Blobworld: A System for Region-Based Image Indexing and Retrieval

Chad Carson, Megan Thomas, Serge Belongie,
Joseph M. Hellerstein, and Jitendra Malik

EECS Department
University of California, Berkeley, CA 94720, USA
{carson,mct,sjb,jmh,malik}@eecs.berkeley.edu

Abstract. Blobworld is a system for image retrieval based on finding coherent image regions which roughly correspond to objects. Each image is automatically segmented into regions ("blobs") with associated color and texture descriptors. Querying is based on the attributes of one or two regions of interest, rather than a description of the entire image. In order to make large-scale retrieval feasible, we index the blob descriptions using a tree. Because indexing in the high-dimensional feature space is computationally prohibitive, we use a lower-rank approximation to the high-dimensional distance. Experiments show encouraging results for both querying and indexing.

1 Introduction

From a user's point of view, the performance of an information retrieval system can be measured by the quality and speed with which it answers the user's information need. Several factors contribute to overall performance:

– the time required to run each individual query,
– the quality (precision/recall) of each individual query's results, and
– the understandability of results and ease of refining the query.

These factors should be considered together when designing a system. In addition, image database users generally want to find images based on the *objects* they contain, not just low-level features such as color and texture [5]; image retrieval systems should be evaluated based on their performance at this task.

Current image retrieval systems tend to perform queries quickly but do not succeed in the other two areas. A key reason for the poor quality of query results is that the systems do not look for meaningful image regions corresponding to objects. Additionally, the results are often difficult to understand because the system acts like a black box. Consequently, the process of refining the query may be frustrating. When individual query results are unpredictable, it is difficult to produce a stream of queries that satisfies the user's need.

In earlier work we described "Blobworld," a new framework for image retrieval based on segmenting each image into regions ("blobs") which generally correspond to objects or parts of objects [2]. The segmentation algorithm is fully automatic; there is no parameter tuning or hand pruning of regions. In this paper we present a complete online system for retrieval in a collection of 10,000 Corel images using this approach. The query system is available at `http://elib.cs.berkeley.edu/photos/blobworld`.

We present results indicating that querying for distinctive objects such as tigers, zebras, and cheetahs using Blobworld produces higher precision than does querying using color and texture histograms of the entire image. In addition, Blobworld false positives are easier to understand because the matching regions are highlighted. This presentation of results means that interpreting and refining the query is more productive with the Blobworld system than with systems that use low-level features from the entire image.

The speed of individual queries is also an important factor, so we describe an approach to indexing Blobworld features. We project each color feature vector down to a lower dimensional vector and index the resulting vector. We find that queries that use the index to retrieve several hundred images and then rank those images using the true distance achieve results whose quality closely matches the results of queries that scan the entire database.

We begin this paper by briefly reviewing the current state of image retrieval. In Section 2 we outline the segmentation algorithm, region descriptors, and querying system. In Section 3 we discuss indexing. In Section 4 we present experiments which examine the performance of querying and indexing in Blobworld. We conclude with a brief discussion. For more details than can be presented here due to space constraints, see the complete version of this paper [3] at `http://www.cs.berkeley.edu/~carson/papers/visual99.html`.

1.1 Related Work

Image retrieval systems based primarily on low-level image features include IBM's Query by Image Content (QBIC) [6], Photobook [15], Virage [7], VisualSEEk [18], and Chabot [14]. Lipson et al. [12] retrieve images based on spatial and photometric relationships within and across simple image regions derived from low-resolution images. Jacobs et al. [11] use multiresolution wavelet decompositions to perform queries based on iconic matching. Ma and Manjunath [13] perform retrieval based on segmented image regions. Their segmentation requires some parameter tuning and hand pruning of regions.

Much research has gone into dimensionality reduction [8] and new index trees [17,19] to cope with the high dimensionality of indices built over color histograms. Work to date has focused on indexing the entire image or user-defined sub-regions, not on indexing automatically created image regions. Our indexing methods are based on those used in QBIC [8].

2 Blobworld

The Blobworld representation is related to the notion of photographic or artistic scene composition. Blobworld is distinct from color-layout matching as in QBIC [6] in that it is designed to find objects or parts of objects; each image is treated as an ensemble of a few "blobs" representing image regions which are roughly homogeneous with respect to color and texture. Each blob is described by its color distribution and mean texture descriptors. Details of the segmentation algorithm may be found in [2].

2.1 Grouping Pixels into Regions

Each pixel is assigned a vector consisting of color, texture, and position features. The three color features are the coordinates in the L*a*b* color space; we smooth these features in the image to avoid oversegmentation arising from local color variations due to texture. The three texture features are contrast, anisotropy, and polarity, extracted at an automatically selected scale. The position features are simply the (x, y) position of the pixel; including the position generally decreases oversegmentation and leads to smoother regions.

We model the distribution of pixels in this 8-D space using mixtures of two to five Gaussians. We use the Expectation-Maximization algorithm [4] to fit the mixture of Gaussians model to the data. To choose the number of Gaussians that best suits the natural number of groups present in the image, we apply the Minimum Description Length (MDL) principle [16]. Once a model is selected, we perform spatial grouping of connected pixels belonging to the same color/texture cluster. Figure 1 shows the segmentation of two sample images.

2.2 Describing the Regions

We store the color histogram over the pixels in each 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; only 218 bins fall in the gamut corresponding to $0 \leq \{R, G, B\} \leq 1$.

To match the color of two regions, we use the quadratic distance between their histograms $\mathbf{x}$ and $\mathbf{y}$, $d^2_{hist}(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^{\mathbf{T}} \mathbf{A} (\mathbf{x} - \mathbf{y})$ [8]. $\mathbf{A} = [a_{ij}]$ 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; adjacent 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 also store the mean texture contrast and anisotropy.

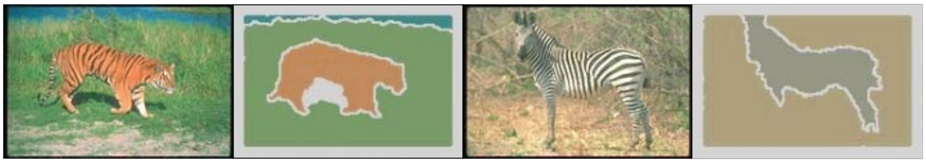

Fig. 1. Sample Blobworld representations. Each blob is shown as an area of constant color.

2.3 Querying in Blobworld

The user composes a query by submitting an image in order to see its Blobworld representation, selecting the relevant blobs to match, and specifying the relative importance of the blob features. We return the best matching images, 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 weights or may specify new blobs to match and then issue a new query. (The details of the ranking algorithm may be found in [3].)

3 Indexing

Indices allow the computer to find images relevant to a query without looking at every image in the database. We used R*-trees [1], index structures for data representable as points in an N-dimensional space. R*-trees are not the state of the art for nearest-neighbor search [10] in multiple dimensions; using a newer tree [17,19] would likely speed up the index by a constant factor. However, our basic observations are independent of this index tuning: (i) indexing over blobs can decrease query time without significantly reducing quality, and (ii) indices over blobs can perform better than whole-image indices. We used the GiST framework [9] to experiment with the indices.

Shorter paths from root to leaf in an index tree lead to fewer disk accesses to reach the leaves, and thus faster data retrieval. Node fanout, the number of data entries that can fit in a node (disk page), dictates tree height. High dimensional data requires large data entries and thus low fanout and slow index retrieval. At sufficiently high dimensions fanout becomes so low that query speed using the index is worse than simply scanning the entire database. To avoid this, we need a low-dimensional approximation to the full color feature vectors.

Computing the full distance $d(\mathbf{x},\mathbf{y}) = \left[(\mathbf{x}-\mathbf{y})^{\mathbf{T}}\mathbf{A}(\mathbf{x}-\mathbf{y})\right]^{1/2}$ would require storing the entire 218-dimensional histogram and performing the full matrix-vector multiplication. To reduce the storage and computation in the index, we use Singular Value Decomposition to find $\mathbf{A}_k$, the best rank-k approximation to the weight matrix $\mathbf{A}$. We then project $\mathbf{x}$ and $\mathbf{y}$ into the subspace spanned by the rows of $\mathbf{A}_k$, yielding $\mathbf{x}_k$ and $\mathbf{y}_k$. The Euclidean distance $\left[(\mathbf{x}_k-\mathbf{y}_k)^{\mathbf{T}}(\mathbf{x}_k-\mathbf{y}_k)\right]^{1/2}$ is a lower bound on the full distance. Since the singular values $\sigma_{k+1},\ldots,\sigma_{218}$ are small for our $\mathbf{A}$, this bound is tight. We can thus index the low-dimensional $\mathbf{x}_k$'s and use the Euclidean distance in the index without introducing too much error.

The index aims to match the quality of full queries. We would like the index to retrieve exactly the images that the full query ranks as the best matches. However, there is a quality/time tradeoff: as the index returns more images, the final query results will get better, but the query will take longer.

4 Experiments

In order to understand the performance of the Blobworld system, we explored several questions:

- What is the precision of Blobworld queries compared to queries using global color and texture histograms? More specifically, for which queries does Blobworld do better, and for which do global histograms do better?
- How well do the indexing results approximate the results from a full scan of the collection? (According to this measure, false positives returned by the full scan should also be returned by the index.) Here we must consider the dimensionality of the indexed data as well as how indices over blobs compare to indices over whole image histograms.
- What do we lose by using an index instead of a full scan? That is, how does the precision of the indexed query compare to the full-scan precision?

We explore each of these questions in turn in the next three sections.

4.1 Comparison of Blobworld and Global Histograms

We expected that Blobworld querying would perform well in cases where a distinctive object is central to the query. In order to test this hypothesis, we performed 50 queries using both Blobworld and global color and texture histograms.

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 bins as Blobworld, along with the same quadratic distance. For texture histograms, we discretized the two texture features into 21 bins each. In this global image histogram case, we found that color carried most of the useful information; varying the texture weight made little difference to the query results.

For each of ten categories we performed five queries using one blob, two blobs, and global histograms. (We used the same feature weights for all queries in a category.) In Figure 2 we plot the average precision for queries in the tiger, cheetah, zebra, and airplane categories; the differences between Blobworld and global histograms show up most clearly in these categories.

The results indicate that the categories fall into three groups:

distinctive objects: The color and texture of cheetahs, tigers, and zebras are quite distinctive, and the Blobworld query precision was higher than the global histogram query precision.

distinctive scenes: For most of the airplane images the entire scene is distinctive, but the airplane region itself has quite a common color and texture. Global histograms did better than Blobworld in this category. (We have added an option to allow the user to use the entire background in place of the second blob. Using the background improves the Blobworld performance on these distinctive-scene queries, since it avoids matching, for example, the small regions of sky found in thousands of images in the database.)

other: The two methods perform comparably on the other six categories: bald eagles, black bears, brown bears, elephants, horses, and polar bears. 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 animals and airplanes are quite distinctive, and by segmenting the image we have some prospect of describing object shape in a meaningful way. (Using the background improves the Blobworld performance in some of these categories as well.)

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

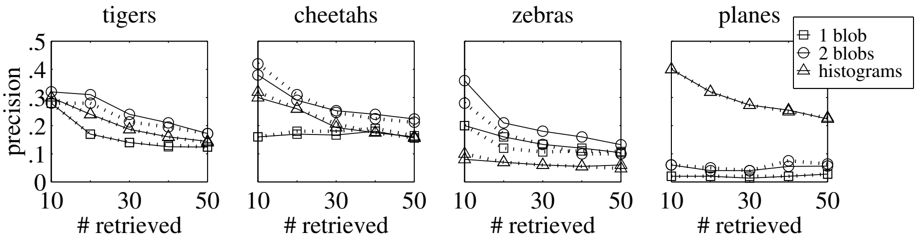

Fig. 2. Average precision (fraction of retrieved images which are correct) vs. number of images retrieved for several query types. Solid lines represent full queries; dashes represent indexed queries. The index tracks the corresponding full query quite closely, except for the zebra case, where the indexed Blobworld precision is lower than the full Blobworld precision because we only index color, not texture.

4.2 Comparison of Indexed to Full Queries over Multiple Index Dimensionalities

Index speed improves as the number of dimensions in the index decreases; therefore, we want to find the minimum number of dimensions that the index may use. Query speed improves as the number of images the index must fetch and the full Blobworld algorithms rank decreases; therefore, we want to find the minimum number of images that the index may return. However, we want to ensure that a query using the index produces nearly the same results as a full Blobworld query. We are also interested in how blob index performance compares to global histogram index performance.

For simplicity, we compared the indices based on color feature vectors alone. We measured the recall of the indices using nearest-neighbor search to retrieve and rank images against the top 40 images retrieved by a full Blobworld query or global histogram query over all the images. Figure 3 shows that in the low-dimensional case recall for the blob indices is higher than for the global histogram indices; blob indices approximate the results of full Blobworld queries better than global histogram indices approximate the results of global histogram queries. We believe this occurs because the blob color histograms, which are derived from relatively uniformly colored regions, cluster better than the global histograms.

Retrieving just a few hundred blobs from the five-dimensional blob index gives us most of the images the full Blobworld query ranked highest. Therefore, for the remaining experiments we use the five-dimensional indices.

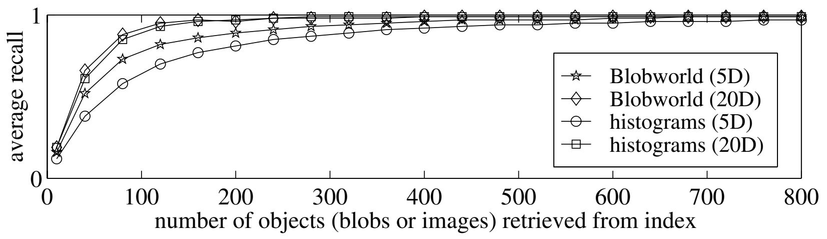

Fig. 3. Recall of (1) blob index compared to the top 40 images from the full Blobworld query and (2) global histogram index compared to the top 40 images from the full whole image query. The plots are the average of 200 queries over a database of 10,000 images, or about 61,000 blobs. Results using five dimensions and twenty dimensions are shown.

4.3 Precision of Indexed and Full Queries

We also wanted to test the behavior of the indexing schemes in terms of precision measured against ground truth. In essence, we wanted to see how indexing affects the quality of Blobworld query results.

We performed the same queries as in Section 4.1, using the index to reduce the number of "true" comparisons required. We passed the query to an index using the five-dimensional projection and retrieved the nearest 400 database objects (400 blobs for Blobworld, 400 images for global histograms). When indexing two-blob queries, we retrieved the nearest 400 matches to each of the two blobs and returned the union of the two result sets.

Figure 2 indicates that the precision of the indexed results (reordered using the "true" matching algorithm) closely mirrors the precision of the full query results. As previously stated, this is the quality goal for the index. Simple timing tests indicate that indexed Blobworld queries run in a third to half of the time of the full query. As the number of images in the collection increases, this speed advantage will become even greater.

5 Conclusions

The effectiveness of an image retrieval system is determined by three factors: the time to perform an individual query, the quality of the query results, and the ease of understanding the query results and refining the query. We have shown that Blobworld queries for distinctive objects provide high precision and understandable results because Blobworld is based on finding coherent image regions. We have also shown that Blobworld queries can be indexed to provide fast retrieval while maintaining precision.

Acknowledgments

This work was supported by an NSF Digital Library Grant (IRI 94-11334), NSF graduate fellowships for Serge Belongie and Chad Carson, and fellowship stipend support for Megan Thomas from the National Physical Science Consortium and Lawrence Livermore National Laboratory.

References

1. Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B.: The R*-tree: An efficient and robust access method for points and rectangles. In Proc. ACM-SIGMOD Int'l Conf. on Management of Data (1990) 322–331 512
2. Belongie, S., Carson, C., Greenspan, H., Malik, J.: Color- and texture-based image segmentation using EM and its application to content-based image retrieval. In Proc. Int. Conf. Comp. Vis. (1998) 509, 511
3. Carson, C., Thomas, M., Belongie, S., Hellerstein, J., Malik, J.: Blobworld: A system for region-based image indexing and retrieval. Technical Report UCB/CSD-99-1041. 510, 512
4. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. J. Royal Statistical Soc., Ser. B, **39** (1977) 1–38 511
5. Enser, P.: Query analysis in a visual information retrieval context. J. Doc. and Text Management, **1** (1993) 25–52 509
6. Flickner, M., Sawhney, H., Niblack, W., Ashley, J., et al: Query by image and video content: The QBIC system. IEEE Computer, **28** (Sept. 1995) 23–32 510
7. Gupta, A., Jain, R.: Visual information retrieval. Comm. Assoc. Comp. Mach., **40** (May 1997) 70–79 510
8. Hafner, J., Sawhney, H., Equitz, W., Flickner, M., Niblack, W.: Efficient color histogram indexing for quadratic form distance functions. IEEE Trans. Pattern Analysis and Machine Intelligence, **17** (July 1995) 729–736 510, 511
9. Hellerstein, J. M., Naughton, J., Pfeffer, A.: Generalized search trees for database systems. In Proc. 21st Int. Conf. on Very Large Data Bases (1995) 562–573 512
10. Hjaltason, G., Samet, H.: Ranking in spatial databases. In Proc. 4th Int. Symposium on Large Spatial Databases (1995) 83–95 512
11. Jacobs, C., Finkelstein, A., Salesin, D.: Fast multiresolution image querying. In Proc. SIGGRAPH (1995) 510
12. Lipson, P., Grimson, E., Sinha, P.: Configuration based scene classification and image indexing. In Proc. IEEE Comp. Soc. Conf. Comp. Vis. and Patt. Rec., (1997) 1007–1013 510
13. Ma, W., Manjunath, B.: NeTra: A toolbox for navigating large image databases. In Proc. IEEE Int. Conf. on Image Proc., (1996) 568–571 510
14. Ogle, V., Stonebraker, M.: Chabot: Retrieval from a relational database of images. IEEE Computer, **28** (Sept. 1995) 40–48 510

15. Pentland, A., Picard, R., Sclaroff, S.: Photobook: Content-based manipulation of image databases. Int. J. Comp. Vis., **18** (1996) 233–254 510
16. Rissanen, J.: Stochastic Complexity in Statistical Inquiry. World Scientific (1989) 511
17. Berchtold, D. K. S., Kriegel, H.: The x-tree: An index structure for high-dimensional data. In Proc. of the 22nd VLDB Conference (1996) 28–39 510, 512
18. Smith, J. R., Chang, S.-F.: Single color extraction and image query. In Proc. IEEE Int. Conf. on Image Processing (1995) 528–531 510
19. White, D., Jain, R.: Similarity indexing with the ss-tree. In Proc. 12th IEEE Int'l Conf. on Data Engineering (1996) 516–523 510, 512