CLUSTERING of TEXTURE FEATURES for CONTENT BASED IMAGE RETRIEVAL

Erbug Celebi, Adil Alpkocak

Dokuz Eylul University
Department of Computer Engineering
35100 Bornova, Izmir, TURKEY
{celebi,alpkocak}@cs.deu.edu.tr

Abstract. Content-based image retrieval has received significant attention in recent years and many image retrieval systems have been developed based on image contents. In such systems, the well-known features to describe an image content are color, shape and texture.

In this paper, we have studied an approach based on clustering of the texture features, aiming both to improve the retrieval performance and to allow users to express their queries easily. To do this, the texture features extracted from images are grouped according to their similarities and then one of them is chosen as a representative for each group. These representatives are then given to users to express their query. Besides the detailed descriptions of clustering process and a summary of results obtained from the experiments, a comparison about statistical texture extraction methods and effects of clustering to them are also presented.

1 Introduction

Image databases are becoming increasingly popular due to large amount of images that are generated by various applications and the advances in computation power, storage devices, scanning, networking, image compression, and desktop publishing. The typical application areas of such systems are medical image databases, photo clip archives, art images, textile pattern archive, photojournalism, WWW, and etc. All these fields need better techniques and mechanism to store and retrieve such huge amount of images.

The early implementation of image databases were based on simply giving descriptive keywords to each image, and allowing users to make query on these keywords for accessing the images. However, this method has some deficiencies such as subjectivity and labor-intensive nature of the keywords assigning process. Moreover, it is sometimes almost impossible to describe content of an image by words, especially for textures found in an image. As a solution to these problems, content-based image retrieval method is proposed based on image features extracted automatically from images. These well-known features are local color, global color, structures in the image (i.e. shape.) and textures found in an image. In this study we focused on textural features of images and aimed both to improve retrieval performance and help users to express their queries easily.

An image database may contain thousands of textured images. The main problem a user faced is locating the images having similar texture given in the query. More clearly, this problem has two main parts: (1) finding the images having the similar texture given in query and (2) specifying a texture in query. A good image retrieval system dealing with textures must provide solutions to both problems.

Specifying the requirement is not a trivial task for querying images since image are hard to describe in nature. Moreover, the information is meaningful only when it can be retrieved through an expressive query. In forming an expressive query for texture, it is quite unrealistic to expect the user to draw the texture (s)he wants. In our approach, all the textures extracted from the database are classified into clusters and one of the textures in each cluster is chosen as a representative. These representative textures can be presented to the user and asked to choose the closest one that of the requested image. In this way, textures can be used for expressive querying.

Although several researchers have been working on texture classification, none of them is aiming to use texture classification for expressive querying. Smith et. al. [9] proposed a method for classification and discrimination of textures based on the energies of image subbands. They also proposed that texture classification may be used for indexing large databases.

In this study, we have used gray level single-textured images to extract their features and construct a feature vector by using co-occurrence matrixes for each textured image. Then, feature vectors are clustered into groups by using hierarchical clustering techniques and one of the textures is selected as a representative for each group. These representatives are then used in user interfaces for query forming process to narrow down the entire search space. In this way, a much smaller subset of the whole database is given to user to query. Cluster based retrieval also has some advantages on retrieval performance. Our experimentation showed that the results are promising.

In the remainder of this paper, first, texture features and their extraction methods are discussed. In section three clustering process of texture features are presented. The results obtained from our experimentation are presented in section four. Section five gives a look to future works and concludes the paper.

2 Representation of Textures Features

Query submitted by the user will be executed on feature vectors rather than on images themselves. Therefore, the representation of texture is directly related to system performance. However, the features must be extracted first via texture analysis that provides an algorithm for extracting texture features from images.

Haralick proposed the usage of co-occurrence matrixes for texture feature representation [4]. This approach explored the gray level spatial dependencies of texture. It is first constructed a co-occurrence matrix based on the orientation and distance between image pixels and then extracted meaningful statistics from the matrix as texture representation. The feature vector for a texture is constructed from many matrices for different orientation and distance. In the literature there are many other approaches for extracting texture features like wavelet transformations [9], spatial/spatial frequency (*s/s-f*) subbands [8], Markov Random Fields [2], Gabor wavelet [5].

A feature is an attribute that characterizes a specific property of an object. An *n*-dimensional feature vector represents an object, where *n* is the selected numbers of attributes. An object may be image, video, sound and etc. More formally, an image *I*, can be represented as following feature vector:

$I : < e_0, e_1, e_2, e_3, e_4, e_5, e_6, e_7, \ldots >$, where each entry ($e_i$) of this vector represents a feature of image *I*.

Spatial placement of pixels are used to make texture analysis. The information about the spatial placement of pixels can be summarized in two dimensional co-occurrence matrices computed for different distances and orientations. In the following subsection, we give a short description on Gray Level Co-occurrence matrices, which is a well-known statistical texture feature extraction techniques and provide a comparison among them.

2.1 Gray Level Co-occurrence matrix

The Gray Level Co-occurrence matrix (GLCM) contains the information about gray levels (intensities) of pixels and their neighbors, at fixed distance and orientation. The idea is to scan the image and keep track of gray levels of each of two pixels separated within a fixed distance *d* and direction θ. But only one distance and one direction generally are not enough to describe textural features. So, we have used more than one direction and distance. It is common to use four directions horizontally and vertically and, two for diagonals. Most of the researchers use four directions and five distances [1, 4]. In our study, we have used four matrices for every value of distance *d* and four matrices for direction θ. We have represented an image by totally 16 matrices.

Each matrix is 256×256 in size assuming that images are in 256 gray levels. But each of matrix, which is 256×256 in size, requires a huge memory to store and it is a time consuming task to produce the matrix. Therefore, we first convert the images in to 16 gray-levels and then produce a co-occurrence matrix for this new image instead of the original one. This reduction allows us to work with GLCM matrices 16×16 in size. Our experiments show that converting the image from 256 in to 16 gray level does not affect the texture query results. However, these matrices are still containing much data (each matrix has got 16×16=256 entries) and needs to be reduced. What is usually done is, to analyze these matrices and compute a few simple numerical values that encapsulates the information about matrixes. Some statistics computed from

GLCM can be used instead of the whole matrix. The gray level co-occurrence matrix is determined as follows:

Let $D_x = \{0, 1, \ldots, N_x-1\}$ and $D_y = \{0, 1, \ldots, N_y-1\}$ be the spatial domains of row and column dimensions respectively, where N_x and N_y are the number of pixels in axis X and Y respectively. And, $G = \{0, 1, \ldots, N_g-1\}$ be the domain of gray levels where N_g is the number of gray levels. The Image I can be represented as a 2D function; $I{:}D_x \times D_y \rightarrow G$. For abbreviation, a new domain can be defined, as $D \subset N^2$ (where N is the set of Natural numbers) instead of $D_x \times D_y$. Positions and orientations are shown in Figure 1 and Figure 2.

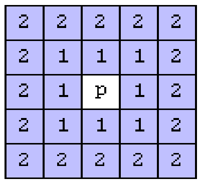


Fig.1. Distances of pixel *p* for co-occurrence matrix.

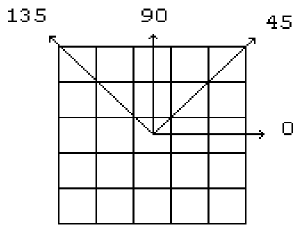


Fig. 2. Directions for co-occurrence matrix.

In our derivation, we used the following definition for the co-occurrence matrix in distance d and direction θ , $P(i,j;d,\theta)$ as in Equation 1.

$$p(i,j;d,\theta) = \#\{((x,y),(x',y') \in DxD | d = \|(x,y),(x',y')\|, \theta = \angle((x,y),(x',y'), I(x,y) = i, I \qquad \mathbf{(1)}$$

where, $P(i,j;d,\theta)$ is the co-occurrence matrix, # stands for the function “number of”, (x,y) and (x',y') are valid image pixel coordinates, D is discrete gray scale image domain, d is the distance between two pixels, θ is the direction of two pixels.

In *Equation 1* we obtained the features of textured images. However, two images with the same texture, but different in size may have different feature vectors. To accomplish this, matrices need to be normalized by the size of images as in Equation 2.

$$P(i,j;d,\theta) = \frac{\#\{((x,y),(x',y') \in D \times D \,|d=\|(x,y),(x',y')\|, \theta = \angle((x,y),(x',y'),I(x,y) = i,I(x',y') = j\}}{\#\{((x,y),(x',y') \in D \times D \,|\, d=\|(x,y),(x',y')\|, \theta = \angle((x,y),(x',y'))\}} \qquad \mathbf{(2)}$$

In order to use the information contained in the gray level co-occurrence matrices, Haralick defined 14 statistical measures, which is based on textural characteristics like homogeneity, contrast, organized structure, complexity and nature of gray level transitions [4]. However, many authors used only one of the characteristics; variance

[1]. We have used only four of GLCM features shown in Table 1. Other features can be found from Haralick's study [4].

Table 1. Four features computed from co-occurrence matrixes.

Features	Equations	Features	Equations
Entropy	$-\sum_{i,j} P(i,j) \log P(i,j)$	Homogeneity	$\sum_{i,j} \frac{p(i,j)}{1+\lvert i-j \rvert}$
Contrast	$\sum_{i,j} \lvert i-j \rvert^{k} P^{\ell}(i,j)$	Variance	$\sum_{i,j} P(i,j).(i-j)^{2}$

Once the texture features are extracted and stored in a database, a similarity function is required to compare texture for similarities. In our study, we have used Euclidean distance metric as similarity function.

3 Clustering Texture Features

The Cluster analysis is a partitioning of data into meaningful subgroups (clusters), when the number of subgroups and other information about their composition or representatives are unknown. A general information for clustering can be found in [10] and [3].

Cluster analysis does not use category labels that tag objects with prior identifiers. In other words, we don’t have prior information about cluster seeds or representatives. The absence of category labels distinguishes cluster analysis from discriminant analysis (and classification and decision analysis). The objective of cluster analysis is simply to find a convenient and valid organization (i.e. group) of the data.

There are many application areas of cluster analysis such as image segmentation, image indexing or, in general, object indexing and to allow users to navigate over the images. The main purpose of clustering is to reduce the size and complexity of the data set. Data reduction is accomplished by replacing the coordinates of each point in a cluster with the coordinates of that cluster’s reference point (cluster’s seed or representative). Clustered data require considerably less storage space and can be manipulated more quickly than the original data. The value of a particular clustering method depends on how closely the reference points represent the data as well as how fast the program runs.

As mentioned before, a clustering schema may represent simply a convenient method for organizing a large set of data so that the retrieval of information may be made more efficiently. Cluster representatives may provide a very convenient summary of the database. In another say, it forms a narrowing down phase of the whole search space.

3.1 Data Types and Data Scales

Clustering algorithms group objects, or data items based on indices of proximity (similarity) between pairs of objects.

Pattern Matrix: If each object in a set of *n* objects is represented by a set of *d* measurements each object is represented by a pattern, or *d*-dimensional vector. The set itself is viewed as $n \times d$ pattern matrix. Each row of this matrix defines a pattern and each column denotes a feature or measurement.

Proximity Matrix: A proximity matrix [$d(i,j)$] accumulates the pair-wise indices of proximity in a matrix in which each row and column represents a pattern. In proximity matrix, d_{ij} denotes the similarity/dissimilarity between object *i* and *j*. Note the matrix is always symmetric.

Similarity and dissimilarity can be summarized as follows:

(*a*) For a dissimilarity: $d(i,i) = 0$, for all *i*,
(*b*) For a similarity: $d(i,i) = max_k d(i,k)$, for all *k*.

3.2 Group similarities

In cluster analysis it is some times convenient to use the distance measurement between groups instead of distance of objects. One obvious method for constructing distance measure between groups is to substitute group mean for the *d* variables in the formula for inter individual measures such as Euclidean distance or other distance metrics. If, for example, group *A* has a mean vector $\overline{X}_A = \left[\overline{x}_{A1}, \overline{x}_{A2}, ..., \overline{x}_{Ad}\right]$ and group *B* has a mean vector, $\overline{X}_B = \left[\overline{x}_{B1}, \overline{x}_{B2}, ..., \overline{x}_{Bd}\right]$, then one measure of the distance between the two group would be as in Equation 3.

$$d_{AB} = \sqrt{\sum_{i=1}^{d} \left(\overline{x}_{Ai} - \overline{x}_{Bi}\right)^2} \qquad \textbf{(3)}$$

3.3 Hierarchical Clustering

A hierarchical classification is a nested sequence of partitions. In this study we have used exclusive (each object belongs to exactly one group) and agglomerative classification. Agglomerative classification places each object in its own cluster and merges these atomic clusters in to larger and larger clusters. The algorithms start with a set of object and merge them to form the clusters and, ends when there are no object to merge with any cluster. In this study, we modified the algorithms to stop when the desired number of clusters reached.

Single linkage is one of the most popular methods, which is used for agglomerative clustering, and it is also known as *nearest neighbor* technique. The characteristic of the method is that distance between groups is defined as the closest pair of objects. For example distance between a cluster with two objects and an object can be defined as follows:

$$d_{(ij)k} = min\ [d_{ik}, d_{jk}]$$

The algorithm starts with searching the proximity matrix and finds the smallest entry. Smallest entry means the most similar textures they can form a cluster. Once they are merged they are considered as a single object. The algorithm works until all the objects are in the same cluster or, the desired number of clusters is reached.

There are different clustering methods, which can be defined according to measurement of distances between clusters, such as centroid clustering, complete-link, group average clustering and single-link clustering as described by [10]. We used group average method in this study. In addition to clustering, we have selected a representative for each cluster to help user to form texture queries. Representatives are selected by the closest object to the average feature vector of each cluster.

4 Experimentation Results

In this section cluster based texture query experiments are presented. We used the well-known methods mentioned in the previous sections. We have implemented an application to observe the results of our study. The purposes of implementing this application is both to provide an example for content-based texture query systems and monitor the effectiveness of a cluster based texture query systems.

4.1 Image Test Bed

There are many texture sets for the purpose of evaluating the textured image retrieval methods. We have used three of them; Brodatz Texture Set, Ohan & Dubes Texture Set and VisTex Texture Set. In our experiments, we have used 14 samples from Brodatz album as a test bed of homogeneously textured images as in some other researches did [7,8]. Each of the 14 sample images has been divided into 25 partly overlapping sub-images of size 170×170. We have selected 5 sub-images randomly from each image and prepared totally 70 images. Our expectation was to obtain 14 clusters containing 5 textures, when desired number of clusters is 14. We are also expecting each sub-texture, which was extracted from the same texture, to be in the same cluster, since they are similar.

4.2 Experimentation Results

We have made a series of experiments to compare their performance on our test bed. In our experiments, first we have tested homogeneity, variance, entropy and contrast features of textures. The results obtained from experiments are summarized in Figure-3.

It is clear that the retrieval performance of entropy is the worst among others. However it is hard to say which one is the best. It depends on the texture domain. For instance, precision value of homogeneity for query 7 is better than variance's value. But variance gives better result than homogeneity for query 4.

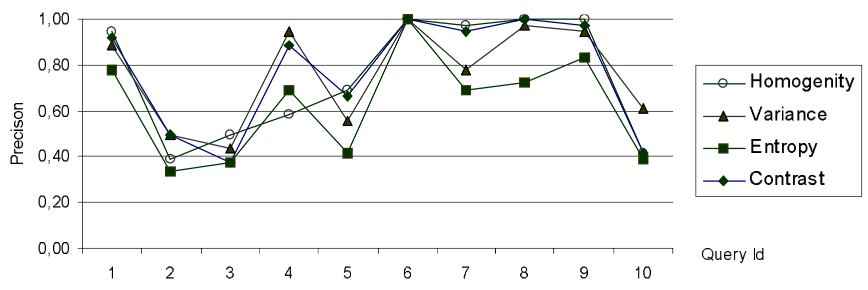


Fig. 3. Precison graph for all features on Texture Query Experiments.

Recall and *precision* are popular measures for retrieval evaluation. Recall signifies the proportion of relevant images in the entire database that are *retrieved* in the query. In other words Recall is the ability to retrieve *relevant* textures. Precision is the ability to reject *nonrelevant* textures. A good system should have high precision and recall values as in [6].

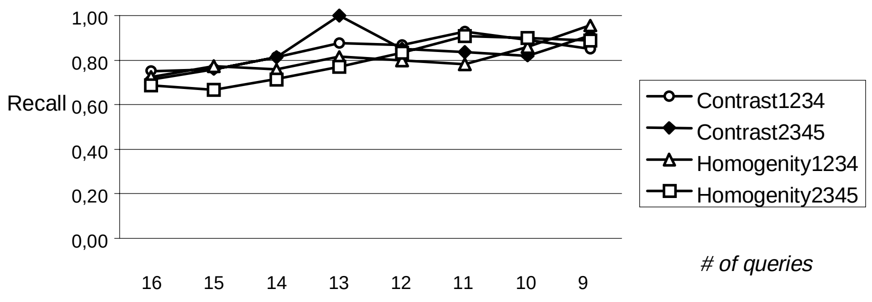


Fig. 4. Recall graph of two feature types and their models

The *X*-axis of the Figure 4 and Figure 5 shows the number of clusters. We used two models; 1234 which means distances, *d*, 1,2,3 and 4 are used to extract features and 2345 means distance 2,3,4 and 5 are used for feature extraction. We did 7 experiments for each feature type to observe the effect of number of clusters in query. We were expecting the performance to be the best, when there are 14 clusters (as explained in previous sections). We could yield two results from these experiments, one is which feature type is better on clustering and other is how many clusters gives the best performance. From Figure 5 we can conclude that the more cluster means the higher value in precision. In another word, ability to reject non-relevant texture increases when we set more clusters. As can be seen from the Figure 5, contrast with model 1234 gives the best performance on precision. The ability to retrieve relevant textures can be measured by the values of recall. As can be seen from the Figure-6, it is best with contrast in model 1234.

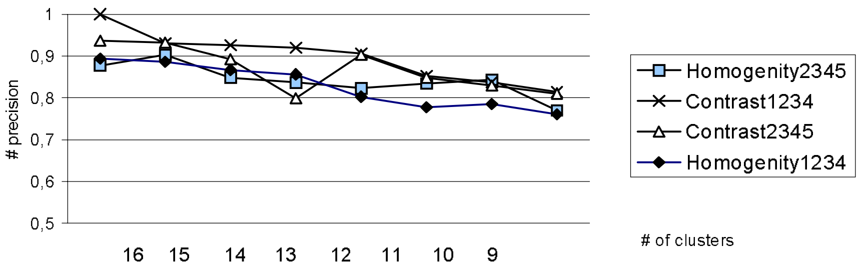
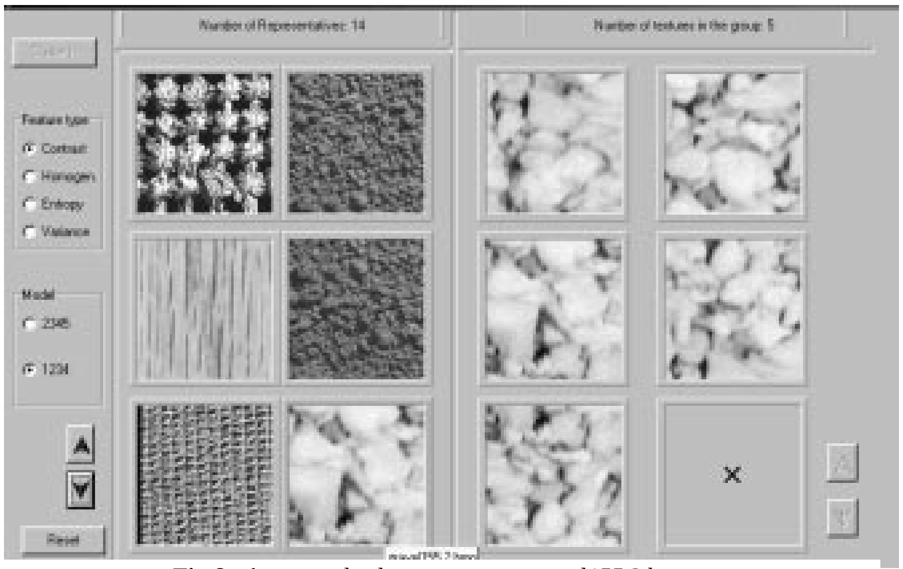


Fig. 5. Precision graph of two feature types and their models

In this study, an application has been developed to see the effects of cluster-based texture retrieval. A typical user interface screen of the application can be seen in Figure-6. In the left side of the screen-shot the representatives of each cluster are shown and, one of the image has been clicked as a query. The query results returned for the query can be seen in right side of Figure-6.



Number of Representatives: 14

Number of textures in the group: 5

Feature type: [✓ Contrast] Homogen. Entropy Variance

Model: 2345 [✓ 1234]

Fig 6 : An example cluster query on gravel155.2.bmp**.**

5 Conclusion and future works

In this study, we have developed a system that allows user to input a textured image and retrieves textures from a database similar to the query. We proposed to cluster textures into groups according to their similarities to make the query expressions easier. Hierarchical clustering techniques have been used for grouping textures in to clusters and selected a representative texture for each group to make the query expressions easier.

We have developed an application both to show its effectiveness and to evaluate the system performance by precision and recall measures. Experimentation has been shown that the results are promising and the cluster-based texture retrieval is acceptable for content based image retrieval.

In this study, we have worked on only single textured images. We are planning to extend our study to the domain independent multi-textured images and test its scalability for more images in future.

References

1. Aksoy, S. & Haralick, R. M. (1998). Content Based Image Database Retrieval Using Variance of Gray Level Spatial Dependencies. IAPR International Workshop on Multimedia Information Analysis & Retrieval (MINAR'98), Hong Kong.
2. Andrey P. & Tarroux P. (August 1995). Unsupervised segmentation of Markov random field modeled textured images using selection relaxation. Technical Report: BioInfo-95-03.
3. Everitt, B. S. (1980). Cluster Analysis. John Wiley & Sons.
4. Haralick, R.M. (May 1979). Statistical and structural approaches to texture. Proceedings of the IEEE, 67(5): 786-804.
5. Manjunath B. S. & Ma W. Y. (August 1996). Texture features for browsing and retrieval of image data. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 18, No:8, pp. 837-842
6. Ozkarahan, E. (1986). Database Machines and Database Management. Prentice Hall, New Jersey.
7. Puzicha, J. & Hofmann, T. & Buhman, J.(1998). Histogram Clustering for Unsupervised Segmentation and Image Retrieval.
8. Smith J. R., & Chang S.F. (May 1996). Automated binary texture feature sets for image retrieval. Proceedings of the IEEE. ICASSP-96. Atlanta, GA.
9. Smith, J. R. & Chang, S. F. (1994). Transform features for texture classification and discrimination in large image databases. Proceedings of IEEE International Conference on Image Processing (ICIP-94), Austin, Texas.
10. Jain, A. K. & Dubes, R. C. (1988). Algorithms for Clustering Data, Prentice Hall.