide is highly anisotropic and in general has high texture contrast. The polarity is largest around the edges, where the shading gradient points primarily in one direction.
(d) The results of clustering these feature vectors into $K = 2, 3, 4, 5$ groups using EM to learn a mixture of Gaussians. Pixel cluster memberships are shown as one of up to five gray levels. The MDL principle suggests that the rightmost image ($K = 5$) provides the best segmentation of the data. Most noticeable in this segmentation are oriented texture, which is found throughout the zebra hide, and uniform or low-contrast texture, which accounts for most of the background.
(e) The segmentation for $K = 5$ after application of a 3×3 max-vote filter. Each connected component in this image which possesses an area greater than 2% of the total image area produces a blob.
(f) The blobworld representation. Each blob encodes summary information about the underlying color, texture and shape properties.

Spatial descriptors

The geometric descriptors of the blob are simply the centroid c and scatter matrix S of the blob region; the centroid provides a notion of position, while the scatter matrix provides an elementary shape description. In the querying process discussed in Section 3.1, centroid separations are expressed using Euclidean distance. The determination of the distance between scatter matrices is based on the three quantities $[\det(S)]^{1/2} = \sqrt{\kappa_1\kappa_2}$, $1 - \kappa_2/\kappa_1$, and θ. (κ_1 and κ_2 are the eigenvalues of S; θ is the argument of the principal eigenvector of S.) These three quantities represent approximate area, eccentricity, and orientation.

3 Image retrieval by querying

Anyone who has used a search engine, text-based or otherwise, is familiar with the reality of unwanted matches. Often in the case of text searches this results from the use of ambiguous keywords, such as "bank" or "interest" [27]. Unfortunately, with image queries it is not always so clear why things go wrong. Unlike with text searches, in which the user can see the features (words) in a document, none of the current content-based image retrieval systems allows the user to see exactly what the system is looking for in response to a similarity-based query. Simply allowing the user to submit an arbitrary image (or sketch) and set some abstract knobs without knowing how they relate to the input image in particular implies a degree of complexity that searching algorithms do not have. As a result, a query for a bear can return just about any object under the sun if the query is not based on image regions, the segmentation routine fails to "find" the bear in the submitted image, or the submitted image contains other distinctive objects. Without realizing that the input image was not properly processed, the user can only wonder what went wrong. In order to help the user formulate effective queries and understand their results, as well as to minimize disappointment due to overly optimistic expectations of the system, the system should display its representation of the submitted image and the returned images.

3.1 Querying in blobworld

In our system, the user composes a query by submitting an image and seeing its blobworld representation, selecting the blobs to match, and finally specifying the relative importance of the blob features. The user may also submit blobs from several different images. (For example, a query might be the disjunction of the blobs corresponding to airplanes in several images, in order to provide a query that looks for airplanes of several shades.)

We define an "atomic query" as one which specifies a particular blob to match (e.g., "like-blob-1"). A "compound query" is defined as either an atomic query or a conjunction or disjunction of compound queries ("like-blob-1 and like-blob-2"). In the future, we might expand this definition to include negation ("not-like-blob-1") and to allow the user to specify two blobs with a particular spatial relationship as an atomic query ("like-blob-1-left-of-blob-2").

Once a compound query is specified, we score each database image based on how closely it satisfies the compound query. The score μ_i for each atomic query (like-blob-i) is calculated as follows:

1. Find the feature vector v_i for the desired blob b_i. This vector consists of the stored color, texture, position, and shape descriptors.
2. For each blob b_j in the database image:
   (a) Find the feature vector v_j for b_j.
   (b) Find the Mahalanobis distance between v_i and v_j using the diagonal covariance matrix (feature weights) set by the user: $d_{ij} = \left[(v_i - v_j)^T \Sigma^{-1} (v_i - v_j)\right]^{\frac{1}{2}}$.
   (c) Measure the similarity between b_i and b_j using $\mu_{ij} = e^{-\frac{d_{ij}}{2}}$. This score is 1 if the blobs are identical in all relevant features; it decreases as the match becomes less perfect.
3. Take $\mu_i = \max_j \mu_{ij}$.

The compound query score for the database image is calculated using fuzzy-logic operations [13]. For example, if the query is "like-blob-1 and (like-blob-2 or like-blob-3)," the overall score for the image is $\min\{\mu_1, \max\{\mu_2, \mu_3\}\}$. The user can also specify a weighting σ_i for each atomic query. If "like-blob-i" is part of a disjunction in the compound query, the weighted score for atomic query i is $\mu_i' = \sigma_i \mu_i$; if it is in a conjunction, its weighted score is $\mu_i' = 1 - \sigma_i \cdot (1 - \mu_i)$.

We then rank the images according to overall score and return the best matches, indicating for each image which set of blobs provides the highest score; this information will help the user refine the query. After reviewing the query results, the user may change the weighting of the blob features or may specify new blobs to match.

3.2 Results

We have performed a variety of queries using a set of 2000 images from the commercial Corel stock photo collection. We used the following categories: African Specialty Animals; Air Shows; Arabian Horses; Bald Eagles; Bears; Canadian Rockies; Caribbean; Cheetahs, Leopards, Jaguars; China; Death Valley; Deserts; Elephants; Fields; France; Kenya; Night Scenes; Sheep; Sunsets; Tigers; and Wild Animals. Sample queries are shown in Figs. 3–4.

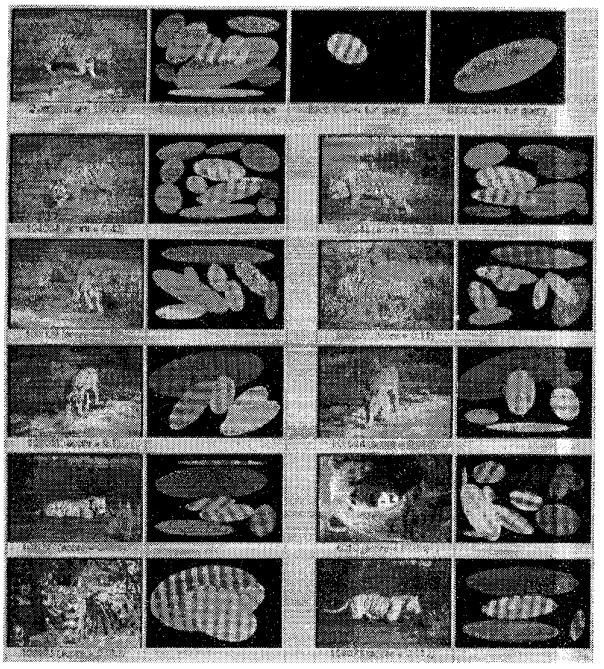


Figure 3. Blobworld query for tiger images. 28% of the top 50 images are tigers; tiger images make up 5% of the database.

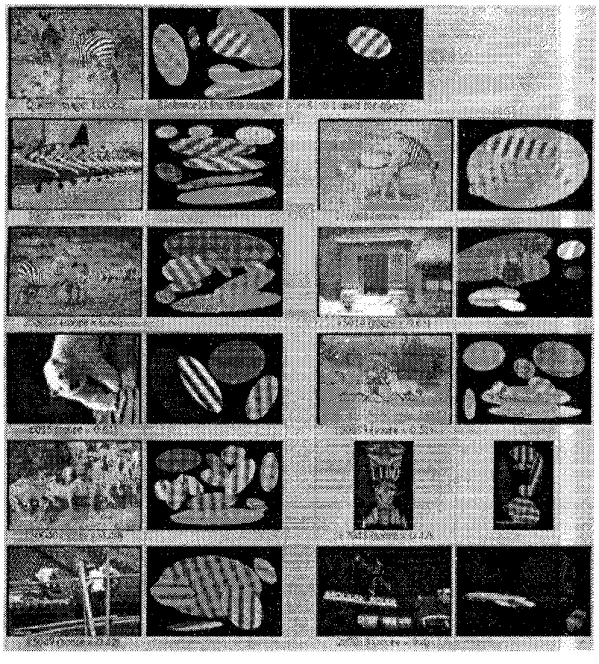


Figure 4. Blobworld query for zebra images. 24% of the top 50 images are zebras, while less than 2% of the images in the database are zebras.

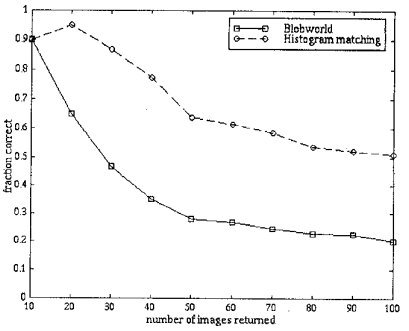


Figure 5. Tiger query performance.

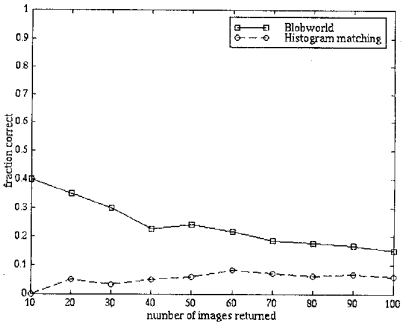


Figure 6. Zebra query performance.

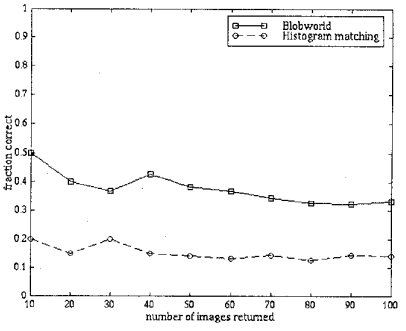


Figure 7. Airplane query performance.

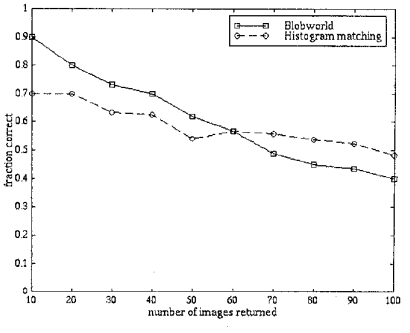


Figure 8. Sunset query performance.

3.2.1 Comparison to color histograms

We have compared our results to those obtained using color histogram matching, following the procedure of Swain and Ballard [24]. The color histogram for each image uses 8 divisions for the intensity axis and 16 for each opponent color axis. Given a query image with histogram Q_i, each database image (with histogram D_i) receives score $\sum_i |Q_i - D_i|$. As before, we rank the database images and return the best matches. Figures 5–8 show how the precision changes as more images are returned; the blobworld query results are better than the color histogram results, except for the tiger query. We believe the good color histogram results for the tiger query occur largely because of the limited nature of the test database; few non-tiger images in this collection have significant amounts of both orange and green. Adding pictures of, say, orange flowers in a field would degrade the color histogram performance without significantly affecting the blobworld performance.

4 Conclusions

We have proposed a new method which uses Expectation-Maximization on color and texture jointly to provide an image segmentation, as well as a new image representation (blobworld) which uses this segmentation and its associated descriptors to represent image regions explicitly. We have demonstrated a query mechanism that uses blobworld to retrieve images and help guide user queries.

Acknowledgments

We would like to thank David Forsyth, Joe Hellerstein, Ginger Ogle, and Robert Wilensky for useful discussions related to this work. This work was supported by an NSF Digital Library Grant (IRI 94-11334) and NSF graduate fellowships for Serge Belongie and Chad Carson.

References

[1] S. Ayer and H. Sawhney. Layered representation of motion video using robust maximum-likelihood estimation of mixture models and MDL encoding. In *Proc. Int. Conf. Comp. Vis.*, pages 777–784, 1995.

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

[3] 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.

[4] P. Enser. Query analysis in a visual information retrieval context. *J. Doc. and Text Management*, 1(1):25–52, 1993.

[5] W. Förstner. A framework for low level feature extraction. In *Proc. Eur. Conf. Comp. Vis.*, pages 383–394, 1994.

[6] D. Forsyth, J. Malik, and R. Wilensky. Searching for digital pictures. *Scientific American*, 276(6):72–77, June 1997.

[7] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[8] J. Gårding and T. Lindeberg. Direct computation of shape cues using scale-adapted spatial derivative operators. *Int. J. Comp. Vis.*, 17(2):163–191, Feb 1996.

[9] G. H. Granlund and H. Knutsson. *Signal Processing for Computer Vision*. Kluwer Academic Publishers, 1995.

[10] A. Gupta and R. Jain. Visual information retrieval. *Comm. Assoc. Comp. Mach.*, 40(5):70–79, May 1997.

[11] T. Hofmann, J. Puzicha, and J. M. Buhmann. Deterministic annealing for unsupervised texture segmentation. In *Proc. Int. Workshop on Energy Min. Methods in Comp. Vis. and Patt. Rec.*, pages 213–228, 1997.

[12] A. K. Jain and F. Farrokhnia. Unsupervised texture segmentation using Gabor filters. *Pattern Recognition*, 24(12):1167–1186, 1991.

[13] J.-S. Jang, C.-T. Sun, and E. Mizutani. *Neuro-Fuzzy and Soft Computing*. Prentice Hall, 1997.

[14] P. Kelly, M. Cannon, and D. Hush. Query by image example: the CANDID approach. In *SPIE Proc. Storage and Retrieval for Image and Video Databases*, pages 238–248, 1995.

[15] T. Leung and J. Malik. Detecting, localizing and grouping repeated scene elements from an image. In *Proc. Eur. Conf. Comp. Vis.*, pages 546–555, 1996.

[16] P. Lipson, E. Grimson, and P. Sinha. Configuration based scene classification and image indexing. In *Proc. IEEE Comp. Soc. Conf. Comp. Vis. and Pattern Recogn.*, pages 1007–1013, 1997.

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

[18] W. Niblack et al. The QBIC project: querying images by content using colour, texture and shape. In *SPIE Proc. Storage and Retrieval for Image and Video Databases*, pages 173–187, 1993.

[19] V. Ogle and M. Stonebraker. Chabot: Retrieval from a relational database of images. *IEEE Computer*, 28(9):40–48, Sep 1995.

[20] A. Pentland, R. Picard, and S. Sclaroff. Photobook: Content-based manipulation of image databases. *Int. J. Comp. Vis.*, 18(3):233–254, 1996.

[21] J. Ponce, A. Zisserman, and M. Hebert. *Object Representation in Computer Vision—II.* Number 1144 in LNCS. Springer, 1996.

[22] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[23] U. Shaft and R. Ramakrishnan. Data modeling and querying in the PIQ image DBMS. *IEEE Data Engineering Bulletin*, 19(4):28–36, Dec 1996.

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

[25] Y. Weiss and E. Adelson. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *Proc. IEEE Comp. Soc. Conf. Comp. Vis. and Pattern Recogn.*, pages 321–326, 1996.

[26] W. Wells, R. Kikinis, W. Grimson, and F. Jolesz. Adaptive segmentation of MRI data. In *Int. Conf. on Comp. Vis., Virtual Reality and Robotics in Medicine*, pages 59–69, 1995.

[27] D. Yarowsky. Word-sense disambiguation using statistical models of Roget's categories trained on large corpora. In *Proc. Int. Conf. Comp. Linguistics*, pages 454–460, 1992.