Multimed Tools Appl (2007) 34:239–248
DOI 10.1007/s11042-006-0087-2

Content-based image retrieval using joint correlograms

Adam Williams · Peter Yoon

Published online: 7 February 2007

Abstract The comparison of digital images to determine their degree of similarity is one of the fundamental problems of computer vision. Many techniques exist which accomplish this with a certain level of success, most of which involve either the analysis of pixel-level features or the segmentation of images into sub-objects that can be geometrically compared. In this paper we develop and evaluate a new variation of the pixel feature and analysis technique known as the color correlogram in the context of a content-based image retrieval system. Our approach is to extend the autocorrelogram by adding multiple image features in addition to color. We compare the performance of each index scheme with our method for image retrieval on a large database of images. The experiment shows that our proposed method gives a significant improvement over histogram or color correlogram indexing, and it is also memory-efficient.

Keywords Content-based image retrieval · Color histograms · Correlograms

1 Introduction

The problem addressed by a content-based image retrieval system is simple: given a query image, find a set of target images which appear most similar to the input image, based solely on the content of the images, that is, with no text information involved. This problem has been gaining its popularity as a vast amount of digital image data are now available for analysis in many disciplines. The applications of

A. Williams' work was supported by Trinity College Student Research Grant Program.

P. Yoon's work was based upon work supported by NASA EPSCoR Core Funding Program.

A. Williams · P. Yoon (✉)
Department of Computer Science, Trinity College, Hartford, CT 06106, USA
e-mail: peter.yoon@trincoll.edu

efficient retrieval systems are very broad, including everything from law enforcement to medical diagnostics.

A typical retrieval system is generally composed of two main components: the indexer and the query engine. Effective indexing is the most important part of any retrieval system. Care must be taken so that each image's index entry captures the distinguishing features of an image while using only a small amount of storage. In this system, the common approach of storing a feature vector for each image is adopted since it allows for the use of standard matrix structures and measures of distance in comparisons. The feature vectors are then stored in the database prior to retrieval by the query engine.

One of the most commonly used method for constructing feature vectors is *color histogramming* [2, 11, 13]. A color histogram is a vector where each entry stores the number of pixels of a certain color in the image. A typical image is represented in the RGB color space. Color histograms are simple to compute and also tolerant to rotation and movement of objects within an image. They are compared using L_1 or L_2 distance measure and effective in querying small databases. However, the plain color histogram loses its discriminatory power when dealing with very large databases. Since the color histogram only captures a global color distribution of an image, two entirely different images may have very similar histograms.

A natural extension to color histogram is to include other features of an image such as intensity and edge density. This technique is known as *joint histogramming* [9]. Joint histograms are generally two to three times more effective than simple color histograms. The reason for the improvement is clear: while dissimilar images may have similar color distributions, it is less likely that the joint distributions of multiple features will be similar.

Both color histogram and joint histogram only store global information about an image. They do not describe a spatial distribution in an image. Thus, they are sensitive to large changes in appearance in the image caused by camera viewpoint, camera zoom, etc [6]. To address this problem the *color correlogram* adds a dimension to the feature vector which represents the relative local distance between the pixels with the specific color. Each entry of color correlogram includes the information on how color is spatially distributed in an image. Thus, two images with similar color histograms may have entirely different color correlograms if colors appear in entirely different locations in the image. The correlogram method is generally more accurate and effective than histogam-based methods [6].

In this paper, we describe a novel technique for feature analysis which is an extension of histograms and color correlograms. We propose a multi-feature correlogram technique called a *joint correlogram* for image comparison and ranking. The joint correlogram captures the joint distribution of several features of the image such as texturedness, gradient magnitude, rank, and color. This approach gives a more comprehensive representation of an image than capturing just the marginal distributions of the features, where each feature is treated independently. We show that our approach for general retrieval system is suitable for use with a large heterogeneous set of images. Our preliminary results demonstrate that the joint correlogram is a more effective method for analyzing and computing features when compared to both histogram and the color-only correlogram.

In the following section, we review several conventional techniques for feature analysis and indexing and describe joint correlogram. The joint correlogram is more

accurate but computationally intensive and requires more space than histograms and color correlograms. Thus, we consider in Section 3 a preprocessing technique that will efficiently reduce the dimensionality of the data. Finally, we give the results from our experiments with joint correlogram for a database of over 24,000 images.

2 Features computation

2.1 Overview

Before the images can be indexed, the features must be quantized into smaller spaces. For example, a 24-bit image in the RGB color space has over 16 million color possibilities, which would require an unnecessarily large amount of storage for the histogram. The solution is to map each color into a smaller space; this must be done in such a way that each discrete value appears equally as often as the others [1], that is, based on the probability distribution of possible values. This must be done for each feature to be indexed.

Another step in indexing an image is the preprocessing phase, which will be described in the next section. This involves decompressing the image from whatever format is stored in and loading it into memory so that it can be operated on. Next, the image must be normalized or scaled to a predetermined size so that different sized images can be accurately compared. Once an image is in memory and has been scaled, its features can be analyzed and computed.

2.2 Histograms

An image histogram refers to the joint probability density function of the image features. Let $\mathcal{I}_N$ be a set of images of N pixels. Suppose we consider m image features (random variables): $F_1, \ldots, F_m$. The most popular choice of images features is color. Other features include texturedness, gradient, edge density, and rank. The histogram of an image $P \in \mathcal{I}_N$ is defined as

$$h_{F_1,\ldots,F_m}(f_1, \ldots, f_m) = N \cdot \text{Prob}(F_1 = f_1, \ldots, F_m = f_m) \tag{1}$$

Thus, h is a vector which contains $n_1 * \cdots * n_m$ entries, where n_i is the number of possible values of F_i for $i = 1, \ldots, m$.

When $m = 1$ and F_1 represents color, h is the familiar color histogram. In this case, the color histogram is an n_1-vector where each entry stores the number of pixels in a particular color. The L_1 or L_2 distance measure can be used to compare histograms.

The color histogram is one of the first feature analysis techniques used in image retrieval [13]. Color histograms record the distribution of colors in an image, and their effectiveness is based on the fact that similar images will have similar distributions. This technique is moderately effective in practice has been successfully implemented in several early systems [2–4, 11].

The histogram method is based on the assumption that the distance between two histograms for two unrelated images must be large. However, it is not always true in practice. Color histograms are less effective when dealing with large sets of image data as unrelated images may have the similar histograms since color histograms

only store color information[12]. We chose five images at random from our image database of over 24,000 images and compared their histograms with the rest. Our test showed that the number of histograms of unrelated images which were similar to those of each test image exceeds 50 in most cases. Even with a randomly generated image, we were able to match its histogram with those of eight other images in the database.

When $m > 1$, we have an extension of this method, known as a *joint histogram* [9], which uses other pixel features in addition to color when computing histograms. The experiments showed that the joint histogram with five features (color, edge density, texturedness, gradient magnitude, rank) are able to render over 85% recall with moderate values of scope, a significant improvement over color histogramming. Other approaches in joint histogramming have also been investigated [5, 7]. This is an important concept which motivates the new approach taken in this paper.

2.3 Joint correlograms

The joint correlogram technique is similar in spirit to the joint histogram. The difference is that the correlogram includes spatial information about the image features in addition to global information about the features. Since histograms do not store the spatial information, they are sensitive to large appearance changes of the image. For example, two images taken in different viewing positions may have very different histograms as shown in Fig. 1.

Correlograms record the probability of finding a pixel with a certain set of values for the given set of features at a certain distance from a given pixel with another set of values for the same set of features. The pixels have, in most cases, entirely different set of values for the given set of image features. Let p_1 and p_2 be any two pixels of an image. Then the correlogram of the image is defined as

$$c^{(k)}_{F_1^{(1)},\ldots,F_m^{(1)},F_1^{(2)},\ldots,F_m^{(2)}}\left(f_1^{(1)},\ldots,f_m^{(1)},f_1^{(2)},\ldots,f_m^{(2)}\right) = \text{Prob}\Big(F_1^{(1)}=f_1^{(1)},\cdots,F_m^{(1)}=f_m^{(1)},F_1^{(2)}=f_1^{(2)},\cdots,F_m^{(2)}=f_m^{(2)},\ |p_1-p_2|=k\Big) \tag{2}$$

Fig. 1 Two images taken in different viewing positions

where $F_1^{(i)}, \ldots, F_m^{(i)}$ are random variables which represent m features of $p_i, i = 1, 2$, and k is the distance between the pixels.

Using this general approach with all possible feature combinations is computationally expensive. From (2) we see that the correlogram c is of length $(\prod_{l=1}^{m} n_l)^2$, which is impractical in a real CBIR system. Here, $n_l, l = 1, \ldots, m$ are the number of possible values of $F_l^{(i)}$ for $i = 1, 2$. So, in practice just autocorrelograms are used [10]. Autocorrelograms only involve the pixels to those having the same set of values for a given set of features. Thus, the length of c is reduced to $\prod_{l=1}^{m} n_l$.

Let $\mathcal{I}_{F_i(j)}$ denote the set of pairs of pixels which have the same value j for the feature F_i. For example, if F_1 represents color and j is a color from the given set of colors in the image, both pixels have the same color j. Autocorrelograms are then fairly simple to compute using the following expression,

$$\frac{\left|\{(p_1, p_2) \in \mathcal{I}_{F_i(j)} | \, |p_1 - p_2| = k\}\right|}{8k \cdot N} \tag{3}$$

Here, the numerator represents the number of pairs of pixels which have the same value for feature F_i and are k distance away. The denominator represents the total number of pairs of pixels which are k distance away in the image (there are $8k$ pixels on the border of the $2k \times 2k$ window centered at a given pixel when L_∞ is used for distance measure).

When $m = 1$ and F_1 represents color, we have the color correlogram described in [6], where methods for computing color correlograms, which are similar to (3), are also presented. This technique has been shown to be superior in recall to color histograms and color coherence vectors although it was not compared to joint histograms. Also, small distance values must be chosen for the correlogram computation as small values are most efficient and effective.

Our approach is to extend the autocorrelogram by joining multiple image features when $m > 1$, thereby, *joint autocorrelogram*. Specifically, the autocorrelograms will be computed using color, texturedness, gradient magnitude, and rank, all as defined in [6]. It is expected that using the additional feature information will increase the recall accuracy just as joint histograms improve over color histograms, although not necessarily in as dramatic as fashion. This technique will be compared to the standard correlogram and joint histogram techniques in a manner described later.

Four elementary features are used in this paper: color, gradient magnitude, rank and texturedness. The RGB color space is used, quantized into a set of discrete values. Gradient magnitude measures the rate at which intensity at a given pixel is changing in the direction of maximum change. It can be calculated from grayscale versions of the images as the maximum of vertical and horizontal neighboring pixel differences. For a pixel $p = (x, y)$ its gradient magnitude can be approximated by

$$\max\{|g(x, y) - g(x + 1, y)|, |g(x, y) - g(x, y + 1)|\} \tag{4}$$

where $g(x, y)$ is the intensity at p in grayscale. The rank quantifies the intensity variation in the neighborhood of pixels within an image. The rank of a pixel can be determined as the number of pixels in its neighborhood whose gray-level intensity is less than the pixel of interest. Texturedness is defined as the number of pixels in the neighborhood whose gray-level intensity exceeds by a certain level from the pixel of interest.

All four features can be calculated simultaneously in a single pass through the image. If we consider the window size of rank and texturedness to be fixed, the time complexity of calculating all four features over a given image is a constant multiple of the image size. Building the correlograms begins with computing all four features for the entire image as is done for the histograms.

A second pass is then made through the image (actually through the precomputed feature values) to compute the actual autocorrelogram. The time complexity of this phase depends on the number of distances chosen to compute the correlograms with, and in the worst case is on the order of the size of the image.

For the sake of comparison and also for a querying efficiency reason which will be described later, both the joint histograms and joint correlograms of the images will be computed and stored on disk. Joint correlograms are computed using (3). While the size of the histograms and correlograms are not particularly large, a sparse matrix storage structure can be used to reduce storage requirements significantly. It is generally preferable to store histogram entries in some kind of sorted order for querying purposes [8]. The histograms and correlograms are computed once for each image, generally in a large batch job. New images can be added by performing the computation on just the new images. Once the images have been indexed, they are ready for querying.

3 Performance evaluation

Querying is performed by simply computing the L_1 distance between the query image's feature vector and the feature vectors of each of the images in the database. Those with the smallest distance are considered to be the best matches.

The system is accessible via a simple Web interface. An example image can be selected from the database and queried upon, and query results can be browsed. The interface will also allow the user to select which indexing method to use in the query. This will make it easier to evaluate the joint correlogram's performance. All of the computation is done on the server-side. The client simply provides the query images to the server and receives a list of results.

For testing the system, 24,000 jpeg images of varying content have been obtained from the Berkeley Digital Library Project.[1] Our CBIR system operated on machine with a 2.19 GHz AMD Athlon CPU. Computing the histogram indicies took approximately 7 min for the 24,000 jpeg images in our database, which includes preprocessing of each image into raw bitmaps and linear scaling to 100×100 pixels (most were around 400×400 pixels to begin with).

When constructing feature vectors, colors were quantized into 64 discrete values. We used the same number of color bins for all four feature extraction strategies. For joint histograms and joint correlograms three additional features were added. Gradient magnitude of each pixel was calculated using (4). The rank of a pixel was determined by computing the number of pixels in a 5 by 5 neighborhood whose gray-level intensity is less than the pixel of interest. Texturedness was computed by counting the number of pixels in a 5 by 5 neighborhood that differ in gray-level intensity by more than 20 from the pixel of interest. The values of these

[1] http://elib.cs.berkeley.edu

additional features were quantized into 5 discrete values. The parameters used in constructing feature vectors were empirically chosen to keep the size of the vectors computationally reasonable so that indexing can be done efficiently and yet still maintaining good performance when retrieving the images from the database.

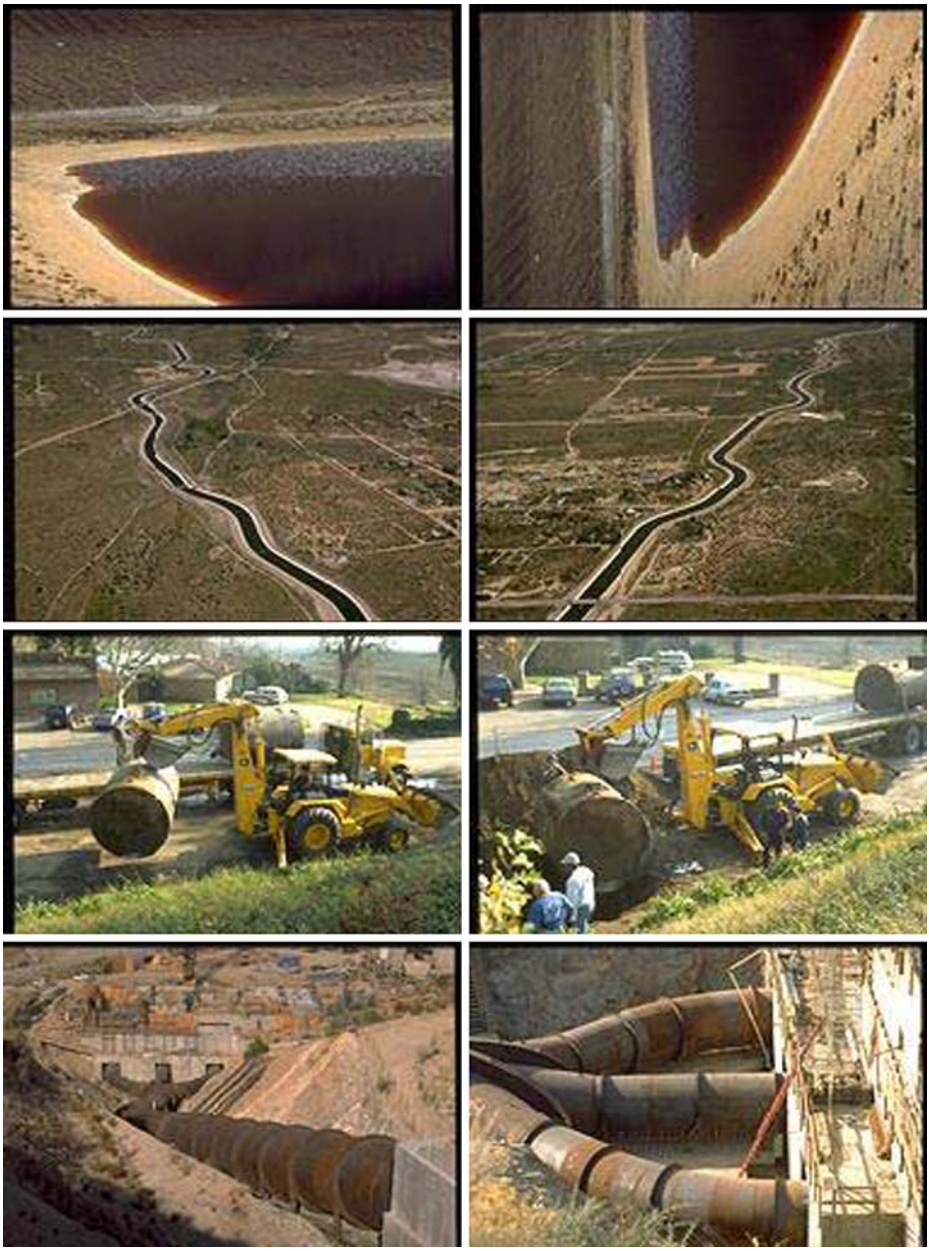

Fig. 2 Query images and examples of the correct answers

Computing the correlogram indicies took approximately 35 min, which includes the computation of histograms and preprocessing. The joint histogram index required 5 MB of disk space versus 65 MB for the joint correlogram. The small size of the indicies allowed them to be loaded entirely into memory while the system was operating, making querying very fast (less than 1s per query).

In evaluating and comparing CBIR systems, we used the standard measure *scope* vs. *recall*. Scope refers to how many images deep into the results one has to look before finding the "correct" result (the correct image is determined by human comparison). Thus in scope vs. recall analysis, one is interested in what percentage of correct images appear in a given scope for several different test queries. Ideally, 70% of correct images should appear in a scope of 10 (that is, within the first 10 images returned by a query). This analysis allows for a simple and practical comparison of several image summary techniques. It will be used to evaluate the performance of the joint correlogram and compare it to histograms and color correlograms.

The joint correlogram method was tested against the color histogram, joint histogram and the color correlogram methods using scope vs. recall analysis. Approximately 30 randomly chosen images were used to query the system using each of the four methods; the best matches for these images were determined manually beforehand.

Examples of query images and correct answers are shown in Fig. 2. By comparing the rank of the correct images produced by each method, we observed that the joint correlogram produced the best results for all 30 queries. In most cases, the rank of the correct images were getting better in the order of color histogram, joint histogram, color correlogram, and joint correlogram. In particular, for the last pair of the example images in Fig. 2, where a significant changes were made to the image including the camera angle, viewpoint, and camera zoom, none of the methods could retrieve the correct answer within the scope except for the joint correlogram.

Further, as shown in Fig. 3, approximately 80% of the best matching images appeared within the first 100 results returned by the system when using the joint

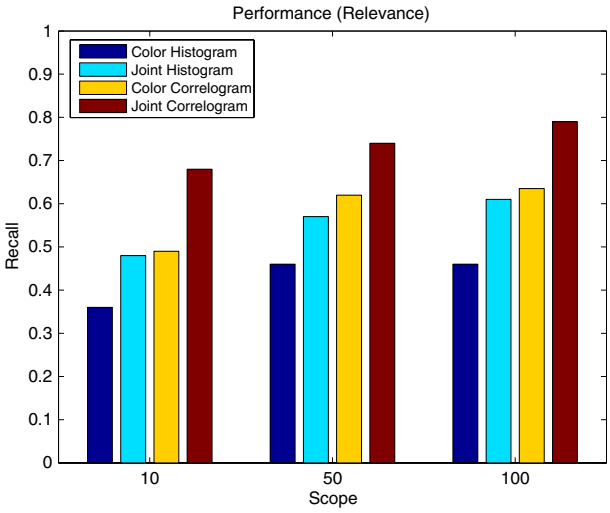

Fig. 3 Performance evaluation of four CBIR systems

correlogram method. Recall percentages of 61, 59, and 49% were achieved by the color correlogram, joint histogram, and color histogram methods, respectively.

4 Conclusion

In this paper we presented a comprehensive approach to feature extraction of digital images in the context of content-based image retrieval systems. The joint correlograms utilize multiple features of an image so that images can be uniquely represented in the database. This ensures an accurate retrieval of relevant images for the query. The memory requirements for the joint correlogram are somewhat greater than other existing approaches, namely, color histogram, joint histogram, and color correlogram. The speed of each query is nearly identical regardless of what method is used. Overall, we have determined that the joint correlogram indexing method gives a noticeable improvement over histogram or color-only correlogram indexing, and its time and space requirements are still more than practical.

References

1. Chang C-Y, Maciejewski AA, Balakrishnan V (2000) Fast eigenspace decomposition of correlated images. IEEE Trans Image Process 9:1937–1949
2. Ciocca G, Schettini R (1999) A relevance feedback mechanism for content-based image retrieval. Inf Process Manag 35:605–632
3. 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 Computer Society Press, Los Alamitos, CA
4. Gevers Th, Smeulders AWM (1999) Color based object recognition. Pattern Recogn 32:453–464
5. Gupta A, Jain R (1997) Visual information retrieval. Commun ACM 40:71–79
6. Huang J, Kumar SR, Mitra M, Zhu W-J, Zabih R (1997) Image indexing using color correlograms. In: Proceedings of the 1997 conference on computer vision and pattern recognition, pp 762–768. IEEE Computer Society, Washington, DC
7. Jain AK, Vailaya A (1996) Image retrieval using color and shape. Pattern Recogn 29:1233–1244
8. Pass G, Zabih R (1996) Histogram refinement for content-based image retrieval. In: Proceedings of the 3rd IEEE workshop on applications of computer vision, pp 96–102. IEEE Computer Society, Washington, DC
9. Pass G, Zabih R (1999) Comparing images using joint histograms. Multimedia Syst 7:234–240
10. Pass G, Zabih R, Miller J (1996) Comparing images using color coherence vectors. In: Proc. ACM Intern. Conf. Multimedia, pp 65–73. ACM Press, New York, NY
11. Scalaroff S, Taycher L, La Cascia M (1997) Imagerover: a content-based image browser for the world wide web. In: IEEE workshop on content-based access and video libraries. IEEE Computer Society, Washington, DC
12. Stricker M, Swain M (1994) The capacity of color histogram indexing. IEEE Press, Piscataway, NJ (pp 704–708)
13. Swain MJ, Ballard BH (1991) Color indexing. Int J Comput Vis 7:11–32

Adam Williams graduated Phi Beta Kappa from Trinity College in 2004 with a B.S. in Computer Science and Mathematics. He joined the Computer Vision Laboratory in the Department of Computer Science at the University of Massachusetts—Amherst as an M.S./Ph.D. student in the fall of 2004. His current work involves building distributed networks of smart cameras for assistive technology applications such as fall detection and health monitoring. His general interests include human tracking, activity recognition and classification, and recovering information about an environment from tracking data.

Peter Yoon received his B.Sc. degree in computer science and B.Sc. degree in applied mathematics both from North Carolina State University in 1986, his M.Sc. degree in mathematics from Purdue University in 1988, and his Ph.D. degree in computer science from Pennsylvania State University (PSU) in 1995. From September 1991 to July 1995 he was a research fellow at the Applied Research Laboratory, PSU, where he developed numerical algorithms for guidance and control systems for underwater signal processing. Since then he has taught at Azusa Pacific University, Azusa, California, and he is currently an associate professor at Trinity College, Hartford, Connecticut. His current research interests include scientific computing, signal processing, and information retrieval.