Multimedia Systems 7: 234–240 (1999)

Multimedia Systems

Comparing images using joint histograms

Greg Pass, Ramin Zabih

Computer Science Department, Cornell University, Ithaca, NY 14853, USA; e-mail: rdz@cs.cornell.edu

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 similar color histograms. 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.

Key words: Content-based indexing and retrieval – Color histograms – Color-spatial indexing

1 Introduction

Many applications require methods for comparing images based on their content. Examples include scene break detection and parsing in video [2, 8, 16, 26] and image database retrieval [1, 5, 15, 18, 20]. 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. In the case of relevance feedback systems [3], example-based retrieval also includes the ability of the user to select images he finds similar to the target image out of a presented set of images. A flexible retrieval system should allow for large changes in the appearance of similar images, as shown in Figs. 4 and 5.

Correspondence to: R. Zabih

Fig. 1. Two images with similar color histograms

Most image retrieval systems operate in two distinct phases.

1. *Image summarization*. 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 [5] 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 to be 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 Fig. 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.

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 Sect. 3, we present joint histograms. In Sect. 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 summarization

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 color space containing n colors. Typically, images are represented in the RGB color space, 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, 5, 15] 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 [23] and 1440 images [5]. However, color histograms have proven less successful on databases with tens of thousands of images [17]. Because a color histogram records only color information, images with similar histograms can have dramatically different appearances, such as those in Fig. 1. In a large database, many unrelated images will happen to have similar color histograms.

Stricker and Swain address this issue in [22]. They define the capacity of a histogram space to be the number of distinguishable histograms that can be stored in an n-dimensional space, given some comparison metric (such as the L_1 distance) and a distance threshold defining similar images. Their color histogram experiments point to a lower bound of around 25,000 histograms when the distance threshold is quite generous. The poor performance of color histograms on our database indicates the capacity of a 64-dimensional histogram space is exceeded by 210,000 images.[1]

Given a constant distance threshold, the only way to increase the capacity of a histogram space is to increase its dimensionality. However, increasing the number of colors in a color histogram sacrifices some degree of robustness. In a high-dimensional histogram space, the same object may be classified as two different colors under slightly different lighting conditions. Ideally, we would like our color buckets to remain large enough to handle slight color changes.

2.2 Other image summaries

In order to increase the performance of color histograms, it is thus necessary to somehow incorporate spatial information. Recently, several authors have proposed such methods. Hsu *et al.* [11] attempt to capture the spatial arrangement of the different colors in the image. The image is partitioned 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 *et al.* present 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 their method requires substantial computation, particularly the partioning algorithm. Additonally, their 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 [21] 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 color space, 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 three queries.

Huang *et al.* [12] 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 *et al.* report good results on a database of over 18,000 images, using an experimental setup closely related to ours.

[1] In fact, our experiments [17] suggest that color histograms perform poorly even on much smaller databases.

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 *et al.*'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 which are largely invariant to changes between similar images (like color), 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 lth 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.

Joint histograms thus increase the dimensionality of the histogram space without changing the capacity of each feature's individual histogram space. This preserves the robustness of each feature, while increasing the capacity of the histogram space.

3.1 Choice of local features

The features we have used were selected empirically based on their invariance to changes within similar images. They can be implemented efficiently in linear time, and lend themselves to parallel programming.

- *Color*. We use the standard RGB color space. Note that any improvements to color histograms (such as better color spaces) 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 [14].
- *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 and McDermott [4] 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 [10].
- *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 and Woodfill [25] 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 partition its range into discrete values that appear with approximately equal likelihood. Such a uniform discretization maximizes the information conveyed by the feature [13].

We discretize a feature by approximating its cumulative distribution over a large space of images, and dividing the distribution into partitions with equal probability. We generate the approximation using a 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.

4 Experimental results

The features introduced in Sect. 3 can be combined to produce many distinct joint histograms. We present retrieval results for four different joint histograms, in addition to color histograms. For convenience, we refer to the joint histograms with particular combinations of properties by the labels

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

We 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, four possible values of edge density, four possible values of texturedness, five possible values of gradient magnitude, and four possible ranks. Color histograms were also implemented with 64 colors. 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[15], the 1,440 images used in QBIC [5], a 1,005-image database available from Corel,

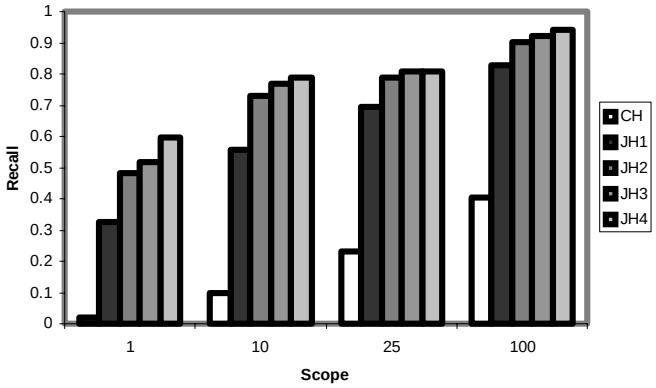

Fig. 2. Scope versus recall results on 52 queries. Higher numbers indicate better performance

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 [19] and match percentile [7, 23], 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 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 percentage of the 52 answers were found within a particular scope. The percentage of correct answers is called the *recall* in the information retrieval literature [19]. These results are shown in Fig. 2. 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.

Figures 4 and 5 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,

[2] These images can be obtained on the web from the URL `http://www.cs.cornell.edu/home/rdz/joint-histograms.html`.

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

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

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 JH4s 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 [24]. In addition, both color histograms and joint 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 4 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. For both color histograms and joint histograms, 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 s, and queried using JH4 in about 26 s. The running time of JH4 could be reduced to 9 s using the approximation technique described in Sect. 5.1. This computation is fully parallelizable. In addition, database indexing techniques [9] could be used to avoid comparing the query histogram with the entire database.

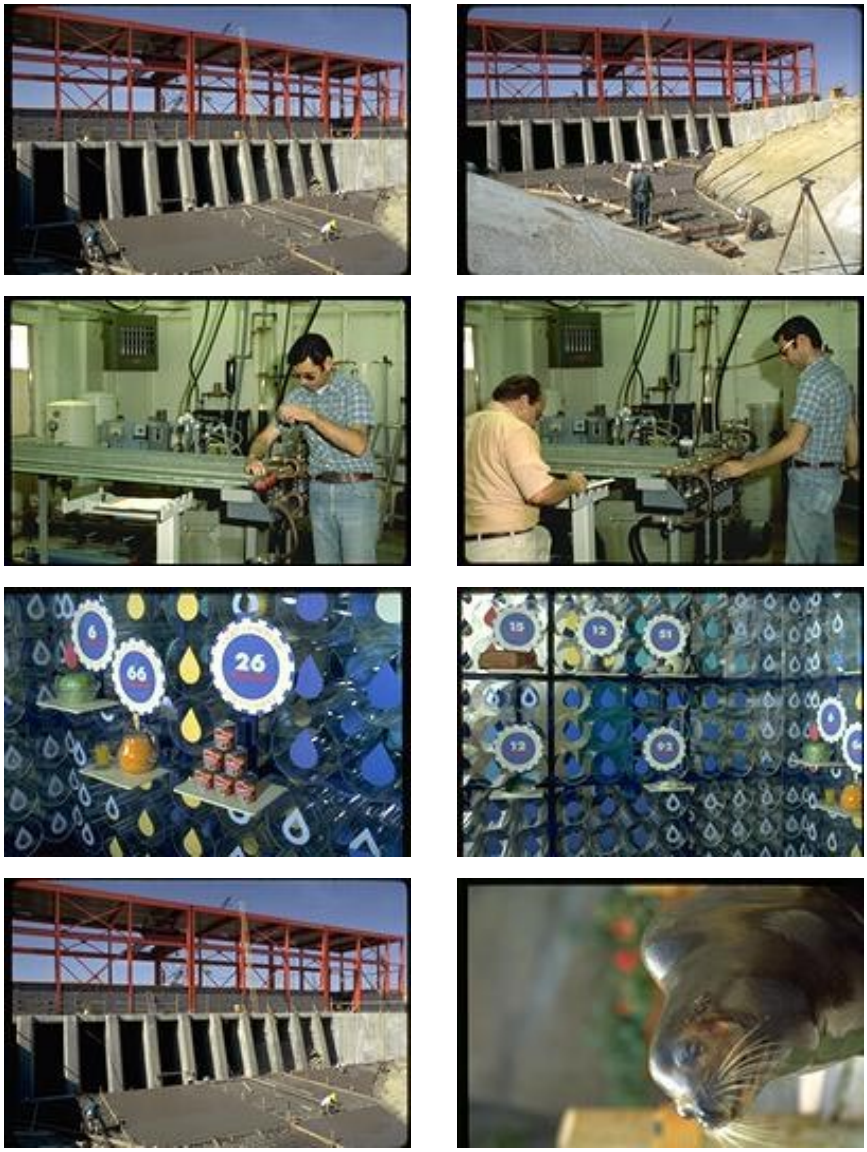

Fig. 4. Example query images and correct answers, and the rank of the correct answer under the two methods. Lower numbers indicate better performance

5 Extensions

5.1 Reduced intersection

In *reduced intersection*, only the c largest entries in each histogram are compared, similar to Swain and Ballard's *incremental intersection* [23]. They show 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 6 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 performs 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 s (ignoring I/O).

We are currently investigating more sophisticated methods for reducing the dimensionality of the data. While reduced intersection uses only the size of each histogram entry as a measure of distinctiveness, methods such as principal-components analysis [6] exploit the statistical structure of the entire space.

Fig. 5. Example query (top left) with multiple correct answers, and the ranks of the correct answers. Lower numbers indicate better performance

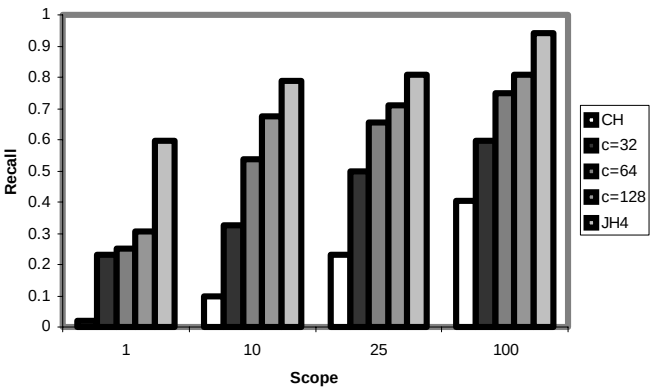

Fig. 6. Scope versus recall, reduced intersection results with JH4 on 52 queries

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 [23], Swain and Ballard use 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, tey then perform histogram backprojection to segment the objects from their backgrounds. Swain and Ballard recognize 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 suggests that only a few entries, or a few pixels, per object may suffice to provide a recognizable cue for that object.

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 average 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.

Acknowledgements. 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. This research has been supported by DARPA contract DAAL01-97-K-0104.

References

1. Bach JR, Fuller C, Gupta A, Hampapur A, Horowitz B, Humphrey R, Jain RC, Shu C (1996) Virage image search engine: an open framework for image management. In: Jain R (ed) Symposium on Electronic Imaging: Science and Technology – Storage and Retrieval for Image and Video Databases IV, 1996, San Jose, CA, USA, pp 76–87
2. Boreczky JS, Rowe LA (1996) A comparison of video shot boundary detection techniques. J Electron Imaging 5(2): 122–128
3. Cox IJ, Miller ML, Omohundro SM, Yianilos PN (1996) PicHunter: Bayesian relevance feedback for image retrieval. In: International Conference on Pattern Recognition, 1996, Vienna, Austria, pp 361–369
4. Engelson SP, McDermott DV (1991) Image signatures for place recognition and map construction. In: Schenker PS (ed) SPIE Sensor Fusion IV, 1991, Boston, MA, USA, pp 282–293
5. Flickner M, Sawhney H, Niblack W, Ashley J, Huang Q, Dom B, Gorkani M, Hafner J, Lee D, Petkovic D, Steele D, Yanker P (1995) Query by image and video content: The QBIC system. IEEE Comput 28(9): 23–32
6. Fukunaga K (1990) Introduction to Statistical Pattern Recognition. Academic Press, London
7. Funt BV, Finlayson GD (1995) Color constant color indexing. IEEE Trans Pattern Anal Mach Intell 17(5): 522–529
8. Hampapur A, Jain R, Weymouth T (1995) Production model based digital video segmentation. J Multimedia Tools Appl 1: 1–38
9. Hellerstein J, Naughton J, Pfeffer A (1995) Generalized search trees for database systems. VLDB J 562–573
10. Horn B (1986) Robot Vision. MIT Press, Cambridge, Mass.
11. Hsu W, Chua TS, Pung HK (1995) An integrated color-spatial approach to content-based image retrieval. In: ACM Multimedia Conference, 1995, San Francisco, CA, USA, pp 305–313
12. Huang J, Kumar SR, Mitra M, Zhu W-J, Zabih R (1997) Image indexing using color correlograms. In IEEE Conference on Computer Vision and Pattern Recognition, 1997, San Juan, PR, USA, pp 762–768
13. Jaynes ET (1982) On the rationale of maximum-entropy methods. Proc IEEE 70(9): 939–952
14. Marr D, Hildreth E (1980) Theory of edge detection. Proc R Soc Lond B 207: 187–217
15. Ogle V, Stonebraker M (1995) Chabot: Retrieval from a relational database of images. IEEE Comput 28(9): 40–48
16. Otsuji K, Tonomura Y (1994) Projection-detecting filter for video cut detection. Multimedia Syst 1: 205–210
17. Pass G, Zabih R (1996) Histogram refinement for content-based image retrieval. In: IEEE Workshop on Applications of Computer Vision, December 1996, Sarasota, FL, USA, pp 96–102
18. Picard RW, Minka TP (1995) Vision texture for annotation. Multimedia Syst 3: 3–14
19. Salton G (1989) Automatic Text Processing. Addison-Wesley, Reading, Mass.
20. Smith JR, Chang S-F VisualSEEK: A fully automated content-based image query system. In: ACM Multimedia Conference, November 1996, Boston, MA, USA, pp 87–98
21. Stricker M, Dimai A (1996) Color indexing with weak spatial constraints. In: Jain RC (ed) Proc SPIE vol. 2670, pp 29–40, San Jose, CA, USA
22. Stricker M, Swain M (1994) The capacity of color histogram indexing. In: IEEE Conference on Computer Vision and Pattern Recognition, 1994, Seattle, WA, USA, pp 704–708
23. Swain M, Ballard D (1991) Color indexing. Int J Comput Vision 7(1): 11–32
24. Webb J (1992) Steps towards architecture-independent image processing. IEEE Comput 25(2): 21–31
25. Zabih R, Woodfill J (1994) Non-parametric local transforms for computing visual correspondence. In: Eklund J-O (ed) 3rd European Conference on Computer Vision, 1994, Stockholm, Sweden, pp 151–158; Revised version available from www.cs.cornell.edu/home/rdz
26. Zhang HJ, Kankanhalli A, Smoliar SW (1993) Automatic partitioning of full-motion video. Multimedia Syst 1: 10–28

GREG PASS received his BA in computer science from Cornell University in 1997. He is currently a researcher at MITRE Corporation in Washington D.C. His research interests are in computational neuroscience and computer vision.

RAMIN ZABIH is an assistant professor of Computer Science at Cornell University. He received undergraduate degrees in Computer Science and Mathematics, and a master's degree in Computer Science, from MIT. His PhD was awarded by Stanford in 1994. His research interests are in computer vision and its applications. He has served on the program committees of conferences in both multimedia and computer vision.