Region-Based Image Querying*

Chad Carson, Serge Belongie, Hayit Greenspan, and Jitendra Malik
Computer Science Division
University of California at Berkeley
Berkeley, CA 94720
{carson,sjb,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 an image representation which provides a transition from the raw pixel data to a small set of localized coherent regions in color and texture space. This so-called "blobworld" image description may be thought of as a summary representation *which captures the basic compositional features of the image. 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. 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

Over the past decade, large image collections have grown 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: users cannot create meaningful visual queries on an object level. While image database users would like to find images containing particular objects ("things"), most existing image retrieval systems search for images based on the low-level features ("stuff") in the images. In addition, query results are often unintuitive and give the user 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 picture of, say, a bear and retrieved very few bears in return.

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, queries in blobworld are queries for proto-objects—regions of coherent stuff—and are more meaningful to the user than simple stuff queries.

Blobworld uses the Expectation-Maximization algorithm to perform automatic segmentation based on image features. After the image is segmented into regions, a description of each region's color, texture, and spatial characteristics is produced. The user can access the regions directly in order to see the segmentation of the query image and specify what aspects of the image are central 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 have shown elsewhere [1] that the blobworld representation can be used to learn image categories in order to provide automatic scene classification.

We begin this paper with a discussion of the blobworld representation and its associated feature extraction algorithms. Next we use several examples to illustrate the process of querying in blobworld. We conclude with a discussion of system limitations and proposed future work.

2. The blobworld image representation

The "blobworld" representation is related to the notion of photographic or artistic scene composition. In the sense discussed in [15], the blobworld features constitute an example of a *summary representation* in that they are concise and relatively easy to process in a querying framework.

The blobworld representation 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 2D ellipses, or "blobs," each of which possesses a number of attributes. Typically the number of blobs is less than ten. Each blob represents a region of the image which is roughly homogeneous with respect to either color or texture. A blob is represented by the following attributes:

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

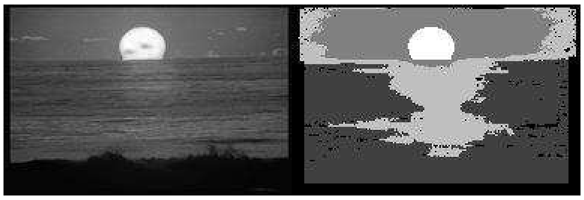

Figure 1. A sunset image and its segmented regions.

- centroid coordinates
- scatter matrix
- mean color within blob
- mean texture descriptors within blob

These four attributes constitute simple expressions of *position*, *shape*, *color*, and *texture*.

To illustrate a very simple example, consider the image containing a sunset shown in Fig. 1. The segmentation process produces four regions, or, to be exact, four connected components: one each for the sun, the sky, the water, and the sun's reflection in the water. Each pixel in each of the four regions possesses a color descriptor and a texture descriptor, each of which is three dimensional. These descriptors are pooled together inside each region and are thereafter summarized by their means. Each connected component itself, which represents the spatial extent of a given blob, is represented by its centroid and scatter matrix.

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 [12, 13, 16]. 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 [10], 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 [17]. 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.

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 [8].

In order to find distances in HSV space, we treat the space as a cone (see Fig. 2): for a given point $(h, s, v)^T$, 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. 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, this scheme ignores the fact that for moderate and large values and saturations, hue is more perceptually relevant than saturation and value.

[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.

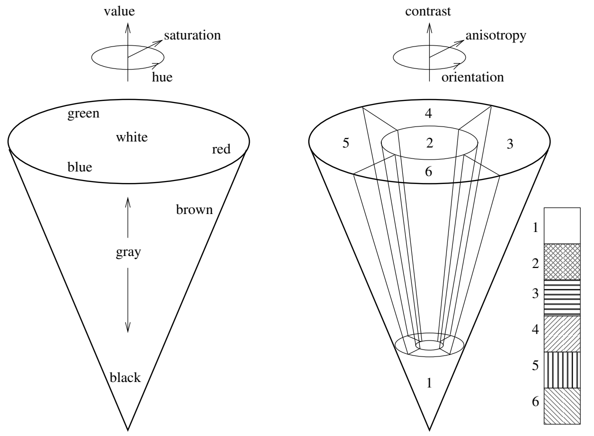

Figure 2. The color and texture cones, showing characteristic textures and a few sample color locations.

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 [7, 11] and the second-moment matrix [4, 6]. 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 between texture patterns which are singly oriented (*1D texture*) and those 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 [4, 6] 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 [2, 4]. 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 both eigenvalues are comparable, there is no preferred orientation. When both eigenvalues are negligible in magnitude, the local neighborhood is approximately a constant gray value, 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.

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. Moreover, how do we compare two different textured regions based on these descriptors? (In more prosaic terms, this is really a question of how to compare two symmetric positive semidefinite matrices.) Several factors need to be taken into account when trying to arrive at a satisfactory metric for this purpose. Note that when both eigenvalues are approximately equal, the angle information is meaningless. Also, the fact that orientation is a 180-degree periodic measure needs to be accounted for. Lastly, variations in anisotropy become meaningless when both eigenvalues are tiny.

In order to address these needs, we have found it beneficial to draw an analogy between the texture values and *hue, saturation, and value* as used in the HSV color-cone definition. In particular, we recast M as the following 3D vector:

$$\mathbf{m} = (ab\cos(2\phi), ab\sin(2\phi), b)$$

where $a = 1 - \lambda_2/\lambda_1$, and $b = \lambda_1 + \lambda_2$. 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 [2].) 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. (See Fig. 2.)

4. From pixels 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 on the order of five.

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 [3, 14]. An iterative procedure forms the basis of the algorithm: the Expectation or E-step 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 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

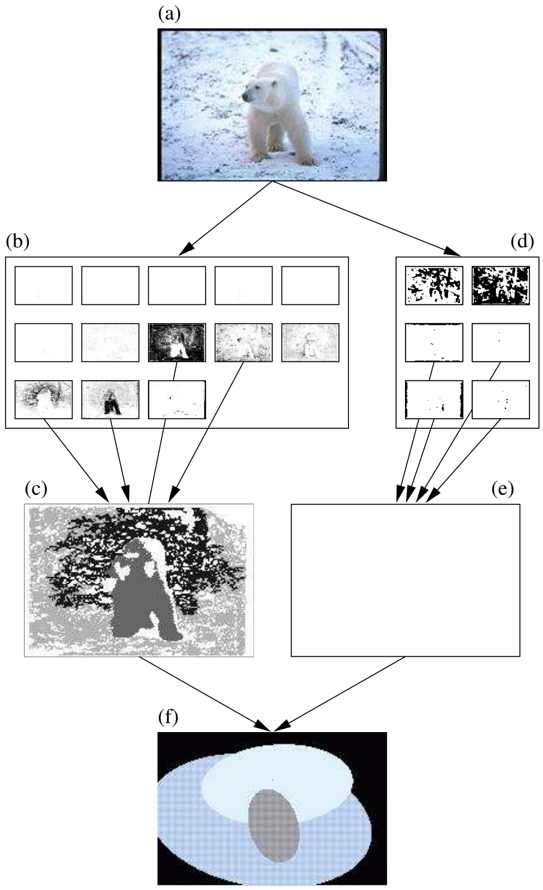

Figure 3. Finding blobs in an image. (a) Sample image. (b) The color channels; channels that initialize EM are indicated. (c) Support maps for the EM results; each gray level represents a different connected component. (d) The texture channels. The four 1D texture channels are allowed to contribute blobs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

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. 3(f) and Figs. 6–9).

4.1. Shape features

In addition to the mean color and texture in each region, simple spatial information is computed for each blob. As discussed in Section 2, the geometric descriptors of the blobs are simply the centroid $\mathbf{c}_i$ and scatter matrix S_i, calculated in the blob region $\mathcal{B}_i$ as

$$\mathbf{c}_i = \frac{1}{N} \sum_{\mathbf{x} \in \mathcal{B}_i} \mathbf{x}$$

and

$$S_i = \frac{1}{N} \sum_{\mathbf{x} \in \mathcal{B}_i} (\mathbf{x} - \mathbf{c}_i)(\mathbf{x} - \mathbf{c}_i)^T$$

The scatter matrix provides an elementary second-order shape description, while the centroid provides a notion of position. In the querying process discussed in Section 5.1, centroid separations are expressed using Euclidean distance. The determination of the distance between scatter matrices, which is slightly more complicated, is based on the three quantities $\beta = [\det(S_i)]^{1/2} = \sqrt{\lambda_1 \lambda_2}$, $\alpha = 1 - \lambda_2/\lambda_1$, and θ, again the argument of the principal eigenvector. These three quantities represent approximate area, anisotropy and orientation.

These preliminary features clearly do not encode all the spatial information about the blob. In the future, we plan to add other shape features, which we expect will provide better performance.

5. 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." [18] Unfortunately, with image queries it is not always so clear why things go wrong. Loosely speaking, the *word* and the *extracted image feature* are analogous entities, though as a community we must agree that there is a long way to go before this analogy can be considered to be fair to the word. Unlike text searches, in which the user can see the 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 to allow the user to submit an arbitrary image (or sketch) and set some abstract knobs without knowing what they mean to the input image in particular is to imply a degree of complexity in the searching algorithms which is for the most part not met. As a result, a query for a polar bear can return just about any object you wish to mention 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. It is therefore our conviction that alongside the submitted image, the user should be allowed to inspect a representation of the extracted features which are used in carrying out the search.

5.1. Retrieval in blobworld

In our system, the user composes a query by submitting an image to the feature extraction/segmentation algorithm in order to see its blobworld representation, selecting the blobs to match, and 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.)

Another way to compose queries would be to paint several blobs on the screen, specifying their color and texture. This would allow the user to search for images without first finding an image that contains the desired object.

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, e.g., "like-blob-1 and like-blob-2." We could expand this definition to allow compound queries to include negation, e.g., "not-like-blob-1," and to allow the user to specify two blobs with a particular spatial relationship as an atomic query, e.g., "like-blob-1-left-of-blob-2."

Once a compound query is specified, we rank the database images based on how closely they satisfy the compound query. For each image, a score is calculated by the following procedure:

- For each atomic query (like-blob-i):
  - Find the feature vector v_i for the desired blob. This vector consists of the three color cone coordinates, three texture cone coordinates, two centroid coordinates, and three shape descriptors.
  - For each blob b_j in the image:
    - Find the feature vector v_j for the image blob.
    - Find the Mahalanobis distance between blob i and blob j using the diagonal covariance matrix (feature weights) set by the user: $d_{ij} = (v_i - v_j)^T \Sigma^{-1} (v_i - v_j)$.
    - Measure the similarity between b_i and b_j using $\text{score}_{ij} = e^{-\frac{d_{ij}}{2}}$. This score is 1 if b_i and b_j are identical in all relevant features; it decreases as the match becomes less perfect.
  - Take $\text{score}_i = \max_j \text{score}_{ij}$.
- Combine the scores using fuzzy-logic operations [9] for each atomic query as specified in the compound query. For example, if the query is "like-blob-1 and

Figure 4. Sample images from the Corel database.

(like-blob-2 or like-blob-3),” the overall score for the image is $\min\{\text{score}_1, \max\{\text{score}_2, \text{score}_3\}\}$.

We then rank the images according to overall score and return the top matches, indicating for each image which set of blobs provided 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.

6. Results

We have performed a variety of queries using a database of 1600 images from the commercial Corel stock photo collection. The categories we used for these experiments include airplanes, flowers, eagles, people, mountains, deserts, fields, sunsets, night scenes, and a wide variety of animals. (See Fig. 4.) Sample queries are shown in Figs. 6–9. The performance for these and other queries is shown numerically in Fig. 5.

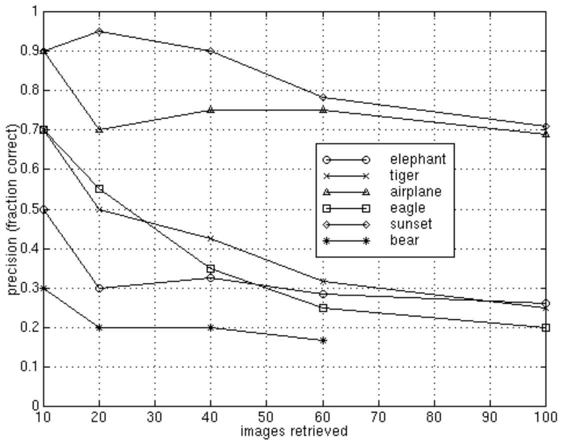

Figure 5. Precision (fraction correct) vs. number of images retrieved for several queries. The downward slope indicates that highly ranked images tend to be correct; for most queries more than half of the top ten images are correct, but the percentage drops as we include more (lower-ranked) images. Note the poor results for the bear query.

7. Conclusions

We make two primary contributions in this paper:

1. A new segmentation procedure for finding regions of coherent color and texture using the EM algorithm. The procedure simultaneously returns a set of descriptors for the color, texture, position, and shape of each region.
2. An implemented procedure for composing queries where an atomic query looks for a region of desired color, texture, position, and shape. The user can specify the tolerance for each specified feature.

We have found that this approach works very well when looking for certain classes of objects—those that have distinctive color or texture. When the color/texture description of an object is not very specific, for example brown regions when searching for brown bears, the approach is less useful. This is not a very surprising conclusion, but it does point out the most important problems for future work. Characterizing shape is perhaps the most important; associated with that is the problem of improved image segmentation. A promising start on shape characterization has been made by Forsyth and Fleck [5] in their work on finding people and horses using body plans.

8. Acknowledgments

We would like to thank David Forsyth, Joe Hellerstein, and Robert Wilensky for useful discussions related to this work.

References

[1] S. Belongie, C. Carson, H. Greenspan, and J. Malik. Recognition of images in large databases using a learning framework. Technical Report 97-939, U.C. Berkeley CS Department, 1997.

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

[5] D. Forsyth and M. Fleck. Body plans. In *Proc. Conf. Comp. Vis. Patt. Rec.*, 1997. to appear.

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

[7] 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.

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

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

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

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

[12] 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.

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

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

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

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

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

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

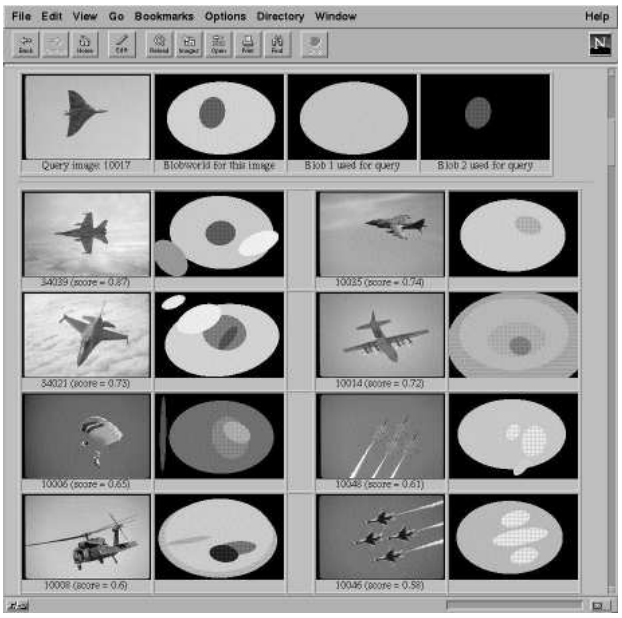

Figure 6. Query for airplane images, using a sky blob (blue) and an airplane blob (gray/black).

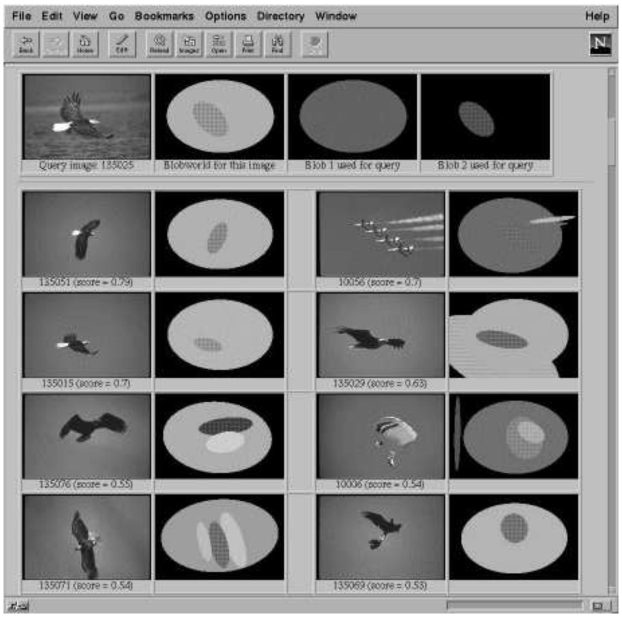

Figure 7. Query for eagle images. (Compare to the airplane query results.)

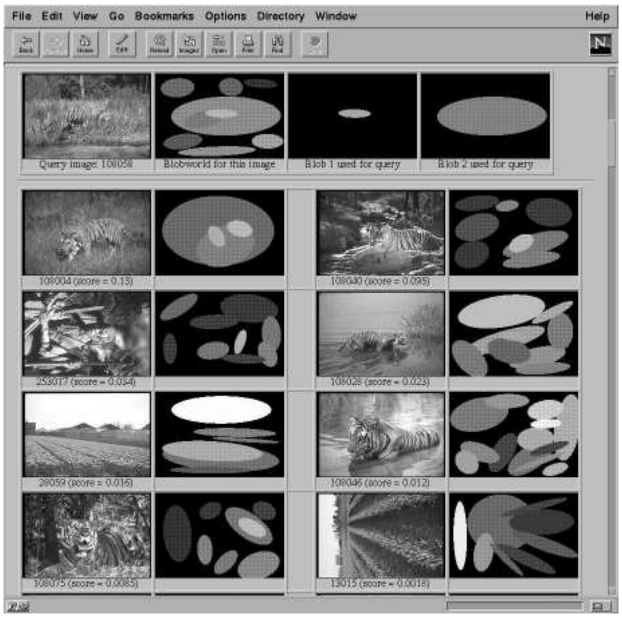

Figure 8. Query for tiger images, using a tiger blob (orange) and a grass blob (green).

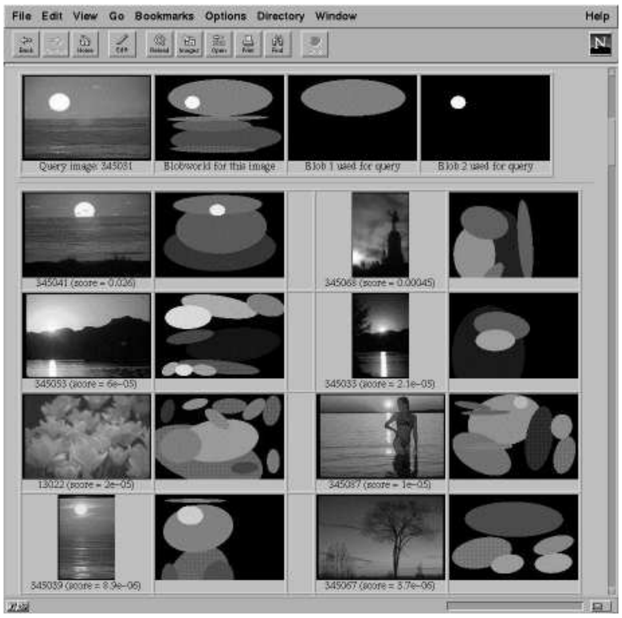

Figure 9. Query for sunset images, using a sky blob (red-orange) and a sun blob.