Comparing Images Using Joint Histograms

Greg Pass
Ramin Zabih
Computer Science Department
Cornell University
Ithaca, NY 14853
{gregpass,rdz}@cs.cornell.edu
607-255-8413

Abstract

Color histograms are widely used for content-based image retrieval due to their efficiency and robustness. However, a color histogram only records an image's overall color composition, so images with very different appearances can have similar color histograms. This problem is especially critical in large image databases, where many images have the same color histogram. In this paper we propose an alternative to color histograms called a *joint histogram*, which incorporates additional information without sacrificing the robustness of color histograms. We create a joint histogram by selecting a set of local pixel features and constructing a multidimensional histogram. Each entry in a joint histogram contains the number of pixels in the image that are described by a particular combination of feature values. We describe a number of different joint histograms, and evaluate their performance for image retrieval on a database with over 210,000 images. On our benchmarks, joint histograms outperform color histograms by an order of magnitude.

Keywords:

Content-based image database indexing and retrieval

1 Introduction

Many applications require methods for comparing images based on their content. Examples include scene break detection and parsing in video [2, 6, 14, 23] and image database retrieval [1, 4, 13, 16, 18]. Keyword-tagging of images by hand is neither flexible enough a solution to satisfy the growing community of imagery users, nor fast enough a process to compete with the rate of information gathering. Instead, fully automated content-based solutions must be employed.

In this paper we focus on content-based methods for *example-based* image retrieval, in which the user presents a query image to the system, and the most similar images are retrieved. A flexible retrieval system should allow for large changes in the appearance of similar images, as shown in figures 5 and 6.

Most image retrieval systems operate in two distinct phases:

1. *Image summary.* Every image in the database is summarized as a vector, utilizing a particular method. The vectors are computed once and stored prior to retrieval.

2. *Summary comparison.* When the user presents a query, a comparison measure is used to retrieve some number of the most similar vectors.

Color histogramming is the most widely used image summary, employed in systems such as IBM's QBIC [4] and Virage's VIR Engine [1]. Color histograms are popular because they are trivial to compute, and robustly tolerate movement of objects in the image and changes in camera viewpoint. Typically color histograms are compared using the L_2 or L_1 distance.

Color histograms have proven effective for small databases, but their limitations become rapidly apparent with larger databases. Because a color histogram records only color information, images with similar color histograms can have dramatically different appearances, as shown in figure 1. The amount of red in the golfer's shirt is approximately equal to that in the flowers. In a large database, it is common for unrelated images to have similar color histograms.

Figure 1: Two images with similar color histograms

In this paper we propose an image summary called a *joint histogram*, designed for use with large databases. A joint histogram is a multidimensional histogram created from a set of local pixel features. An entry in a joint histogram counts the number of pixels in the image that are described by a particular combination of feature values. Joint histograms can be compared with the same measures as color histograms.

We begin with a review of color histograms and related image summaries. In section 3 we present joint histograms. In section 4 we describe our experimental setup and show that joint histograms can significantly outperform color histograms for a database of over 210,000 images. Finally, we discuss a number of extensions to our basic method.

2 Image summaries

An image retrieval system should allow for large changes in the appearance of similar images, such as

- rotation and translation of objects in the image,
- addition, occlusion and subtraction of objects in the image, and
- changes in camera viewpoint and magnification.

It is also important that the summary method be efficient in order to handle large imagery collections.

2.1 Color histograms

A color histogram is a vector where each entry stores the number of pixels of a given color in the image. All images are scaled to contain the same number of pixels before histogramming, and the colors of the image are mapped into a discrete colorspace containing n colors. Typically images are represented in the RGB colorspace, using a few of the most significant bits per color channel to discretize the space.

Color histograms are widely used for content-based image retrieval [1, 4, 13] because they are trivial to compute, and despite their simplicity, exhibit attractive properties. Since color histograms do not relate spatial information with the pixels of a given color, they are larely invariant to the rotation and translation of objects in the image. Additionally, color histograms are robust against occlusion and changes in camera viewpoint.

Image retrieval using color histograms has been shown to be effective for image databases containing 66 images [20] and 1440 images [4]. However, color histograms have proven less successful on databases with tens of thousands of images [15]. Because a color histogram records only color information, images with similar histograms can have dramatically different appearances, such as those in figure 1. In a large database, many unrelated images will happen to have similar color histograms.

2.2 Other image summaries

Recently, several authors have proposed improvements to color histograms that incorporate spatial information. Hsu *et al.* [9] attempts to capture the spatial arrangement of the different colors in the image. The image is partioned into rectangular regions using maximum entropy, where each is region is predominantly a single color. The similarity between two images is the degree of overlap between regions of the same color. Hsu presents results from a database with 260 images, which show that their approach can give better results than color histograms. While the authors do not report running times, it appears that Hsu's method requires substantial computation, particularly the partioning algorithm. Additonally, Hsu's algorithm is affected by changes in orientation and position. Their method could be extended to be independent of these effects, at the cost of still greater overhead.

Stricker and Dimai [19] divide the image into five partially overlapping regions and compute the first three moments of the color distributions in each image. They compute moments for each color channel in the HSV colorspace, where pixels close to the border of the image have less weight. The distance between two regions is a weighted sum of the differences in each of the three moments. The distance between two images is the sum of the distance between the center regions, plus (for each of the four side regions) the minimum distance of that region to the corresponding region in the other image, when rotated by 0, 90, 180 or 270 degrees. Because the regions overlap, their method is insensitive to small rotations and translations. They also explicitly handle a limited set of rotations. Their database contains over 11,000 images, however the performance of their algorithm is only illustrated with 3 queries.

Huang *et al.* [10] propose a method that captures the spatial correlation between colors. Their approach is called color correlograms, and is related to the correlogram technique from spatial data analysis. A color correlogram for a given pair of colors (i, j) and a distance k contains the probability that a pixel with color i will be k pixels away from a pixel of color j. To reduce the storage requirements, they concentrate on autocorrelograms, where $i = j$. Huang reports good results on a database of over 18,000 images, using an experimental setup closely related to ours.

3 Joint histograms

Most of the summary methods described in the previous section improve upon color histograms by incorporating global spatial information. Although for many classes of imagery these methods can give better results than color histograms, global constraints necessarily sacrifice some flexibility in what it means for two images to be similar. For example, Stricker and Dimai's method is disrupted when objects in the image undergo significant translation, and Hsu's method cannot easily accomodate arbitrary rotation and translation of color regions.

Our approach incorporates additional information into the summary while preserving the robustness of color histograms. We create a *joint histogram* by selecting a set of local pixel features and constructing a multidimensional histogram. Each entry in a joint histogram contains the number of pixels in the image that are described by a particular combination of feature values.

For example, consider a joint histogram that combines color information with the intensity gradient. A given pixel in an image has a color (in the discretized range $0 \ldots n_{color} - 1$) and an intensity gradient (in the discretized range $0 \ldots n_{gradient} - 1$). The joint histogram for color and intensity gradient will contain $n_{color} \cdot n_{gradient}$ entries. Each entry corresponds to a particular color and a particular intensity gradient. The value stored in this entry is the number of pixels in the image with that color and intensity gradient.

More precisely, given a set of k features, where the l'th feature has n_l possible values, we can construct a joint histogram. A joint histogram is a k-dimensional vector, such that each entry in the joint histogram contains the number of pixels in an image that are described by a k-tuple of feature values. The size of the joint histogram is therefore $n = \prod_{l=1}^{k} n_l$, the number of possible combinations of the values of each feature. Just as a color histogram

approximates the density of pixel color, a joint histogram approximates the joint density of several pixel features.

3.1 Choice of local features

The features we have used were selected empirically. They can be implemented efficiently in linear time, and lend themselves to parallel programming.

- *Color.* We use the standard RGB colorspace. Note that any improvements to color histograms (such as better colorspaces) can also be applied to joint histograms.
- *Edge density.* We define the edge density at pixel (j, k) to be the ratio of edges to pixels in a small neighborhood surrounding the pixel. The edge representation of the image is computed with a standard method [12].
- *Texturedness.* We define the texturedness at pixel (j, k) to be the number of neighboring pixels whose intensities differ by more than a fixed value. This definition is similar to the texturedness feature used by Engelson [3] for place recognition.
- *Gradient magnitude.* Gradient magnitude is a measure of how rapidly intensity is changing in the direction of greatest change. The gradient magnitude at a pixel (j, k) is computed using standard methods [8].
- *Rank.* The rank of pixel (j, k) is defined as the number of pixels in the local neighborhood whose intensity is less than the intensity at (j, k). This feature is used by Zabih [22] to compute optical flow.

3.2 Discretization of features

An arbitrary feature will have some large (possibly infinite) range of possible values. We would like to discretize its range to produce a smaller number of possible values. It is important to perform this discretization carefully. The range of values should be partitioned so that each discrete value appears with approximately equal likelihood. Such a uniform discretization maximizes the information conveyed by the feature [11].

We discretize a feature by approximating its cumulative distribution over the entire space of images, and dividing the distribution into partitions with equal probability. We generate the approximation by using a large subset of our image database. Each partition of the cumulative distribution is indicative of the range of continuous values which will be treated as a single discrete value. As an example, figure 2 shows the experimental cumulative distribution for the *rank* feature described above, partitioned into four discrete values.

4 Experimental results

The features introduced in section 3 can be combined to produce many distinct joint histograms. We present retrieval results for four different joint histograms, in addition to color

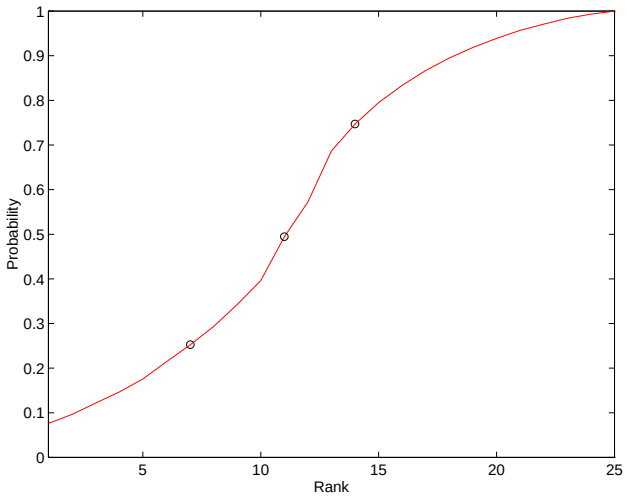

Figure 2: Experimental cumulative distribution of *rank*. The boundaries between the 4 discrete values of rank are marked.

histograms. For convenience, we refer to the joint histograms with particular combinations of properties by the labels:

color, edge density	JH1
color, edge density, texturedness	JH2
color, edge density, texturedness, gradient magnitude	JH3
color, edge density, texturedness, gradient magnitude, rank	JH4

We will label color histograms with CH.

Each of these joint histograms successively incorporates an additional local feature beyond color, from JH1 to JH4. We allowed for 64 possible discrete colors in the image, 4 possible values of edge density, 4 possible values of texturedness, 5 possible values of gradient magnitude, and 4 possible ranks. Color histograms were also implemented with 64 colors.[1] Both color histograms and joint histograms were compared using the L_1 distance.

4.1 Benchmarks

Our image database consists of over 210,000 images. The images were drawn from a variety of sources and with varying resolution and quality. This collection includes the 11,667 images used in Chabot[13], the 1,440 images used in QBIC [4], a 1,005 image database available from Corel, a number of MPEG storyboards, several groups of images taken with a digital camera, and over 180,000 images from CNN, taken once every few seconds.

Most measures used by authors to evaluate retrieval performance, such as precision [17] and match percentile [5, 20], are dependent on the number of images in the database. We believe that a retrieval performance measure should be independent of the number of images. Typically a user is willing to browse a certain number of the retrieval results by hand, similar

[1]We also experimented with different variants of color histograms, involving alternative colorspaces and different numbers of buckets. These yielded essentially similar results.

to text-based search on the web. This number is unlikely to change as the database fluctates in size, as it is really a measure of human patience. We call this number the *scope* of the user. A good performance measure should judge the retrieval method within a particular scope.

We have selected by hand 52 pairs of images which contain different views of the same scene, or different arrangements of the same scene.[2] One image is selected as the query, and the other represents a "correct" answer. For the 52 queries, we ask what percent of the 52 answers were found within a particular scope. The percentage of correct answers is called the *recall* in the information retrieval literature [17]. These results are shown in figure 4. Figure 3 summarizes the data for scopes of 1 and 100. Note that most of the joint histograms have a higher recall level at a scope of 1 than color histograms have for a scope of 100.

Summary	Scope 1	Scope 100
CH	.02	.40
JH1	.33	.83
JH2	.48	.90
JH3	.52	.92
JH4	.60	.94

Figure 3: Recall levels at scopes of 1 and 100. Higher numbers indicate better performance.

Figures 5 and 6 show examples of query images and correct answers. For each pair of images we give the rank of the correct image in the retrieval results according to color histograms and joint histograms. JH4 produced better results than color histograms for all 52 queries, except one (the single case in which color histograms and JH4 both ranked the correct answer first). The average improvement in ranks of JH4 over color histograms was 2,261 positions.

4.2 Efficiency

Summary computation and storage occurs only once per image, and is typically done as a batch process. Summary comparison, in contrast, occurs whenever the user queries the database. In this section, we provide the performance of JH4, the most computationally expensive of the joint histograms, for these distinct phases. We also report the performance of color histograms for comparison. All experiments were run on a 200 MHz Pentium Pro.

Summary computation. The images used for benchmarking were 192×128. Color histograms could be computed at 25 images per second, while JH4's could be computed at just over 7 images per second. The current implementation of joint histograms is unoptimized. For example, the computation speed of the features in JH4 could be substantially improved by making use of dynamic programming [21]. In addition, both color histograms and joint

[2]These images are available at `http://www.cs.cornell.edu/home/rdz/joint-histograms.html`.

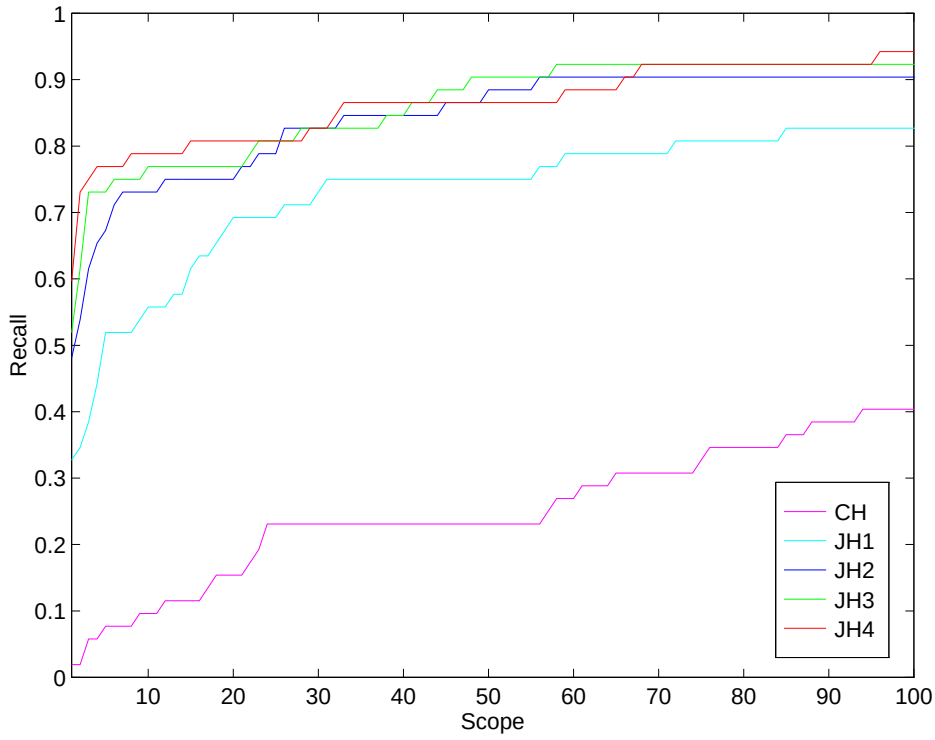

Figure 4: Scope versus recall results on 52 queries. Higher numbers indicate better performance.


Color histogram:
11968
Joint histogram JH4:
5


Color histogram:
308
Joint histogram JH4:
2


Color histogram:
649
Joint histogram JH4:
2


Color histogram:
1896
Joint histogram JH4:
3

Figure 5: Example query images and correct answers, and the rank of the correct answer under the two methods. Lower numbers indicate better performance.

Color histogram:
8629
Joint histogram JH4:
2

Color histogram:
8994
Joint histogram JH4:
1

Color histogram:
7219
Joint histogram JH4:
10

Figure 6: Example query (top left) with multiple correct answers, and the ranks of the correct answers. Lower numbers indicate better performance.

histograms can be computed in parallel. Using three single processor machines, we computed color histograms and all four joint histograms for every image in the 210,000 image database in just under four hours (the images were of varying dimensions).

Storage requirements. While the size of a joint histogram is significantly larger than a color histogram, most of the entries in a joint histogram are zero. The following table shows the total number of entries in color histograms and in several joint histograms, the average percentage of empty entries, and the average number of nonempty entries.

Summary	Entries	Sparseness	Nonempty entries
CH	64	70%	19
JH1	256	75%	64
JH2	1024	82%	184
JH3	5120	89%	563
JH4	20480	93%	1434

While the number of entries in a joint histogram increases substantially with additional features, the actual number of nonzero entries that must be stored remains quite practical.

Summary comparison. Since both color histograms and joint histograms use the same measures for comparison, the efficiency of comparison is the same for both methods. The speed of comparison is determined by the number of histogram entries to be compared. The actual (wall clock) running time on such a large database is dominated by implementation issues, such as I/O strategies. In our current implementation, entries can be compared at over 23 million entries per second, ignoring I/O. This implies that the database of 210,000 images could be queried using color histograms in under 1 second, and queried using JH4 in about 26 seconds. The running time of JH4 could be reduced to 9 seconds using the approximation technique described in section 5.1. This computation is fully parallelizable. In addition, database indexing techniques [7] could be used to avoid comparing the query histogram with the entire database.

5 Extensions

5.1 Reduced intersection

In *reduced intersection* only the c largest entries in each histogram are compared, similar to Swain's *incremental intersection* [20]. Swain shows that for color histograms, comparing only a small number of entries can produce results which closely approximate retrieval using all n entries in the histogram. This reveals that the largest entries in a histogram capture the most distinctive features of the image. Additionally, since the smaller entries in the histogram are more likely to be noise, reducing the number of entries compared can even slightly improve the retrieval results.

If the histogram entries have been sorted and stored prior to the search, the time required for retrieval with a fixed scope is $O(mc)$, where m is the number of images in the database and c is the number of histogram entries used for indexing. For large databases, reduced intersection is considerably faster when $c \ll n$.

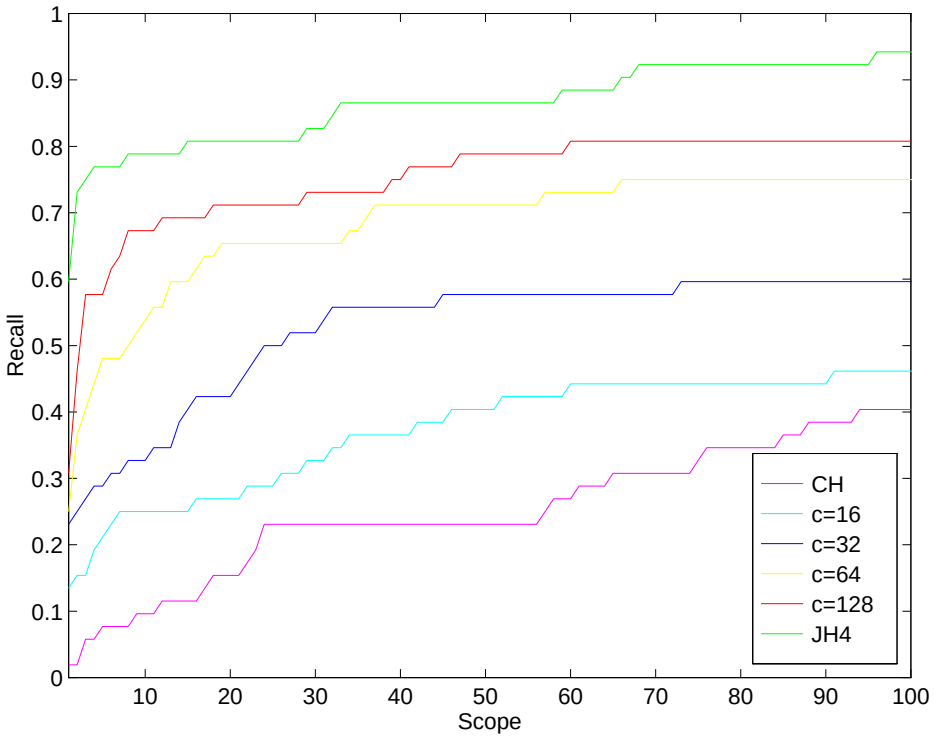

Figure 7: Scope versus recall, reduced intersection results with JH4 on 52 queries.

Figure 7 shows the results of reduced intersection on JH4 with several different values of c. Results from color histograms and JH4 without reduced intersection are shown for comparison. We see that JH4 preforms better than color histograms even when using fewer entries than the number of entries in a color histogram. Reduced intersection with $c = 8$ produces results comparable to those of color histograms. We find that once $c = 512$, the results are comparable to those of using all of the entries in JH4, and the running time for a query is reduced to 9 seconds (ignoring I/O).

5.2 Related applications

Most research in content-based image retrieval has focused on example-based queries. However, other types of queries are also important. For example, it is often useful to search for images in which a subset of another image (e.g. a particular object) appears. This would be particularly helpful for queries on a database of videos.

In [20] Swain uses color histograms to recognize individual objects in an image by comparing the histogram of a query object with the histograms of the images in the database. For the best matches, he then performs histogram backprojection to segment the objects from their backgrounds. Swain recognizes that the histogram comparison can be upset by a pixel in the background in two ways: (1) the pixel has the same color as one of the colors in the query object. (2) the number of pixels of that color in the object is less than the number of pixels of that color in the query object.

Joint histograms reduce the probability that a pixel of a particular color in an object is matched against a pixel of that same color in the background. Different similarly-colored regions of the image will tend to have different local features. Replacing color histograms with joint histograms should therefore improve the results of histogram-based object retrieval.

Additionally, the reduced intersection data in section 4 suggests that only a few entries, or a few pixels, per object may suffice to provide a recognizable cue for that object. This aspect of joint histograms is a direction of future study.

6 Conclusions

Our method is motivated by the problem of image retrieval in large databases. The first experiments with our method were done with a much smaller database containing approximately 18,000 images. The transition from this smaller database to our current collection provided some suggestive data about the way our methods scale. We have measured the increase in rank for our benchmark image pairs that occured when we added almost 200,000 images to our database.

Summary	Average rank increase
CH	2173
JH1	100
JH2	50
JH3	28
JH4	22

As shown in the table above, the average rank increase is substantially smaller for joint histograms than for color histograms. This suggests that our methods may scale to much larger databases without a significant degradation in performance.

Acknowledgments

We wish to thank Frank Wood and the Cornell Theory Center for access to the CNN newsfeed, and Virginia Ogle for access to the Chabot imagery. We also thank Meir Gottlieb and Justin Voskuhl for helping implement the retrieval system.

References

[1] J. R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. C. Jain, and C. Shu. Virage image search engine: an open framework for image management. In *Symposium on Electronic Imaging: Science and Technology - Storage and Retrieval for Image and Video Databases IV*, pages 76–87, 1996.

[2] J. S. Boreczky and L. A. Rowe. A comparison of video shot boundary detection techniques. *Journal of Electronic Imaging*, 5(2):122–128, April 1996. Also appears in SPIE Proceedings number 2670.

[3] Sean P. Engelson and Drew V. McDermott. Image signatures for place recognition and map construction. In *SPIE Sensor Fusion IV*, pages 282–293, 1991.

[4] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, David Steele, and Pater Yanker. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23–32, September 1995.

[5] Brian V. Funt and Graham D. Finlayson. Color constant color indexing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(5):522–529, May 1995.

[6] Arun Hampapur, Ramesh Jain, and Terry Weymouth. Production model based digital video segmentation. *Journal of Multimedia Tools and Applications*, 1:1–38, March 1995.

[7] Joseph Hellerstein, Jeffrey Naughton, and Avi Pfeffer. Generalized search trees for database systems. In *VLDB*, pages 562–573, 1995.

[8] Berthold Horn. *Robot Vision*. The MIT Press, 1986.

[9] Wynne Hsu, T. S. Chua, and H. K. Pung. An integrated color-spatial approach to content-based image retrieval. In *ACM Multimedia Conference*, pages 305–313, 1995.

[10] Jing Huang, S. Ravi Kumar, Mandar Mitra, Wei-Jing Zhu, and Ramin Zabih. Image indexing using color correlograms. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 762–768, 1997.

[11] Edwin T. Jaynes. On the rationale of maximum-entropy methods. *Proceedings of the IEEE*, 70(9):939–952, sep 1982.

[12] David Marr and Ellen Hildreth. Theory of edge detection. *Proc. of the Royal Society of London B*, 207:187–217, 1980.

[13] Virginia Ogle and Michael Stonebraker. Chabot: Retrieval from a relational database of images. *IEEE Computer*, 28(9):40–48, September 1995.

[14] K. Otsuji and Y. Tonomura. Projection-detecting filter for video cut detection. *Multimedia Systems*, 1:205–210, 1994.

[15] Greg Pass and Ramin Zabih. Histogram refinement for content-based image retrieval. In *IEEE Workshop on Applications of Computer Vision*, pages 96–102, December 1996.

[16] Rosalind W. Picard and Tom P. Minka. Vision texture for annotation. *Multimedia Systems*, 3:3–14, 1995. Also appears as MIT Media Lab TR-302, 1994.

[17] Gerard Salton. *Automatic Text Processing*. Addison-Wesley, 1989.

[18] J. R. Smith and S.-F. Chang. VisualSEEK: A fully automated content-based image query system. In *ACM Multimedia Conference*, pages 87–98, November 1996.

[19] Markus Stricker and Alexander Dimai. Color indexing with weak spatial constraints. *SPIE proceedings*, 2670:29–40, February 1996.

[20] Michael Swain and Dana Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[21] J. Webb. Steps towards architecture-independent image processing. *IEEE Computer*, February 1992.

[22] Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. In *3rd European Conference on Computer Vision*, pages 151–158, 1994.

[23] HongJiang Zhang, Atreyi Kankanhalli, and Stephen William Smoliar. Automatic partitioning of full-motion video. *Multimedia Systems*, 1:10–28, 1993.