Blobworld: Image Segmentation Using Expectation-Maximization and Its Application to Image Querying

Chad Carson, *Member, IEEE*, Serge Belongie, *Member, IEEE*,
Hayit Greenspan, *Member, IEEE*, and Jitendra Malik, *Member, IEEE*

Abstract—Retrieving images from large and varied collections using image content as a key is a challenging and important problem. We present a new image representation that provides a transformation from the raw pixel data to a small set of image regions that are coherent in color and texture. This "Blobworld" representation is created by clustering pixels in a joint color-texture-position feature space. The segmentation algorithm is fully automatic and has been run on a collection of 10,000 natural images. We describe a system that uses the Blobworld representation to retrieve images from this collection. An important aspect of the system is that 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, query results from these systems can be inexplicable, despite the availability of knobs for adjusting the similarity metrics. By finding image regions that roughly correspond to objects, we allow querying at the level of objects rather than global image properties. We present results indicating that querying for images using Blobworld produces higher precision than does querying using color and texture histograms of the entire image in cases where the image contains distinctive objects.

Index Terms—Segmentation and grouping, image retrieval, image querying, clustering, Expectation-Maximization.

✦

---

1 INTRODUCTION

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

- While users generally want to find images containing particular objects ("things") [9], [13], 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 but in return has retrieved many irrelevant images and very few pictures of bears.

- *C. Carson is with the Computer Science Division, 387 Soda Hall #1776, University of California at Berkeley, Berkeley, CA 94720-1776. E-mail: carson@eecs.berkeley.edu.*
- *S. Belongie is with the Department of Computer Science and Engineering, AP&M Building, Room 4832, University of California, San Diego, La Jolla, CA 92093-0114. E-mail: sjb@cs.ucsd.edu.*
- *H. Greenspan is with the Department of Biomedical Engineering, Faculty of Engineering, Tel-Aviv University, Ramat-Aviv, Tel-Aviv 69978 Israel. E-mail: hayit@eng.tau.ac.il.*
- *J. Malik is with the Computer Science Division, 725 Soda Hall, University of California at Berkeley, Berkeley, CA 94720-1776. E-mail: malik@cs.berkeley.edu.*

In this paper, we present "Blobworld," a new framework for image retrieval based on segmentation into regions and querying using properties of these regions. The regions generally correspond to objects or parts of objects. While Blobworld does not exist completely in the "thing" domain, it recognizes the nature of images as combinations of objects, and querying in Blobworld is more meaningful than it is with simple "stuff" representations.

Image segmentation is a difficult problem. Segmentation algorithms inevitably make mistakes, causing some degradation in performance of any system that uses the segmentation results. As a result, designers of image retrieval systems have generally chosen to use global image properties, which do not depend on accurate segmentation. However, segmenting an image allows us to access the image at the level of *objects*. We believe this ability is critical to image retrieval and to progress in object recognition, in general. We have developed a segmentation algorithm that, while imperfect, provides segmentations that are good enough to yield improved query performance compared to systems that use global properties.

In order to segment each image automatically, we model the joint distribution of color, texture, and position features with a mixture of Gaussians. We use the Expectation-Maximization (EM) algorithm [8] to estimate the parameters of this model; 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 and texture 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 also sees the Blobworld representation of each retrieved image; this information assists greatly in refining the query.
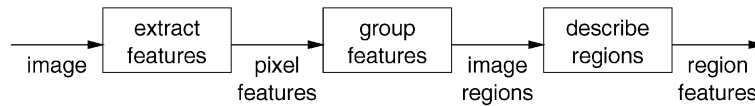
Fig. 1. The stages of Blobworld processing: From pixels to region descriptions.

We begin this paper by briefly discussing the current state of image retrieval. In Sections 2 and 3, we describe the feature extraction and segmentation algorithm. In Section 4, we discuss the descriptors assigned to each region. In Section 5, we present a query system based on Blobworld, as well as results from queries in a collection of 10,000 highly varied natural images. We conclude with a brief discussion of our approach and some proposed directions for future work.

Portions of this work have been published in [4], [6], [7].

1.1 Related Work

The best-known image database system is IBM's Query by Image Content (QBIC) [10], 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 recent versions of QBIC [2] contain simple automated segmentation facilities. Photobook [32] incorporates more sophisticated representations of texture and a degree of automatic segmentation. Other examples of systems that identify materials using low-level image properties include Virage [17], VisualSEEk [39], Candid [24], and Chabot [30].

Color histograms [42], [43] are commonly used in image retrieval systems and have proven useful; however, this global characterization lacks information about how the color is distributed spatially. Several researchers have attempted to overcome this limitation by incorporating spatial information in the descriptor. Stricker and Dimai [41] store the average color and the color covariance matrix within each of five fuzzy image regions. Huang et al. [20] store a "color correlogram" that encodes the spatial correlation of color-bin pairs. Smith and Chang [40] store the location of each color that is present in a sufficient amount in regions computed using histogram backprojection.

Lipson et al. [26] retrieve images based on spatial and photometric relationships within and across simple image regions. Little or no segmentation is done; the regions are derived from low-resolution images. Jacobs et al. [21] use multiresolution wavelet decompositions to perform queries based on iconic matching.

Some of these systems encode information about the spatial distribution of color features, and some perform simple automatic or manually-assisted segmentation. However, none provides the level of automatic segmentation and user control necessary to support object queries in a very large image collection. These approaches generally work well in a query-by-example task when the entire scene is distinctive and relevant; they are not suited to the task of querying for general objects such as animals, where large parts of the scene are irrelevant.

Our approach is most similar to Ma and Manjunath [27] who perform retrieval based on segmented image regions. Their segmentation algorithm includes an optional manual region-pruning step, and the user must specify the expected number of regions. Whether this is a pro or a con is not clear; it is still an open question what effect the number of regions has on system performance.

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 the case of natural images: the shape, size, and color of objects like tigers and airplanes are quite variable, and segmentation is imperfect. Clearly, classical object recognition does not apply. More recent techniques [33] 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.

Our approach to segmentation uses the EM algorithm to estimate the parameters of a mixture of Gaussians model of the joint distribution of pixel color and texture features. This approach is related to earlier work using EM and/or the Minimum Description Length (MDL) principle to perform segmentation based on motion [3], [44] or scaled intensities [45]. Related approaches such as deterministic annealing [34] and classical clustering [22] have been applied to texture segmentation without color. Panjwani and Healey [31] have performed segmentation using a Markov random field color texture model.

2 FEATURE EXTRACTION

Creating the Blobworld representation of an image involves three steps (see Fig. 1):

1. Select an appropriate scale for each pixel and extract color, texture, and position features for that pixel at the selected scale.
2. Group pixels into regions by modeling the distribution of pixel features with a mixture of Gaussians using Expectation-Maximization.
3. Describe the color distribution and texture of each region for use in a query.

Fig. 2 illustrates these steps for a sample image.

2.1 Extracting Color Features

Each image pixel has a three-dimensional color descriptor in the L*a*b* color space. This color space is approximately perceptually uniform; thus, distances in this space are meaningful [47]. We smooth the color features as discussed in Section 2.3 in order to avoid oversegmenting regions such as tiger stripes based on local color variation; otherwise, each stripe would become its own region.

2.2 Extracting Texture Features

Texture is a well-researched property of image regions and many texture descriptors have been proposed, including multiorientation filter banks [28] and the second-moment matrix [11], [15]. We will not elaborate here on the classical approaches to texture segmentation and classification, both
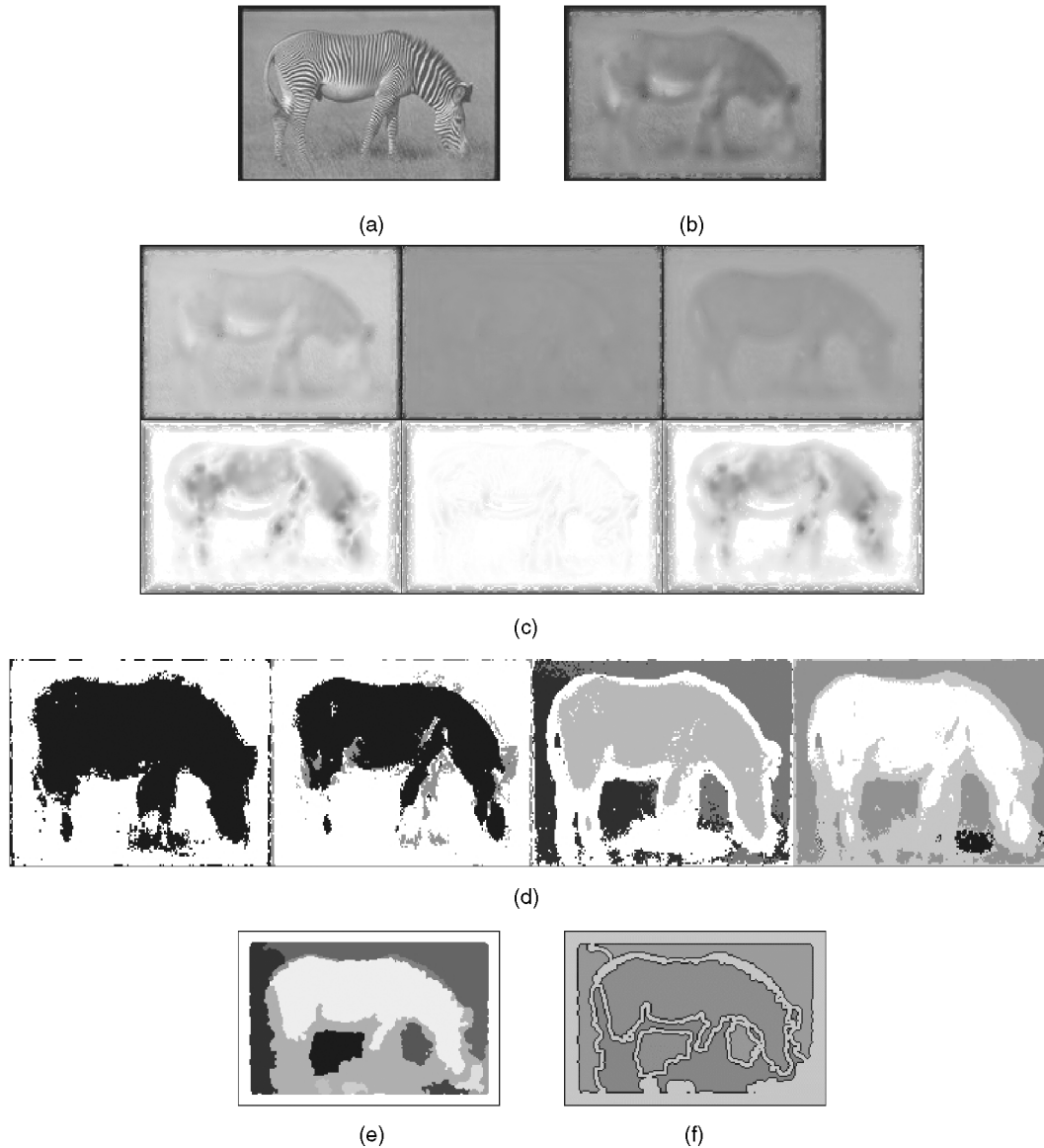
Fig. 2. Creating the Blobworld representation. (a) Original image. (b) The image after spatially variant smoothing at the selected scale. Note that the zebra stripes have been smoothed away and replaced by gray. (c) The six components of the color/texture feature vectors. The top images represent the three locally smoothed L*a*b* color coordinates. The bottom images represent the coordinates in texture space; from left to right, we have anisotropy, polarity, and contrast, ranging from 0 (white) to 1 (black). Note that the zebra hide is highly anisotropic (oriented) and has high texture contrast. Note also that the striped regions are roughly uniform in each of the six features. (d) The results of clustering the feature vectors into $K = 2, 3, 4, 5$ Gaussian clusters using EM. Pixel cluster memberships are shown as one of up to five gray levels. Application of the MDL principle suggests that the $K = 4$ image (third from the left) provides the best segmentation of the data. (e) The segmentation for $K = 4$ (as chosen by MDL) after postprocessing. (f) The Blobworld representation of the image. Each region with an area greater than 1 percent of the total image area yields a blob.

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.

Whereas 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 multiorientation filter banks such as

steerable filters; we chose a simpler method that is sufficient for our purposes. The second requirement, which may be thought of as the problem of *scale selection*, has not received 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 L* component, 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 pre-processing 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

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

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 working with the raw entries in M_σ, it is more common to deal with its eigenstructure [5], [11]. Consider a fixed scale σ 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.

2.2.1 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 [11], [15] 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

$$p_\sigma = \frac{|E_+ - E_-|}{E_+ + E_-}$$

The definitions of E_+ and E_- are

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

and

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

1. Strictly speaking, (1) is not a convolution since $\sigma(x,y)$ is spatially variant.
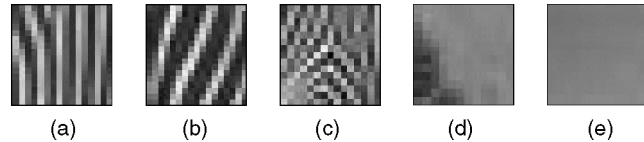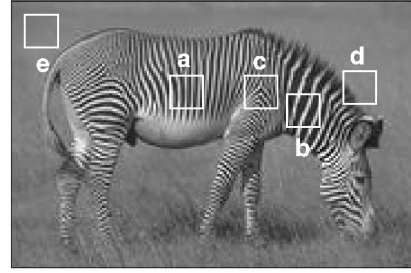2. Polarity is related to the *quadrature phase* as discussed in [14], [16].

Fig. 3. Five sample patches from a zebra image. Both (a) $\sigma = 1.5$ and (b) $\sigma = 2.5$ have stripes (1D flow) of different scales and orientations, (c) is a region of 2D texture with $\sigma = 1.5$, (d) contains an edge with $\sigma = 0$, and (e) is a uniform region with $\sigma = 0$.

where $[\cdot]_+$ and $[\cdot]_-$ are the rectified positive and negative parts of their argument and $\hat{n}$ is a unit vector perpendicular to ϕ. We can think of E_+ and E_- as measures of how many gradient vectors in the window $G_\sigma(x,y)$ are on the "positive side" and "negative side" of the dominant orientation, respectively. Note that p_σ ranges from 0 to 1. (A similar measure is used in [25] to distinguish a flow pattern from an edge.)

The polarity p_σ varies as the scale σ changes; its behavior in typical image regions can be summarized as follows (see Fig. 3):

- **Edge**. The presence of an edge is signaled by p_σ holding values close to 1 for all σ.
- **Texture**. In regions with 2D texture or 1D flow, p_σ decays with σ: as the window size increases, pixels with gradients in multiple directions are included in the window, so the dominance of any one orientation decreases.
- **Uniform**. When a neighborhood possesses a constant intensity, 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 $\tilde{p}_{\sigma_k}(x,y)$. For each pixel (x,y), we select the scale $\sigma^*(x,y)$ as the first value of $\sigma_k(x,y)$ for which the difference between values of polarity at successive scales $(\tilde{p}_{\sigma_k} - \tilde{p}_{\sigma_{k-1}})$ is less than 2 percent. By this process, 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. Because we stop at $\sigma_k = 3.5$, the largest period we can detect is approximately 10 pixels. Note that in uniform regions the selected scale is not meaningful and is set to zero. We declare a region to be uniform if its mean contrast across scale is less than 0.1.
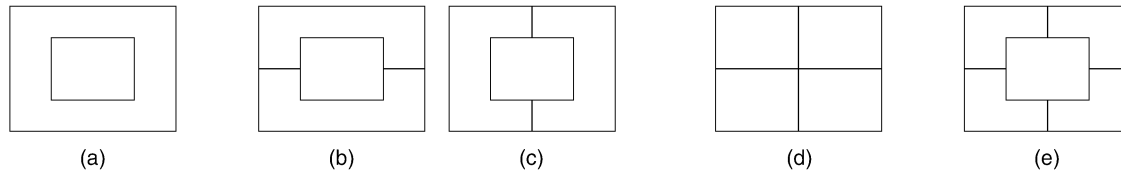
$$\begin{array}{ccccc} \text{(a)} & \text{(b)} & \text{(c)} & \text{(d)} & \text{(e)} \end{array}$$

Fig. 4. Windows for initializing the EM means. For a given K, each of the K initial means is found by averaging the feature vectors in one of the K windows. On subsequent restarts, Gaussian noise is added to each mean to choose the initial EM means. (When $K = 3$, the two sets of windows are used for alternate restarts.) Because the images in our collection generally follow the conventions of photographic composition, these window arrangements tend to initialize one mean corresponding roughly to the dominant object in the image and the other means corresponding to other objects or background regions. (a) K = 2, (b) K = 3, (c) K = 3, (d) K = 4, and (e) K = 5.

Another method of scale selection that has been proposed [15] 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.

2.2.2 Texture Features

Once a scale σ^* is selected for each pixel, that pixel is assigned three texture descriptors. The first is the polarity at that scale, $p = p_{\sigma^*}$. 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 [15].

2.3 Combining Color, Texture, and Position Features

The final color/texture descriptor for a given pixel consists of six values: three for color and three for texture. The three color components are the L*a*b* 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 since they are meaningless in regions of low contrast. In effect, a given textured patch in an image first has its texture properties extracted and then is replaced by a smooth patch of averaged color (see Fig. 2b). In this manner, the color and texture properties in a given region are decoupled; for example, a zebra becomes 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 orientation and scale.

Finally, we append the (x, y) position of the pixel to the feature vector. Including the position generally decreases oversegmentation and leads to smoother regions. As seen in Section 3.3, large, uniform background areas in the image are sometimes arbitrarily split into two pieces due to the use of position as a feature. On the whole, however, including position yields better segmentation results than excluding it.

3 GROUPING PIXELS INTO REGIONS

Once an image has been processed using the above feature extraction scheme, the result is a large set of feature vectors,

which we may regard as points in an eight-dimensional feature space. In order to divide these points into groups, we make use of the Expectation-Maximization (EM) algorithm [8] to determine the maximum likelihood parameters of a mixture of K Gaussians in the 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 Gaussian cluster 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. In the case where EM is applied to learning the parameters for a mixture of Gaussians, it turns out that both steps can be combined into a single update step, as we shall see in the following description.

Assume that we are using K Gaussians in the mixture model. (We will return to the matter of choosing K shortly.) The form of the probability density is as follows:

$$f(x|\Theta) = \sum_{i=1}^{K} \alpha_i f_i(x|\theta_i),$$

where x is a feature vector, the α_k's represent the *mixing weights* ($\sum_{i=1}^{K} \alpha_i = 1$), Θ represents the collection of parameters $(\alpha_1, \ldots, \alpha_K, \theta_1, \ldots, \theta_K)$, and f_i is a multivariate Gaussian density parameterized by θ_i (i.e., μ_i and Σ_i):

$$f_i(x|\theta_i) = \frac{1}{(2\pi)^{d/2} \det \Sigma_i^{1/2}} e^{-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)}$$

with $d = 8$, the dimension of the feature space.

The first step in applying the EM algorithm to the problem at hand is to initialize K mean vectors $\mu_1, \ldots, \mu_K$ and K covariance matrices $\Sigma_1, \ldots, \Sigma_K$ to represent each of the K groups. We set the initial covariances to be the identity matrix. We initialize the means by finding the average feature vector in each of K windows in the image (see Fig. 4). On subsequent restarts of the EM iteration, we add a small amount of noise to each mean. We have found that using this data-driven initialization yields slightly better results than random initialization, because the initial Gaussians better cover the occupied regions of the feature space; however, the exact initialization is not critical to the success of the segmentation algorithm.

The update equations take on the following form:

$$\alpha_i^{new} = \frac{1}{N} \sum_{j=1}^{N} p(i|x_j, \Theta^{old})$$

---

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

$$\mu_i^{new} = \frac{\sum_{j=1}^{N} x_j p(i|x_j, \Theta^{old})}{\sum_{j=1}^{N} p(i|x_j, \Theta^{old})}$$

$$\Sigma_i^{new} = \frac{\sum_{j=1}^{N} p(i|x_j, \Theta^{old})(x_j - \mu_i^{new})(x_j - \mu_i^{new})^T}{\sum_{j=1}^{N} p(i|x_j, \Theta^{old})},$$

where N is the total number of feature vectors, i.e., the number of pixels, and $p(i|x_j, \Theta)$ is the probability that Gaussian i fits the pixel x_j, given the data Θ:

$$p(i|x_j, \Theta) = \frac{\alpha_i f_i(x_j|\theta_i)}{\sum_{k=1}^{K} \alpha_k f_k(x_j|\theta_k)}.$$

This 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 are suited to our problem since many plausible feature clusters require elongated covariance shapes, e.g., the shades of gray along the L* axis of the color space.

The above update equations are repeated until the log likelihood

$$\log \mathcal{L}(\Theta|\mathcal{X}) = \log \prod_{k=1}^{N} f(x_k|\Theta)$$

increases by less than 1 percent from one iteration to the next. (If this does not happen within 10 iterations, a stop is forced.) We repeat this iteration four times, adding Gaussian noise (with $\sigma = 1.5$) to the initial means each time; this allows us to avoid shallow local maxima.

3.1 Model Selection

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. (Note that each of these color/texture groups may include several spatially disjoint regions 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 [35], [36] to select among values of K. This can be operationalized as follows [35], [38]: choose K to maximize

$$\log \mathcal{L}(\Theta|\mathcal{X}) - \frac{m_K}{2} \log N,$$

where m_K is the number of free parameters needed for a model with K mixture components. In the case of a Gaussian mixture with full covariance matrices, we have

$$m_K = (K-1) + Kd + K\frac{d(d+1)}{2}.$$

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.

It is important to note that we are not trying to find the "true" K or "true" Gaussian distribution. There is no such thing, the images were not produced by drawing pixels from a mixture of Gaussian distributions. Rather, we are trying to choose clusters that allow us to segment the images effectively.

3.2 Postprocessing

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. 2d) and running a connected-components algorithm to find image regions. (There may be more than K of these regions.)

While spatially averaging the color features allows us to group, for example, black and white zebra stripes into one region, it also causes object boundaries to be blurred in the color-feature image. As a result, boundaries in the raw cluster-membership image do not align exactly with boundaries in the original image. In order to mitigate this problem, we perform a simple postprocessing step:

1. Find the color histogram of each region (minus its boundary) using the *original* pixel colors (before smoothing).
2. For each pixel (in color bin i) on the boundary between two or more regions, reassign it to the region whose histogram value i is largest. (This is the maximum likelihood estimate of the region membership based on the regions' color distribution.)

We iterate this process four times. (We have found that the boundaries from the EM clustering are rarely misaligned with the true object boundaries by more than four pixels.)

Finally, to enforce a minimal amount of spatial smoothness in the final segmentation, we apply a 3×3 maximum-vote filter[4] to the output of the postprocessing step (see Fig. 2e).

The segmentation process (mostly Matlab code) takes 5 to 7 minutes per image on a 300MHz Pentium II. We process the images offline and on multiple machines in parallel.

3.3 Segmentation Results

Figs. 5 and 6 show the final segmentation of 80 randomly selected images. (Segmentations for all 10,000 images may be seen at http://elib.cs.berkeley.edu/segmentation.) The segmentation results are generally good, but several kinds of "errors" may be seen in a few of the images:

*   Large background areas may be arbitrarily split into two regions due to the use of position in the feature vector.
*   The region boundaries sometimes do not follow object boundaries exactly, even when the object boundary is visually quite apparent. This occurs because the color feature is averaged across object boundaries. This problem is mitigated but not entirely eliminated by the postprocessing described in Section 3.2.
*   In some cases, the object of interest is missed, split, or merged with other regions because it is not visually distinct. This occurs in such cases as camouflaged cheetahs and elephants which merge with the background. These are not actually errors of the segmentation algorithm, since finding the "correct" segmentation would rely on high-level semantic knowledge.

4. This "filter" assigns its output value as that value that occurs most often in the 3×3 window.
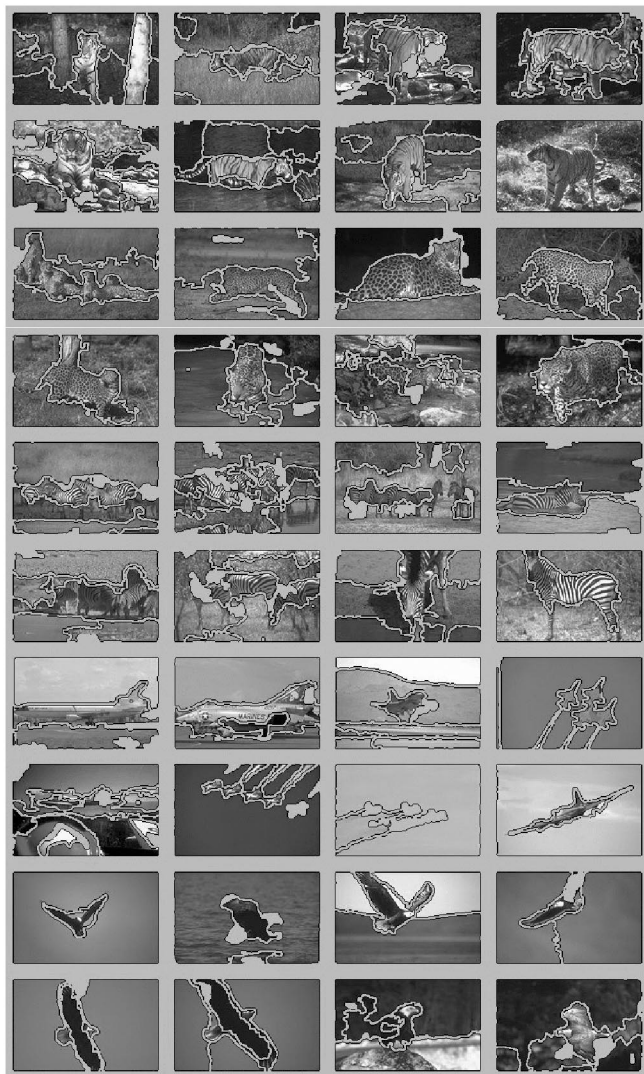
Fig. 5. Segmentation of randomly selected images of tigers, cheetahs/leopards/jaguars, zebras, airplanes, and bald eagles. Boundaries and regions smaller than 1 percent of the image (which do not become blobs) are shown in gray.
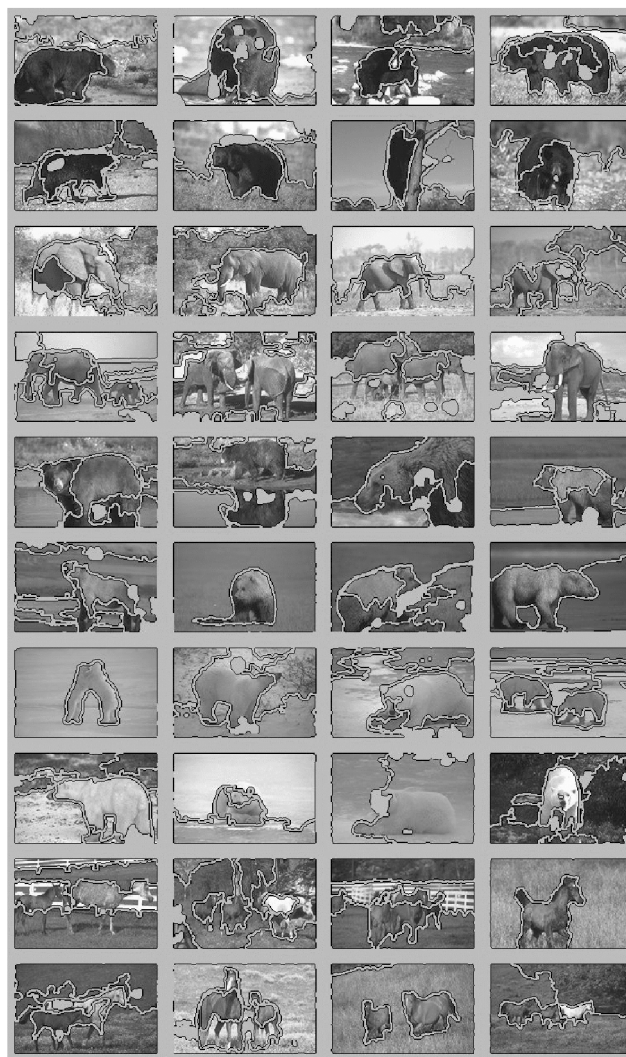

Fig. 6. Segmentations of randomly selected images of black bears, elephants, brown bears, polar bears, and brown horses. Boundaries and regions smaller than 1 percent of the image (which do not become blobs) are shown in gray.

- In rare cases, a visually distinct object is simply missed. This error occurs mainly when no initial mean falls near the object's feature vectors.

The first effect does not hamper query performance; oversegmenting uniform background regions just yields two blobs with similar color and texture descriptors, and the shape of background blobs is not important. Missing the true object boundary simply perturbs the region's color and texture descriptors slightly. Missing hard-to-find objects may cause those images not to be ranked highly in a query; however, users are generally looking for good examples of an easily visible object, so the practical effect on query performance is minimal. The last error truly hampers performance, because it causes good images to be missed in the query. Fortunately, this error occurs only rarely, less than 0.1 percent of the time, with smaller objects more likely to be missed than larger ones.

4 DESCRIBING THE REGIONS

We store a simple description of each region's color and texture characteristics.

In order to represent the color distribution of each region, we store the color histogram of the pixels in the region. This histogram is based on bins with width 20 in each dimension of L*a*b* space. This spacing yields five bins in the L* dimension and ten bins in each of the a* and b* dimensions, for a total of 500 bins. However, not all of these bins are valid; the gamut corresponding to $0 \leq R, G, B \leq 1$ contains only 218 bins that can be filled.

To match the color of two regions, we use the quadratic distance between their histograms $\mathbf{x}$ and $\mathbf{y}$ [18]:

$$d_{hist}^2(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^{\mathbf{T}} \mathbf{A} (\mathbf{x} - \mathbf{y}),$$

where $\mathbf{A} = [a_{ij}]$ is a symmetric matrix of weights between 0 and 1 representing the similarity between bins i and j based on the distance between the bin centers; neighboring bins have a weight of 0.5. This distance measure allows us to give a high score to two regions with similar colors, even if the colors fall in different histogram bins.

For each blob, we store the mean texture contrast and anisotropy. The distance between two texture descriptors is

defined as the Euclidean distance between their respective values of contrast and anisotropy × contrast. (Anisotropy is modulated by contrast because it is meaningless in areas of low contrast.) We do not include polarity in the region description because it is generally large only along edges; it would not help us distinguish among different kinds of regions.

5 IMAGE RETRIEVAL BY QUERYING

In the past few years, a variety of image retrieval systems have become available. Most of these systems operate in a similar way: the user performs a query by choosing an image which is somewhat similar to the desired image (or by submitting a sketch of the desired image) and setting a few "knobs" to specify which properties (e.g., overall color, overall texture, composition) are important to the query. Upon seeing the query results, the user may adjust the knobs and submit a new query based on the original image or one of the returned images. In a few systems, the user may also label the retrieved images as good or bad matches in order to provide more information to the retrieval algorithm; this is called relevance feedback [19], [29].

Two major shortcomings of such interfaces are a lack of user control and the absence of information about the computer's view of the image. Unlike with text searches, in which the user can see the features (words) in a document, none of the current image retrieval systems allows the user to see exactly what the system is looking for in response to a query. As a result, a query for a bear can return just about anything if the query is not based on image regions, the segmentation fails to "find" the bear in the submitted image, or the submitted image contains other distinctive objects. Without knowing 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, we believe the system should display its representation of the submitted and returned images and should allow the user to specify which aspects of that representation are relevant to the query.

5.1 Querying in Blobworld

In our system, the user composes a query by submitting an image to the segmentation/feature extraction algorithm in order to see its Blobworld representation, selecting the blobs to match, and finally specifying the relative importance of the blob features. (Query performance is robust to changes in the blob and feature weights; perturbing the weights usually changes the query results only slightly.)

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"). The user may also 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) with feature vector $\mathbf{v}_i$ is calculated as follows:

1. For each blob b_j in the database image (with feature vector $\mathbf{v}_j$):
   a. Find the Mahalanobis distance between $\mathbf{v}_i$ and $\mathbf{v}_j$: $d_{ij} = (\mathbf{v}_i - \mathbf{v}_j)^{\mathbf{T}} \Sigma (\mathbf{v}_i - \mathbf{v}_j)$.
   b. Measure the similarity between the two blobs 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.
2. Take $\mu_i = \max_j \mu_{ij}$.

The matrix Σ is block diagonal. The block corresponding to the texture features is an identity matrix, weighted by the texture weight set by the user. The block corresponding to the color features is the $\mathbf{A}$ matrix used in finding the quadratic distance, weighted by the color weight set by the user.

The compound query score for the database image is calculated using fuzzy-logic operations [23]. 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 provided the highest score; this information helps 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 and then issue a new query.

5.1.1 Including the Background

In many cases, desired images are characterized by one important object and a distinctive background (e.g., an eagle in the sky). In order to facilitate such queries, we allow the user to choose "background" rather than a second blob. When this option is used, the score for each blob in the database image is based on the distance between the query blob and the database blob as well as the distance between the color histogram of the complement of the query blob (i.e., all pixels not in the query blob) and the complement of the database blob. Note that this is quite different from matching global histograms of the entire image, as we are still looking for regions of coherent color and texture.

5.2 Results

We have performed a variety of queries using a set of 10,000 images from the commercial Corel stock photo collection. The query system is online at http:// elib.cs.berkeley.edu/photos/blobworld. Source code for segmentation and querying is also available at http:// dlp.cs.berkeley.edu/src/blobworld/. Sample queries are shown in Figs. 7, 8, 9, and 10. Queries on 10,000 images require 10 to 15 seconds for global histogram and single-blob queries, 20 to 25 seconds for two-blob queries, and 35 to 55 seconds for blob+background queries. (The Web server is a dual-processor 167 MHz UltraSparc.) Else-where, we describe an indexing scheme which provides faster retrieval with only a small reduction in precision [7].

5.3 Comparison to Global Histograms

We expected that Blobworld querying would perform well in cases where a distinctive object is central to the query. In

Fig. 7. Blobworld query for tiger images using two blobs. The overall weights are 1.0 for the tiger blob and 0.5 for the grass blob. For both blobs, the color weight is 1.0 and the texture weight is 0.5.

Fig. 9. Blobworld query for cheetah images using one blob plus the background. The overall weights are 1.0 for the cheetah blob and 0.5 for the background. For the cheetah blob, both color and texture weights are 1.0. (Only color is used for the background score.)

order to test this hypothesis, we performed 50 queries using both Blobworld and global color and texture histograms.

We selected 10 object categories: airplanes, black bears, brown bears, cheetahs, eagles, elephants, horses, polar bears, tigers, and zebras. There were between 30 and 200 examples of each category among the 10,000 images.

We compared the Blobworld results to a ranking algorithm that used the global color and texture histograms of the same 10,000 images. The color histograms used the same 218 $L^*a^*b^*$ bins as Blobworld, along with the same quadratic distance as described in Section 4. For texture

histograms, we discretized the two texture features (mean contrast and anisotropy) into 21 bins each, spaced uniformly on the interval from 0 to 1. When using global histograms, we found that color carried most of the useful information; varying the texture weight made very little difference to the query results.

For each category we tested queries using two blobs, one blob plus background, and global histograms. In each case, we performed a few test queries to select the weights for color and texture and for each blob. We then queried using five new images. (We used the same weights for each image in a category.) In Fig. 11, we plot the average precision (the
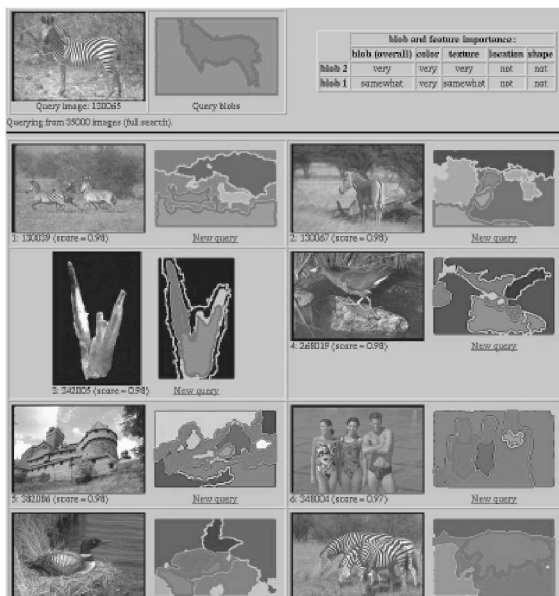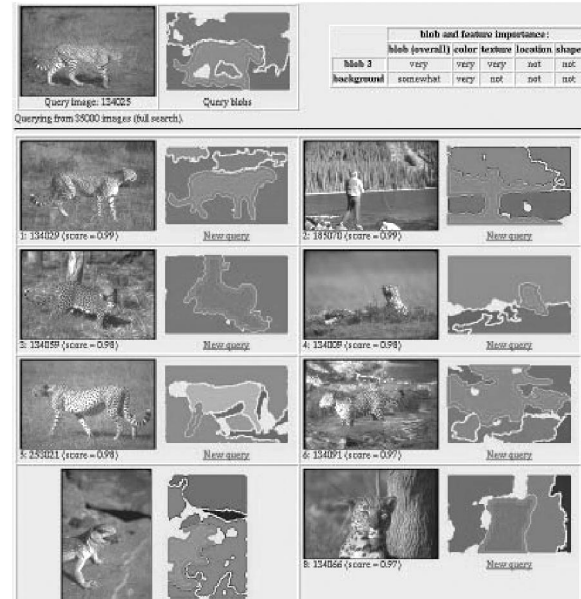

Fig. 8. Blobworld query for zebra images using two blobs. The overall weights are 1.0 for the zebra blob and 0.5 for the grass blob. For the zebra blob, both color and texture weights are 1.0. For the grass blob, the color weight is 1.0 and the texture weight is 0.5.
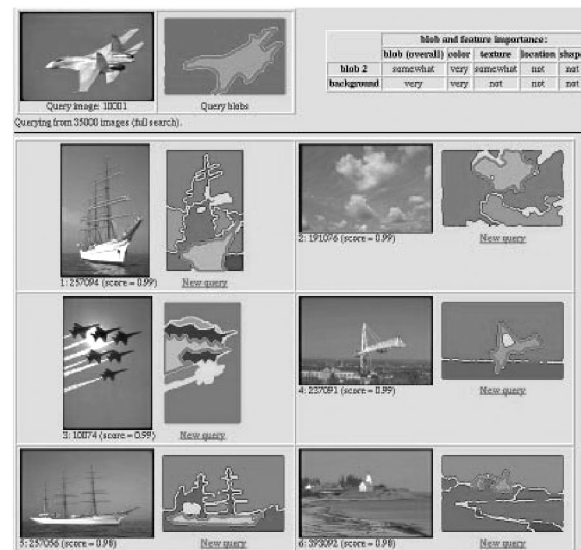

Fig. 10. Blobworld query for airplane images using one blob plus the background. The overall weights are 0.5 for the airplane blob and 1.0 for the background. For the airplane blob, the color weight is 1.0 and the texture weight is 0.5. (Only color is used for the background score.)
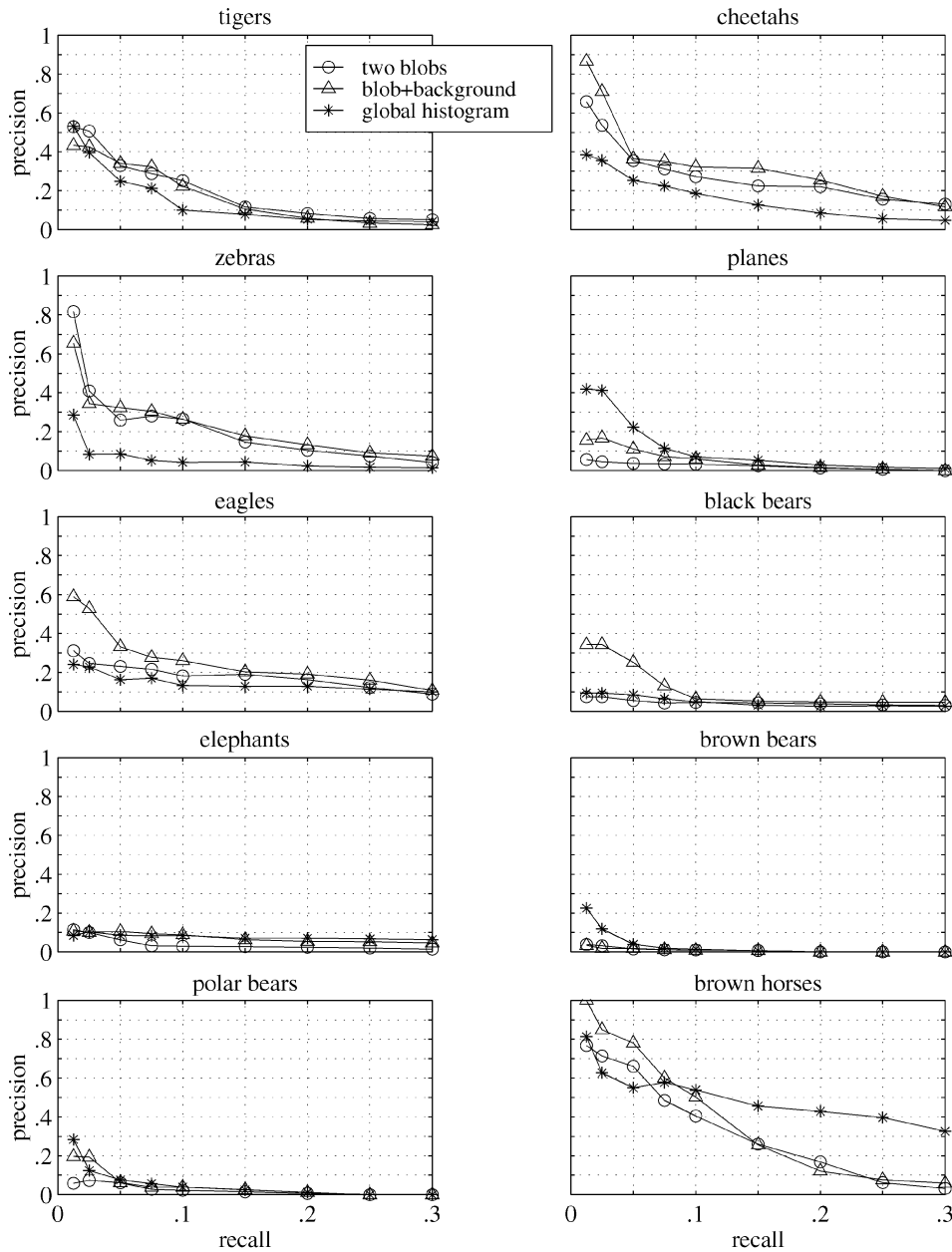
Fig. 11. Precision versus recall of queries using two blobs, one blob plus background, and global histograms for 10 categories. Blobworld performs better on queries for *distinctive objects* such as tigers, cheetahs, and zebras, while global histograms perform better on queries for *distinctive scenes* such as airplane scenes. Queries using one blob plus the background do better than both two-blob queries and global histogram queries for eagles and black bears, where *both the object and the scene* are distinctive. (Chance would yield precision ranging from 0.003 for zebras to 0.02 for airplanes.)

fraction of retrieved images which are relevant) versus recall (the fraction of relevant images which are retrieved) for each of the 10 categories. (The 10,000 database images were marked as "relevant" or "not relevant" to each category—i.e., containing the object or not—by a human observer.)

The results indicate that the categories fall into four groups:

Distinctive objects. The color and texture of cheetahs, tigers, and zebras are quite distinctive, and Blobworld performance (using either two blobs or blob+background) is better than global histogram performance.

Distinctive scenes. For most of the airplane images, the entire scene is distinctive (a small gray object and large

amounts of blue), but the airplane region itself has quite a common (and variable) color and texture. Global histograms do better than Blobworld in the airplane category. Global histograms are also slightly better than Blobworld on brown bear images.

Distinctive objects *and* scenes. Bald eagle images are characterized by both a somewhat distinctive object and a distinctive background. Blob+background queries perform much better in this category than both two-blob and global histogram queries. (Contrast this to the airplane case; bald eagles are much more consistent in color than are airplanes.) Blob+background queries for black bears also perform better than both two-blob and global histogram queries.
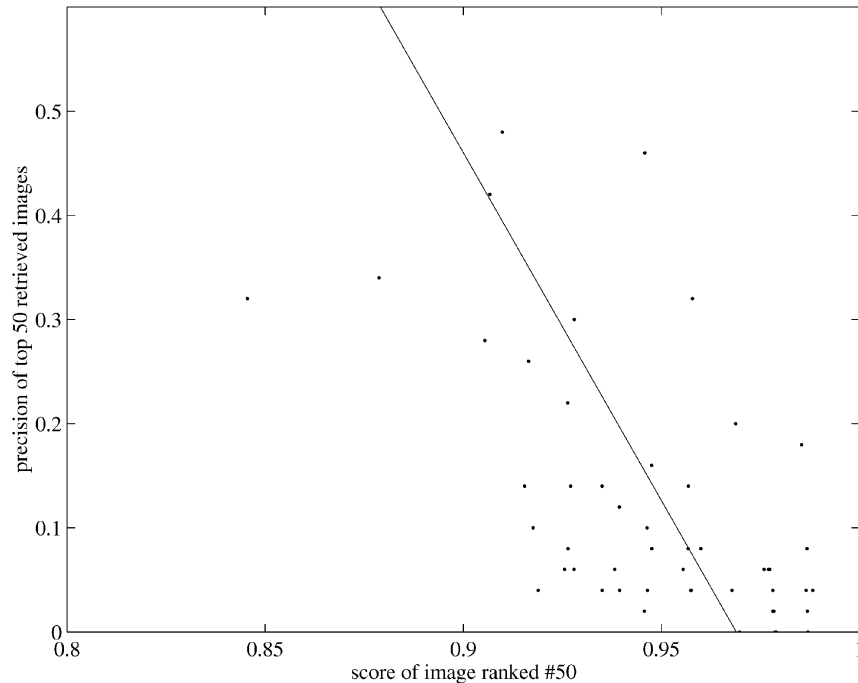
Fig. 12. Scatterplot of precision of top 50 retrieved images versus score of image ranked in 50th place. Data are from five queries in each of ten categories, using two blobs. The linear correlation coefficient is -0.59, indicating a strong negative correlation. The best-fitting line is shown.

Other. The two methods perform comparably on the other categories: elephants, polar bears, and brown horses. Blobs with the same color and texture as these objects are common in the database, but the overall scene (a general outdoor scene) is also common, so neither Blobworld nor global histograms has an advantage, given that we used only color and texture. However, histograms can be taken no further, while Blobworld has much room left for improvement. For example, the shapes of elephants, bears, and horses (as well as airplanes and other objects) are distinctive.

These results support our hypothesis that Blobworld yields good results when querying for distinctive objects.

In addition to better query performance, Blobworld has other advantages relative to global histograms. For example, we have developed a sketch interface that allows a user to construct a query by drawing several blobs and specifying their characteristics; building a query in this way is much more intuitive than specifying a global color histogram from scratch. In addition, Blobworld has the potential to incorporate shape information in the region description, while global histograms (as well as methods incorporating simple spatial information in the color histogram) do not encode the region information necessary to perform shape queries.

5.4 Why Are Some Queries Harder than Others?

In addition to the variation in relative performance between Blobworld and global histograms in several categories, there is also a variation in absolute performance among the categories. Queries for tigers, cheetahs, zebras, bald eagles, and brown horses are "easy" in that they yield high precision for at least one of the query types. Queries for elephants, black bears, and polar bears, in contrast, are "hard"—all query types yield low precision for these objects. Intuitively, the latter categories are difficult because the objects and scenes

are not distinctive; there are thousands of images in the database with gray, brown, or white blobs set in an outdoor scene. An image in one of these categories will have a large number of images from other categories nearby in the description space, which will result in low query precision.

This intuition can be quantified by examining the score of images returned near the top but not at the top of the ranked list. Most of these images are false positives; their scores indicate how close they are to the query image and thus suggest how many false positives are located near the query image. For each of the two-blob queries, we examined the score of the image ranked 50th (as calculated by the query algorithm) and the precision of the top 50 images (based on ground truth). We find that queries with high scores and, thus, many nearby neighbors, generally have low precision. Fig. 12 shows the expected strong negative correlation (linear correlation coefficient = -0.59) between the score of the image ranked 50th and the precision of the top 50 images. These results support the intuition that "hard" queries are characterized by having many nearby neighbors in the description space.

6 DISCUSSION

The basic goal in content-based image retrieval is to bridge the gap from the low-level image properties ("stuff") we can directly access to the objects ("things") users generally want to find in image databases; we see image retrieval ultimately as an object recognition problem. We propose a general approach to this problem, to proceed in three steps:

1. **Group** pixels into regions which are coherent in low-level properties and which generally correspond to objects or parts of objects.
2. **Describe** these regions in ways that are meaningful to the user.

3. **Access** these region descriptions, either automatically or with user intervention, to retrieve desired images.

In the current implementation, we group pixels into regions by modeling the joint distribution of color, texture, and position features with a mixture of Gaussians using Expectation-Maximization and the Minimum Description Length principle. After grouping, we describe the regions using simple color and texture properties. Finally, we access these descriptions in a querying framework to retrieve images.

This approach is modular; the three stages are separate and somewhat independent. In the future, we might replace or improve the three modules:

1. Since any segmentation algorithm will sometimes oversegment objects, we might include a grouping algorithm in the system to build objects out of object parts. For example, a zebra may be oversegmented into trunk, legs, and head; we would like to recognize that the stripes are similar and group the parts into one region. One such algorithm would use Gestalt factors [46] such as proximity, similarity, and symmetry to join segmented regions into groups that are likely to correspond to complete objects. We might also use simple object models to hypothesize groupings.

2. Our current features clearly do not encode all the important information about the blob: a zebra is fundamentally different from a striped awning, and shape is *the* defining feature that differentiates the two.

3. We plan to explore automatic classification of images into categories based on the region descriptions. We might also use information about the spatial relationships among blobs for both querying and classification. (One possibility is the body plan approach of Forsyth and Fleck [12].)

Added complexity in the latter stages depends to some extent on improvements in the first stage; richer shape and configuration information will be most useful if segmentation quality improves.

Finally, some remarks on the broader implications of this line of research to computer vision in general. It is a common belief in the computer vision community that general-purpose image segmentation is a hopeless goal. This has led to two distinct responses in the content-based querying community and the object recognition community. Researchers interested in content-based querying have focused on descriptors such as color histograms which can operate in the absence of segmentation; researchers in object recognition have either set up situations where segmentation is not an issue (e.g., putting the object on a black background) or searched for a well-defined, specific, geometric model or photometric template in the image.

Our belief is that segmentation, while imperfect, is an essential first step, as the combinatorics of searching for all possible instances of a class is intractable. A combined architecture for segmentation and recognition is needed, analogous to inference using Hidden Markov Models in speech recognition. We cannot claim that our framework provides an ultimate solution to this central problem in computer vision. What the task does offer are a number of desirable attributes as a testbed: a richness of imagery that excludes "cheat" solutions, something like the right kind of modules, and a useful and important role for learning to capture intraclass variations. The results in this paper provide a baseline performance on a widely available image data set that could be used for testing other proposed approaches. Individual modules can also be tested for their impact on overall performance.

REFERENCES

[1] Special issue on digital libraries, *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 8, Aug. 1996.

[2] J. Ashley, R. Barber, M. Flickner, J. Hafner, D. Lee, W. Niblack, and D. Petkovic, "Automatic and Semiautomatic Methods for Image Annotation and Retrieval in QBIC," *SPIE Proc. Storage and Retrieval for Image and Video Databases,* pp. 24-35, 1995.

[3] S. Ayer, H. Sawhney, "Layered Representation of Motion Video Using Robust Maximum-Likelihood Estimation of Mixture Models and MDL Encoding," *Proc. Int'l Conf. Computer Vision,* pp. 777-784, 1995.

[4] S. Belongie, C. Carson, H. Greenspan, and J. Malik, "Color- and Texture-Based Image Segmentation Using EM and Its Application to Content-Based Image Retrieval," *Proc. Int'l Conf. Computer Vision,* pp. 675-682, 1998.

[5] J. Bigün, G. Granlund, and J. Wiklund, "Multidimensional Orientation Estimation with Applications to Texture Analysis and Optical Flow," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 8, pp. 775-790, Aug. 1991.

[6] C. Carson, S. Belongie, H. Greenspan, and J. Malik, "Region-Based Image Querying," *Proc. IEEE Workshop Content-Based Access of Image and Video Libraries,* 1997.

[7] C. Carson, M. Thomas, S. Belongie, J.M. Hellerstein, and J. Malik, "Blobworld: A System for Region-Based Image Indexing and Retrieval," *Proc. Int'l Conf. Visual Information System,* pp. 509-516, 1999.

[8] A. Dempster, N. Laird, and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *J. Royal Statistical Soc., Ser. B,* vol. 39, no. 1, pp. 1-38, 1977.

[9] P. Enser, "Query Analysis in a Visual Information Retrieval Context," *J. Document and Text Management,* vol. 1, no. 1, pp. 25-52, 1993.

[10] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by Image and Video Content: The QBIC System," *IEEE Computer,* vol. 28, no. 9, pp. 23-32, Sept. 1995.

[11] W. Förstner, "A Framework for Low Level Feature Extraction," *Proc. European Conf. Computer Vision,* pp. 383-394, 1994.

[12] D. Forsyth and M. Fleck, "Body Plans," *Proc. IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition,* pp. 678-683, 1997.

[13] D. Forsyth, J. Malik, and R. Wilensky, "Searching for Digital Pictures," *Scientific Am.,* vol. 276, no. 6, pp. 72-77, June 1997.

[14] W.T. Freeman and E.H. Adelson, "The Design and Use of Steerable Filters," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 9, pp. 891-906, Sept. 1991.

[15] J. Gårding and T. Lindeberg, "Direct Computation of Shape Cues Using Scale-Adapted Spatial Derivative Operators," *Int'l J. Computer Vision,* vol. 17, no. 2, pp. 163-191, Feb. 1996.

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

[17] A. Gupta and R. Jain, "Visual information retrieval," *Comm. Assoc. Computer Machine,* vol. 40, no. 5, pp. 70-79, May 1997.

[18] J. Hafner, H. Sawhney, W. Equitz, M. Flickner, and W. Niblack, "Efficient Color Histogram Indexing for Quadratic Form Distance Functions," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 17, no. 7, pp. 729-736, July 1995.

[19] D. Harman, "Relevance Feedback and Other Query Modification Techniques," *Information Retrieval: Data Structures & Algorithms,* W.B. Frakes and R. Baeza-Yates, eds. Prentice Hall 1992.

[20] J. Huang, S.R. Kumar, M. Mitra, W.-J. Zhu, and R. Zabih, "Image Indexing Using Color Correlograms," *Proc. IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition,* pp. 762-768, 1997.

[21] C. Jacobs, A. Finkelstein, and D. Salesin, "Fast Multiresolution Image Querying," *Proc. SIGGRAPH,* pp. 277-286, 1995.

[22] A.K. Jain and F. Farrokhnia, "Unsupervised Texture Segmentation Using Gabor Filters," *Pattern Recognition,* vol. 24, no. 12, pp. 1167-1186, 1991.

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

[24] P. Kelly, M. Cannon, and D. Hush, "Query by Image Example: The CANDID Approach," *SPIE Proc. Storage and Retrieval for Image and Video Databases,* pp. 238-248, 1995.

[25] T. Leung and J. Malik, "Detecting, Localizing and Grouping Repeated Scene Elements from an Image," *Proc. European Conf. Computer Vision,* pp. 546-555, 1996.

[26] P. Lipson, E. Grimson, and P. Sinha, "Configuration Based Scene Classification and Image Indexing," *Proc. IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition,* pp. 1007-1013, 1997.

[27] W. Ma and B. Manjunath, "NeTra: A Toolbox for Navigating Large Image Databases," *Proc. IEEE Int'l Conf. Image Processing,* pp. 568-571, 1997.

[28] J. Malik and P. Perona, "Preattentive Texture Discrimination with Early Vision Mechanisms," *J. Optical Soc. Am. A,* vol. 7, no. 5, pp. 923-932, 1990.

[29] T.P. Minka and R.W. Picard, "Interactive Learning Using a 'Society of Models,'" *Pattern Recognition,* vol. 30, no. 4, 1997.

[30] V. Ogle and M. Stonebraker, "Chabot: Retrieval from a Relational Database of Images," *IEEE Computer,* vol. 28, no. 9, pp. 40-48, Sept. 1995.

[31] D. Panjwani and G. Healey, "Markov Random Field Models for Unsupervised Segmentation of Textured Color Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 17, no. 10, pp. 939-954, Oct 1995.

[32] A. Pentland, R. Picard, and S. Sclaroff, "Photobook: Content-Based Manipulation of Image Databases," *Int'l J. Computer Vision,* vol. 18, no. 3, pp. 233-254, 1996.

[33] J. Ponce, A. Zisserman, and M. Hebert, "Object Representation in Computer Vision—II," *Lecture Notes in Computer Science,* No. 1144, 1996.

[34] J. Puzicha and J.M. Buhmann, "Multiscale Annealing for Real-Time Unsupervised Texture Segmentation," *Proc. Int'l Conf. Computer Vision,* pp. 267-273, 1998.

[35] J. Rissanen, "Modeling by Shortest Data Description," *Automatica* vol. 14, pp. 465-471, 1978.

[36] J. Rissanen, *Stochastic Complexity in Statistical Inquiry.* World Scientific, 1989.

[37] C. Schmid and R. Mohr, "Combining Grey Value Invariants with Local Constraints for Object Recognition," *Proc. IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition,* pp. 872-877, 1996.

[38] G. Schwarz, "Estimating the Dimension of a Model," *Annals of Statistics,* vol. 6, pp. 461-464, 1978.

[39] J.R. Smith and S.-F. Chang, "Single Color Extraction and Image Query," *Proc. IEEE Int'l Conf. Image Processing,* pp. 528-531, 1995.

[40] J.R. Smith and S.-F. Chang, "Tools and Techniques for Color Image Retrieval," *SPIE Proc. Storage and Retrieval for Image and Video Databases,* vol. 2670, pp. 426-437, 1996.

[41] M. Stricker and A. Dimai, "Spectral Covariance and Fuzzy Regions for Image Indexing," *Machine Vision and Applications,* vol. 10, no. 2, pp. 66-73, 1997.

[42] M. Stricker and M. Swain, "The Capacity and the Sensitivity of Color Histogram Indexing," Technical Report 94-05, Univ. of Chicago, Mar. 1994.

[43] M. Swain and D. Ballard, "Color Indexing," *Int'l J. Computer Vision,* vol. 7, no. 1, pp. 11-32, 1991.

[44] Y. Weiss and E. Adelson, "A Unified Mixture Framework for Motion Segmentation: Incorporating Spatial Coherence and Estimating the Number of Models," *Proc. IEEE Computer Soc. Conf. Computer Vision and Pattern Recognition,* pp. 321-326, 1996.

[45] W. Wells, R. Kikinis, W. Grimson, and F. Jolesz, "Adaptive Segmentation of MRI Data," *Int'l Conf. Computer Vision, Virtual Reality, and Robotics in Medicine,* pp. 59-69, 1995.

[46] M. Wertheimer, "Laws of Organization in Perceptual Forms," *A Source Book of Gestalt Psychology,* W.D. Ellis, ed. Harcourt Brace, 1938.

[47] G. Wyszecki and W. Stiles, *Color Science: Concepts and Methods, Quantitative Data and Formulae,* second ed. Wiley, 1982.

Chad Carson received the BS degree in electrical engineering and the BA degree in history (cum laude) from Rice University in 1994 and the MS degree in electrical engineering from the University of California at Berkeley in 1997. He received the PhD degree in electrical engineering and computer sciences from the University of California at Berkeley in 1999. He was awarded the Rice Engineering Senior Scholar Award and the US National Science Foundation Graduate Research Fellowship. While at Rice, he served as the editor of the campus newspaper. His research interests include content-based image retrieval and digital libraries. He is a member of the IEEE.

Serge Belongie received the BS degree (with honors) in electrical engineering from the California Institute of Technology in 1995 and the MS and PhD degrees in electrical engineering and computer sciences from the University of California at Berkeley in 1997 and 2000, respectively. While at Berkeley, his research was supported by a US National Science Foundation Graduate Research Fellowship and the Chancellor's Opportunity Predoctoral Fellowship. He is also a cofounder of Digital Persona, Inc. and the principal architect of the Digital Persona fingerprint recognition algorithm. He is currently an assistant professor in the Computer Science and Engineering Department at University of California at San Diego. His research interests include computer vision, pattern recognition, and digital signal processing. He is a member of the IEEE.

Hayit Greenspan received the BS and MS degrees in electrical engineering from the Technion Israel Institute of Technology, Haifa, in 1986 and 1989, respectively, and the PhD degree in electrical engineering from the California Institute of Technology, Pasadena, in 1994. She was a postdoctoral candidate with the Computer Science Division, University of California, Berkeley, from 1995 to 1997. In 1997, she joined the BioMedical Engineering Department, Faculty of Engineering, Tel-Aviv University, Tel-Aviv, Israel. Her research interests include content-based image search and retrieval, medical image processing, texture recognition, image enhancement, and pattern recognition. She is a member of the IEEE.

Jitendra Malik received the BTech degree in electrical engineering from Indian Institute of Technology, Kanpur in 1980 and the PhD degree in computer science from Stanford University in 1986. In January 1986, he joined the faculty of the Computer Science Division, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, where he is currently a professor. From 1995 to 1998, he also served as vice-chair for graduate matters. He is a member of the cognitive science and vision science groups at the University of California at Berkeley. His research interests are in computer vision and computational modeling of human vision. His work spans a range of topics in vision including image segmentation and grouping, texture, stereopsis, object recognition, image-based modeling and rendering, content-based image querying, and intelligent vehicle highway systems. He has authored or coauthored more than a hundred research papers on these topics. He received the gold medal for the best graduating student in electrical engineering from IIT Kanpur in 1980, a Presidential Young Investigator Award in 1989, and the Rosenbaum fellowship for the Computer Vision Programme at the Newton Institute of Mathematical Sciences, University of Cambridge in 1993. He received the Diane S. McEntyre Award for Excellence in Teaching from the Computer Science Division, University of California at Berkeley, in 2000. He is an editor-in-chief of the *International Journal of Computer Vision*. He is a member of the IEEE.