A Clustering Approach for Large Visual Databases*

Gholamhosein Sheikholeslami and Aidong Zhang
Department of Computer Science
State University of New York at Buffalo
Buffalo, NY 14260

Abstract

The representation and organization of images in a large-scale image database is crucial to support effective and efficient accesses to image data on the basis of content. In this process, significant features must first be extracted from image data in their pixel format. These features must then be classified and indexed to assist efficient retrieval of image content. In this paper, we investigate effective image data representation and organization approaches for large-scale image databases. An effective block-oriented image decomposition structure is used as a fundamental data model for representing image content. A clustering mechanism is then proposed to categorize images based on feature similarity. Efficient image retrieval is supported. Experimental analysis are conducted and presented to demonstrate the effectiveness and efficiency of the approaches.

1 Introduction

Large-scale image database systems require effective and efficient ways of indexing and accessing image data based on content. In this process, significant features must first be extracted from image data in their pixel format. These features must then be classified and indexed. With the large volume of data stored in an image database, data classification is an important step to achieve efficient indexing and retrieval. Indexing and accessing alpha-numerical data in traditional databases have been known as a well-understood problem. In these databases, categorization of data is straightforward on the basis of their semantics. Efficient data access can be facilitated by B-trees and other data structures. However, such traditional approaches to indexing may not be appropriate in the context of content-based image retrieval [1, 4, 18, 12, 5, 7, 2, 19]. In this context, a challenging problem arises with many image databases, within which queries are posed via visual or pictorial examples (termed *visual queries*). A typical visual query might entail the location of all images in a database that contain a subimage similar to a given query image. Such a query could, for example, be used in a remotely sensed satellite image setting to find a specific type of land, such as forest or barren, in a remotely sensed image.

Approaches which have drawn much recent attention involve the extraction of various features of images using various image processing techniques. These texture characteristics are then used to formulate the feature vectors of images which can be utilized to support content-based image retrieval. Thus, an image database can be assumed to consist of feature vectors. Substantial research has been directed toward the support of efficient indexing techniques based on feature

*This research is supported by Xerox Corporation.

vectors of images [2, 14, 17, 15, 3, 9]. Given feature vectors of images, indexing algorithms such as R^*-tree, R^+-tree, and TV-tree [15, 3, 9] have been proposed to support efficient accesses to image databases.

In a large-scale image database, a critical problem to be solved is to classify feature vectors into different semantic clusters before indexing them within each cluster. The goal of clustering is to reduce the search of irrelavant images, while indexing tries to narrow down the search to the relevant images. A more effective clustering approach is needed to be integrated into indexing techniques for efficient retrieval of image data. Once the query is narrowed to a specific category, image retrieval can then proceed efficiently. Images in a database can thus be categorized in clusters on the basis of their similarity to a set of iconic images. These clusters can then be further classified into subclusters based on more specific icons. At the lowest level, the feature vectors of subclusters are indexed using a variation of R-tree structure. Thus, a hierarchical clustering and indexing structure is designed for image databases to support efficient visual retrieval. The incorporation of clustering into indexing techniques intends to provide an effective method to classify image data based on their similarity features. In a large geographical database, such a classification is not intended to be very accurate but to provide a good approximate categorization of image data based on a given similarity measurement.

We have conducted experimental research on image data retrieval based on image compression techniques [20, 22, 21]. A critical problem on indexing has been encountered. In most existing content-based retrieval approaches, feature vectors of images are first constructed (for example, from wavelet transforms), which are then utilized to distinguish images through measuring distances between feature vectors. Our experimental results demonstrate that this type of approaches can be effectively used to perform content-based similarity comparison between images. A more effective clustering approach is needed to be integrated into indexing techniques for efficient retrieval of image data. Once the query is narrowed to a specific category, image retrieval can then proceed efficiently.

In this paper, we investigate effective image data representation and organization approaches for large-scale image databases. An effective block-oriented image decomposition structure is used as a fundamental data model for representing image content. A clustering mechanism is then proposed to categorize images based on feature similarity. Efficient image retrieval is supported. Experimental analysis are conducted and presented to demonstrate the effectiveness and efficiency of the approaches. Images in a database are categorized in clusters on the basis of their similarity to a set of iconic images. After training the system, the process of clustering operates automatically, with no human assistance required to determine the points of similarity between images and icons.

The rest of the paper is organized as follows. Section 2 introduces the image organization and representation in large-scale visual information systems. In section 3, we present a novel clustering technique that can be used to support effective image classification. In Section 4, experimental analysis of the proposed approach will be offered. Concluding remarks are offered in Section 5.

2 Image Organization and Representation

To support efficient content-based retrieval of images in a large-scale image database system, image data must be first well-organized and represented in the system. Following recent research in image content-based search and retrieval, we assume that images are represented by their feature vectors in the database. Thus, an image database can be viewed as a feature space that contains all generated feature vectors of images. The comparison of similarity between images is performed by first calculating the distance between their feature vectors and comparing the distance with a distance threshold. The similarity between two feature vectors is then obtained by translating the distance to a number in the range of $[0, 1]$, with 0 representing minimum and 1 representing maximum similarity.

2.1 Image Feature Extraction

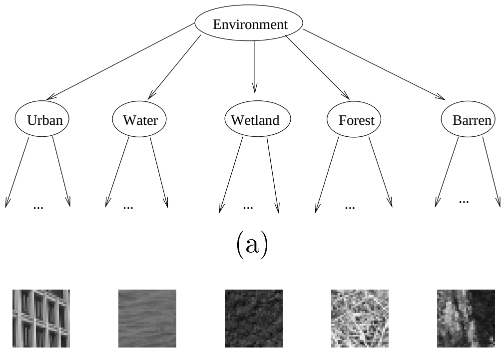
Several methods of image feature extraction are available in literature. These include methods for extracting information about edges, curves, color, texture, etc. In particular, we have developed algorithms for supporting image texture content queries using textural energy features derived from low and high pass spatial frequency bands [13] using a multi-level wavelet-based subband decomposition[11]. The computational framework is intended to provide an open architecture for allowing different feature extraction schemes (possibly developed by other researchers) including methods for handling new types of queries.

2.2 Image Decomposition

In content-base querying of images (or video frames), often the answer to a query lies in a small region of the image (small segment of the video). Performing detailed segmentation into object-containing regions to enable subsequent query matching is not only computationally expensive but almost impossible to achieve. An alternative is to decompose the image into meaningful groups that are likely to contain single objects using grouping methods developed in computer vision[8, 10]. Grouping methods, however, avoid generating all possible groups to keep computational costs low, thus leading to false negatives in query localization. We have explored an alternative scheme of image decomposition using a hierarchical overlapping window structure called the nona-tree[13] where the image region spanned by each level in the hierarchy is divided into nine overlapping segments. Using this nona-tree decomposition, we have shown that for any specified query size, the window size and the amount of overlap between adjacent windows can be derived that guarantees the localization of queried object in one such region of the decomposition[13].

2.3 Image Organization

Organizing images in the database by clustering in a feature space derived from images has been a popular technique used in the past. Sine such methods do not incorporate spatial location information, they can give false matches during indexing. Images in a database can be categorized in clusters on the basis of their similarity to a set of iconic images. These clusters can then be further classified into subclusters based on more specific icons. At the lowest level, the feature vectors of subclusters can be indexed using a variation of R-tree structure [3, 15]. Thus, a hierarchical clustering and indexing structure can be designed for media databases to support efficient visual retrieval. Applying this approach to remotely sensed images, a set of images in an environmental image database may be classified into the hierarchy such as the one shown in Figure 1 (a). For these first-level five clusters, a selection of the icons for each cluster, respectively, could be as shown in Figure 1 (b).



(a)

(b)

Figure 1: (a) A hierarchical clustering of an environment image setting. (b) Clustering icons.

3 A framework of feature-based image clustering and retrieval

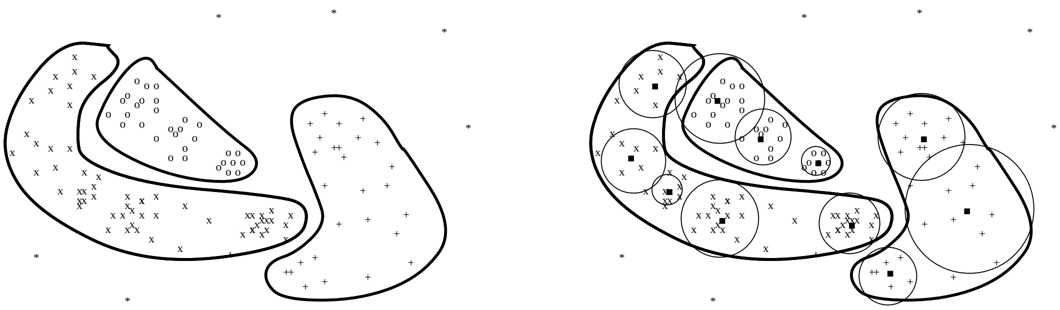
In this section, we will present a feature-based image clustering and retrieval approach. We first describe how cluster icons are chosen. Given clustering icons and features, images in a database can be classified based on their similarity to the cluster icons. Based upon the resulting clusters, image retrieval can be efficiently supported.

3.1 Cluster icons and features

To support efficient content-based image retrieval, we propose to first cluster database images based on their similarity to predefined image icons, which are termed *cluster icons*. To retrieve images containing a query icon, the query icon will first be compared to the the cluster icons. Only database images in the clusters corresponding to the chosen cluster icons will be searched. An effective clustering of database images can greatly reduce the number of images to be searched, resulting in a more efficient retrieval. Indexing techniques can be incorporated within each cluster to provide a faster retrieval. Clusters are application dependent and are determined by the application experts.

For example, in a geographical image database, we can define the clusters of residential, agriculture, water, grass, and a special cluster *other*. The cluster *other* contains all images which cannot be included in any of the residential, agriculture, water, or grass clusters. We want to represent clusters by cluster icons to be compared to the database images. Since in the feature space, clusters can have arbitrary shape, it is not possible to represent each cluster by only one cluster icon. So, we need a method to find multiple cluster icons representing each cluster. Figure 2-a shows an example in two dimensional space where clusters have arbitrary shapes. One approach to find multiple cluster icons could be to determine the subclusters within each cluster and use the centroids of those subclusters as the cluster icons of the cluster. Figure 2-b shows such centroids within each cluster. Cluster *other* should include all the objects in Figure 2-a shown by "$_*$".

We used a hierarchical method to find the subclusters within each cluster at different levels. For each cluster we used a training data set to determine the subclusters. Suppose training set of a cluster C_c has n objects. Initially, we assume there are n subclusters corresponding to n objects in the cluster. We compute the distances between all pairs of subclusters. We then group together the two closest subclusters, resulting in $n-1$ subclusters. We repeat the process of computing the distances between all pairs of current subclusters, and grouping the two closest ones until having

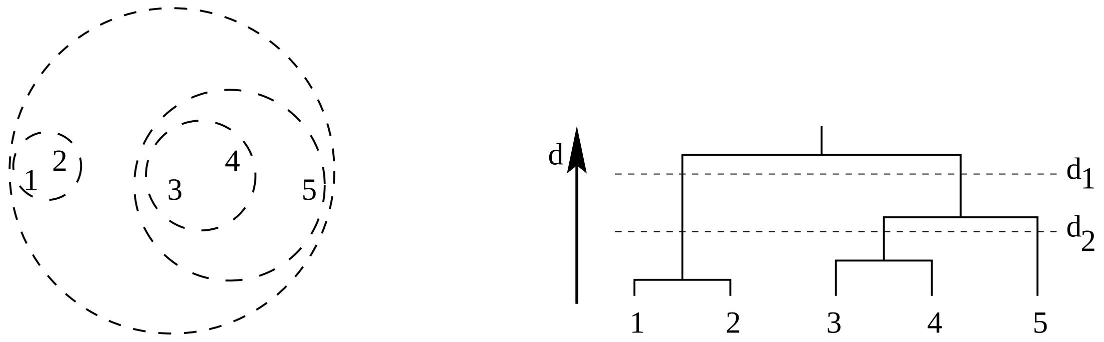


(a) (b)

Figure 2: a) arbitrary shaped clusters. b) circular subclusters of clusters.

only one subcluster.

This hierarchically nested set of subclusters can be represented by a tree-diagram, or a *dendrogram* [6]. Figure 3 shows a group of 5 objects and their corresponding dendrogram. Each of n leaves of the tree represents the single object whose label appears below it. Each position up the tree at which branches join, has an associated numerical value d, which determines the distance between the two grouped subclusters. The smaller the value of d, the more similar two subclusters will be. We considered the distance between the centroids of subclusters as the distance between subclusters. We can cut the dendrogram of each cluster at different levels to obtain different number of subclusters (cluster icons). For example, if we cut the dendrogram in Figure 3-b, where the distance between subclusters is d_1, we will have two subclusters, whereas if we cut at level d_2, we will have three subclusters. let $S = \{o_1, o_2, \ldots, o_n\}$ be the subcluster containing objects $o_1, o_2, \ldots, o_n$. If we cut a dendrogram at the level in which the distance between the last two grouped subclusters is d_i, then the maximum distance between the children of the remaining subclusters will be d_i. For example, in Figure 3-b, if we cut the dendrogram at level d_1, distance between subclusters {1} and {2}, or {3} and {4}, or {3,4} and {5}, will be less than d_1.



(a) (b)

Figure 3: A group of objects and the corresponding dendrogram.

In the dendrogram of each cluster, we store the mean and variance of the distances of subcluster's objects, to the centroid of the subcluster. This information helps us know about the distribution of objects in each subcluster. We cut the dendrogram of clusters at different levels to have sufficient number of cluster icons per cluster. Using our method in a two dimensional feature space, subclusters can be represented by a circle, where center of circle is the centroid of the subcluster and the radius of circle is dependent to the distribution of objects in the subcluster.

It should be noted that, using our clustering approach, subclusters can have arbitrary shapes based on neighboring subclusters. Figure 2-b shows some possible subclusters of Figure 2-a. The black square-d dots in Figure 2-b represent the cluster icons. As Figure 2-b shows, subclusters of the same cluster (or even different clusters) may overlap. Also, there might be some objects in the clusters that do not belong to any subclusters. Next section describes how our clustering approach processes such cases.

Assume a cluster C_c consists of m subclusters $S_{c1}, S_{c2}, \ldots, S_{cm}$. In clustering database images, whenever a subcluster S_{ci} is chosen, the database images will be assigned to the corresponding cluster C_c. Similarly, in image retrieval, if a subcluster S_{ci} is chosen, only the corresponding cluster C_c will be searched.

During the clustering, the features of database images and clustering icons are used to determine the similarity between the database images and cluster icons. These features are termed *cluster features*. Thus, each database image or cluster icon will contain a *cluster feature vector* which represents the cluster features, denoted $CV(a_1, ..., a_l)$.

3.2 Clustering

To cluster database images, we first compute the distance between the cluster feature vectors of cluster icons and all database images. Using multi-resolution property of wavelet transform, we extract the image features at different scales. In comparing cluster feature vectors, only *coarse* scale features will be used; while in retrieving images *all* the features will be compared. Let $CV_1(a_1, ..., a_n)$ and $CV_2(b_1, ..., b_n)$ be two cluster feature vectors. We use the root mean square metric to compare the distance between the two vectors, as follows:

$$dist(CV_1, CV_2) = \sqrt{\frac{\sum_{i=1}^{n}(a_i - b_i)^2}{n}}. \tag{1}$$

If the distance between the database image and the closest cluster icon is less than a threshold T, it will be assigned to the corresponding cluster represented by the chosen cluster icon. If there is no cluster icon for which the distance between the database image and the cluster icon is less than T, the database image will be assigned to the special cluster *other*. The cluster *other* contains all the database images that cannot be assigned to the predefined clusters.

In determining the correct cluster(s) for a database image, we use the distribution of objects in each subcluster. For each subcluster we define a *scope*. Let $Dist\_Mean_i$ and $Dist\_Var_i$ be the mean and variance of distances of objects to the centroid of the subcluster i. The scope of subcluster i is part of the feature space whose distance to the centroid of subcluster is less than $Dist\_Mean_i + \beta * Dist\_Var_i$. Parametrer β is a factor that is determined by experiments. In a two dimensional space, the scope of subcluster is a circle with radius $Dist\_Mean_i + \beta * Dist\_Var_i$. In our clustering approach, whenever a database image falls within scope of any subcluster, it will be assigned to the cluster corresponding to the subcluster. The reason is that a database image may contain the features of several different cluster icons and it should be assigned to all the corresponding clusters.

As an example, Figure 4 shows the scope of 3 subclusters A, B, and C in a two dimensional space. In general, these subclusters do not necessarily belong to different clusters. The object denoted by point 1 falls in the scope of subclusters A and B and will be assigned to both.

However, some objects may not fall in any of the scopes considered above. For example, Point 2 in Figure 4 is not in the scope of any of subclusters. Our clustering approach assigns such an object

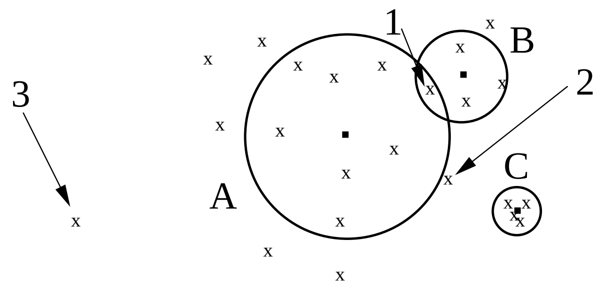


Figure 4: Three sample subclusters.

to the closest subcluster. Although the distance between point 2 and the centroid of subcluster C is less than that of subclusters A and B, but based on distribution of objects within clusters, it is more logical to assign object 2 to subcluster A. So, we first compute the distance between the object and the border of the subclusters's scope, and then choose the closest cluster. Hence, object 2 will be assigned to subcluster A. The object shown as point 3 in Figure 4 will be assigned to cluster *other*, because it is far away from all clusters and its distance to the closest cluster is greater than threshold T.

Algorithm 3.2 shows the main steps in choosing the cluster(s) for database images.

3.3 Image retrieval

Within each cluster, the feature vectors are then indexed. A novel indexing technique can be devised on the basis of proposed variations of the R-tree [15, 3, 9]. These indexing techniques, which include the R^*-tree and TV-tree, can be used to classify the feature vectors of images into clusters on the basis of their locations within the feature space. An efficient retrieval for a given query icon can then be conducted by searching through the R^*-tree or TV-tree. High dimensionality must be handled in the defined feature vectors. In addition, some dimensions in the feature vector are more important to be used to index than others. Using the TV-tree indexing, in each cluster, feature vectors are clustered into leaf nodes of the indexing tree and the description of their *minimum bounding region (MBR)* is stored in the parent node. In case an MBR contains too many elements, additional features will be used to split the MBR. Parent nodes are then recursively grouped until the root is constructed. Each parent node contains an MBR which covers all descendants of that branch. We assume that each node (may not include root) contains between m and M ($m < M$) children or data items. To conduct the similarity search in the indexing tree for a given query icon, the search process starts from the root and examines each branch that overlaps with a given search region until the leaf MBRs are selected.

To retrieve database images containing the query icon, the distance between query icon and cluster icons will first be computed. If the distance between query icon and the closest cluster icon is greater than threshold T, the cluster *other* will be searched. Otherwise, the system finds all the subclusters whose scope include the query icon, and then searches the corresponding cluster(s). If query icon is not in the scope of any subclusters, the cluster corresponding to the closest subcluster will be searched.

To compute the distance between a database image and a query icon (in retrieving) or a cluster icon (in clustering), we should first decompose the database image into subsegments. Then the system extracts the features of all the subsegments of image and compares them to that of icons. We used nona-tree decomposition [16] to segment database images into subsegments. We consider

Algorithm 1
$Choose\_Cluster\_To\_Add(Image)$
/* Based on distance between $Image$ and subclusters, adds $Image$ to the corresponding cluster(s). */

```
{
    /* compute the distance between database image and all the subclusters */
    for i = 0 to Number_Of_Subclusters
        { Distance_i = Compute_Distance(Image, Subcluster_i);
          Adjusted_Distance_i = Distance_i − (Dist_Mean_i + β ∗ Dist_Var_i);
                                      /* distance to the border of subcluster's scope */
        }
    Closest_Cluster = Find_Min(Adjusted_Distance); /* find the index of the closest subcluster */

    if (Adjusted_Distance_Closest_Cluster > T)
        {Add_Image_To_Cluster(other); /* add to special cluster other */
          return;
        }

    Cluster_Added = FALSE;
    /* add image to "all" clusters that are close enough */
    for i = 0 to Number_Of_Subclusters
        {if (Distance_i ≤ Dist_Mean_i + β ∗ Dist_Var_i)
            {Add_Image_To_Cluster(i); /* add to the corresponding cluster of subcluster i */
              Cluster_Added = TRUE;
            }
        }

    if (Cluster_Added == FALSE) /* image not added to any cluster */
        Add_Image_To_Cluster(Closest_Cluster); /* add to the closest cluster */
}
```

the distance between a database image and an icon as the minimum of the distance between its subsegments and the icon.

4 Experimental analysis

In this section, we will present the experimental analysis on the effectiveness of the clustering approach and the impact of the clustering on the efficiency of image retrieval. We first briefly explain the extracted features from images.

4.1 Feature vectors

We experimented the proposed approach with four different types of feature sets. More details about these feature sets can be found in the paper [?]. We used Haar wavelet transform to extract textural features of images. In feature set 1, we applied wavelet transform 3 times, giving 10 sub-bands. For each sub-band, we used the mean and variance of absolute value of wavelet coefficients as the features of that sub-band. So the feature vector for each sub-band is $< mean, variance >$. Each subsegment of image will have totally 20 features. To compute the distance between feature vectors in clustering we only consider the features of the third and second scales of wavelet transform (7 sub-bands, 14 features).

To obtain the features in feature set 2, we apply wavelet transform twice. For the sub-band LL of the coarsest scale of wavelet transform, we consider mean and variance of absolute value of wavelet coefficients as its features $< mean, variance >$. For the other sub-bands, in addition to mean and variance, the system counts number of edge pixels in the sub-band. So the feature vector for each of these sub-bands is $< mean, variance, number\ of\ edge\ pixels >$. In this feature set we will have 7 sub-bands with 20 $(2 + 6 \times 3)$ features. That is, in clustering, we just consider the features of the sub-bands of the coarsest scale of wavelet transform (11 features) to compute the distance.

In feature set 3, the wavelet transform is applied twice. For the sub-band LL of the coarsest scale of the wavelet transform, we compute the mean and variance of the absolute value of wavelet coefficients as its features. Similar to feature set 2, we count the number of edge pixels in the other sub-bands. But we first divide each sub-band into 9 equal non-overlapping regions and count the number of edge pixels in each region separately. Thus, the feature vector for each of these sub-bands will be

$$< mean, variance, number\ of\ edge\ pixels_1, number\ of\ edge\ pixels_2, \ldots, number\ of\ edge\ pixels_9 >,$$

where $number\ of\ edge\ pixels_i$ is the number of edge pixels in the region i. Knowing the number of edge pixels in the 9 regions gives us more information about the edges and their relative positions. In feature set 3, there will be 7 sub-bands with 68 $(2 + 6 \times 11)$ features. In other words, we only consider the features of the coarsest scale of wavelet transform (35 features) to compute the distance in clustering.

Feature set 4 is similar to feature set 1. In this feature set, the wavelet transform is applied twice giving 7 sub-bands. The feature vector for each sub-band is $< mean, variance >$. Each subsegment of image will have 14 features. To compute the distance between feature vectors in clustering, we only consider the features of the second scale of wavelet transform (4 sub-bands, 8 features).

4.2 Effectiveness of Clustering and Retrieval

As test data we used 405 air photo images of size 128 $\times$ 128 pixels. To find the cluster icons representing each of residential, water, grass, and agriculture clusters, we used 50 training image data of size 32 $\times$ 32 pixels, per cluster. Figure 5 shows the sample cluster icons found by our hierarchical approach. Figures 6 shows the sample images in the four clusters.

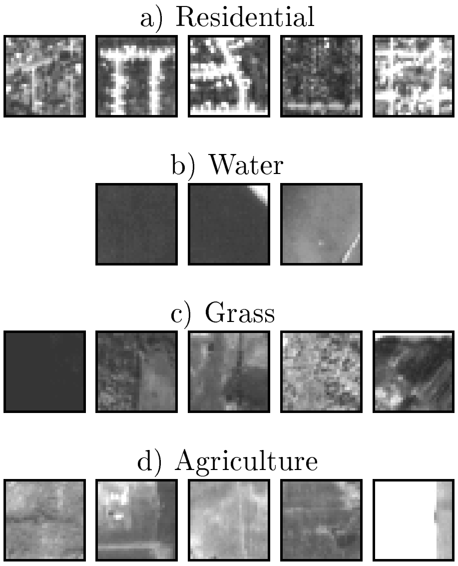


Figure 5: Sample cluster icons.

To evaluate the performance of our clustering approach, we first manually assigned all database images to their corresponding clusters. We consider these clusters as ideal clusters. We then compared the result of our clustering approach to the ideal clusters. Let C_c be the set of images in the system's output for cluster c. Let I_c be the set of images in the ideal cluster for cluster c. We define P_c, the precision of clustering for cluster c as

$$P_c = \frac{|C_c \bigcap I_c|}{|C_c|}.$$

Recall of cluster c, R_c, is defined as

$$R_c = \frac{|C_c \bigcap I_c|}{|I_c|}.$$

The higher P_c and R_c be, the better will clustering be.

As mentioned in Section 3.2, we used two parameters in our clustering approach: threshold T to assign a database image (or search) in cluster *other*, and parameter β to determine the scope of each subcluster.

In our experiments we chose 150 as the value of threshold T. We then tried the system for different values of parameter β. Table 1 shows P_c and R_c for the 4 feature sets using Haar wavelet transform.

A good clustering should have high recall and precision. But recall is more important than precision in clustering, because in retrieval from a cluster, we want to miss as few images as possible comparing to retrieval from the whole database.

Cluster c	$\beta = 0.0$		$\beta =$ -0.5		$\beta =$ -1.0		$\beta =$ -1.5	
	R_c	P_c	R_c	P_c	R_c	P_c	R_c	P_c
Water	0.84	0.67	0.72	0.74	0.56	0.86	0.51	0.97
Agriculture	0.98	0.32	0.93	0.33	0.90	0.34	0.80	0.35
Residential	0.98	0.59	0.97	0.61	0.93	0.62	0.83	0.67
Grass	0.97	0.39	0.89	0.43	0.76	0.50	0.51	0.58

a) Feature set 1

Cluster c	$\beta = 0.0$		$\beta =$ -0.5		$\beta =$ -1.0		$\beta =$ -1.5	
	R_c	P_c	R_c	P_c	R_c	P_c	R_c	P_c
Water	0.75	0.82	0.69	0.94	0.47	0.97	0.31	1.00
Agriculture	0.95	0.32	0.93	0.36	0.86	0.40	0.68	0.41
Residential	0.97	0.64	0.91	0.66	0.79	0.72	0.55	0.81
Grass	0.95	0.41	0.88	0.44	0.77	0.49	0.62	0.49

b) Feature set 2

Cluster c	$\beta = 0.0$		$\beta =$ -0.5		$\beta =$ -1.0		$\beta =$ -1.5	
	R_c	P_c	R_c	P_c	R_c	P_c	R_c	P_c
Water	0.79	0.64	0.71	0.83	0.51	0.95	0.35	0.96
Agriculture	0.92	0.33	0.86	0.38	0.79	0.43	0.68	0.45
Residential	0.98	0.62	0.94	0.65	0.85	0.71	0.64	0.80
Grass	0.94	0.40	0.89	0.43	0.78	0.49	0.66	0.55

c) Feature set 3

Cluster c	$\beta = 0.0$		$\beta =$ -0.5		$\beta =$ -1.0		$\beta =$ -1.5	
	R_c	P_c	R_c	P_c	R_c	P_c	R_c	P_c
Water	0.81	0.53	0.72	0.61	0.53	0.72	0.34	0.88
Agriculture	0.92	0.31	0.92	0.34	0.88	0.37	0.81	0.36
Residential	0.99	0.61	0.94	0.65	0.86	0.71	0.62	0.78
Grass	0.95	0.40	0.89	0.43	0.78	0.48	0.60	0.49

d) Feature set 4

Table 1: Precision and recall of clustering for different values of β.

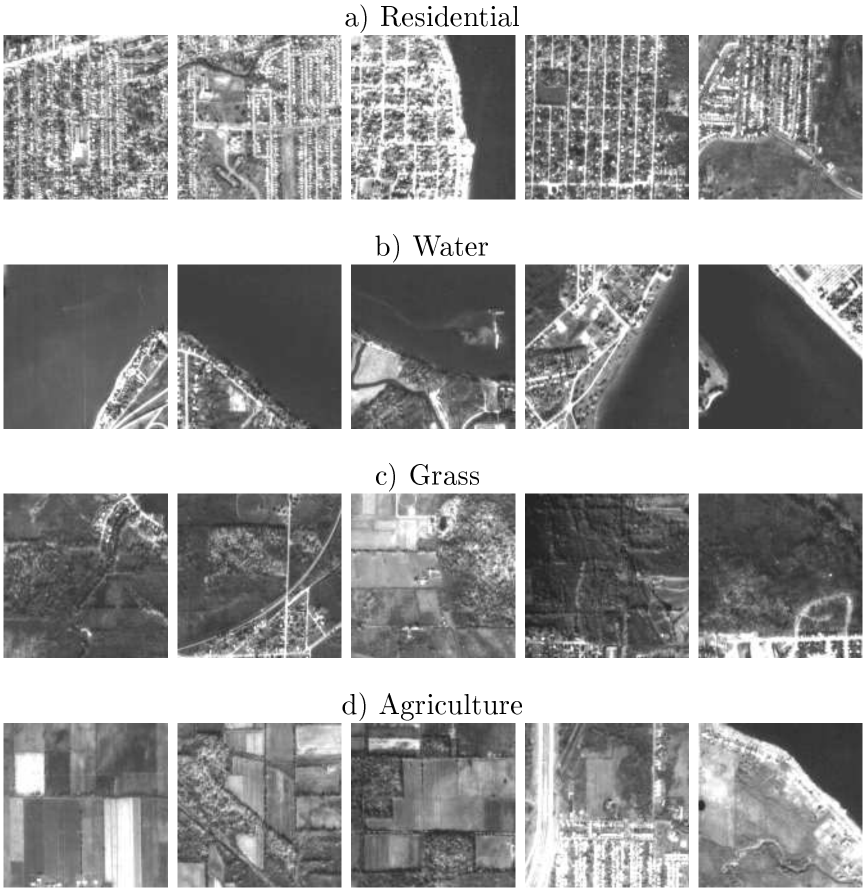


Figure 6: Sample database images in four clusters.

As Table 1 shows, decreasing value of β (and thus decreasing scope of subcluster), results in decreasing recall of clustering and increasing precision of clustering. By choosing a good value for β, we can obtain a reasonable recall and precision of clustering. In our experiments, recall of cluster water is lower than the rest of clusters. The reason could be the distribution of images in that cluster.

To study the impact of clustering on effectiveness of retrieval, we chose 10 query icons of size 32 $\times$ 32 pixels. Our system processes queries such as "retrieve top N (say 50) images that have the same texture as the query icon". We calculated the effectiveness of retrieval in terms of $retrieval\_rate$. Retrieval rate for top n images is defined as

$$retrieval\_rate = \frac{number\ of\ relevant\ images\ in\ top\ n\ images}{n}.$$

We calculated retrieval rates for values 10, 20, 30, 40, and 50 of n for the 10 query icons and then found their average value. We first computed retrieval rate when clustering is not used. We then computed retrieval rate when the clustering approach is used (with value 150 for threshold T and values 0.0, -0.5, -1.0, and -1.5 for β). Table 2 shows the average retrieval rate of top n images (n = 10, 20, 30, 40, and 50) for different feature sets. For feature set 1, retrieval rate is as good as without clustering. For feature sets 2, 3, and 4, retrieval rate of clustering approach is between 2-8% lower than without clustering, depending on value of β. It shows that, for good feature sets, retrieval from a clustered database can be almost as effective as retrieval from the whole database.

β	Feature set 1	Feature set 2	Feature set 3	Feature set 4
0.0	0.65	0.69	0.66	0.68
-0.5	0.64	0.68	0.67	0.67
-1.0	0.60	0.68	0.66	0.65
-1.5	0.58	0.64	0.64	0.62
Without clustering	0.65	0.70	0.71	0.70

Table 2: Average retrieval rate.

β	Feature set 1	Feature set 2	Feature set 3	Feature set 4
0.0	0.84	0.78	0.79	0.80
-0.5	0.79	0.68	0.67	0.69
-1.0	0.63	0.55	0.53	0.56
-1.5	0.51	0.41	0.39	0.45
Without clustering	1.00	1.00	1.00	1.00

Table 3: Average *DB_ratio* (ratio of database search).

As Table 2 shows, when using clustering, feature set 2 has a higher retrieval rate than feature sets 1, 3, and 4.

The other issue that should be considered in comparing the feature sets is the number of images that are searched in retrieval from a clustered database. We define ratio of database search DB_ratio as

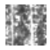
$$DB\_ratio = \frac{number\ of\ images\ searched\ in\ database}{total\ number\ of\ images\ in\ database}.$$

Table 3 shows the average DB_ratio for the 10 query icons. As Tables 2 and 3 show, feature set 1 has the lowest retrieval rate and the highest *DB_ratio* comparing to that of feature sets 2, 3, and 4. It means that larger number of images will be searched but lower retrieval rate is achieved. Feature set 4 has a lower retrieval rate and higher *DB_ratio* than that of feature sets 2 and 3. Performance of feature set 2 is slightly better than performance of feature set 3 in our clustering approach.

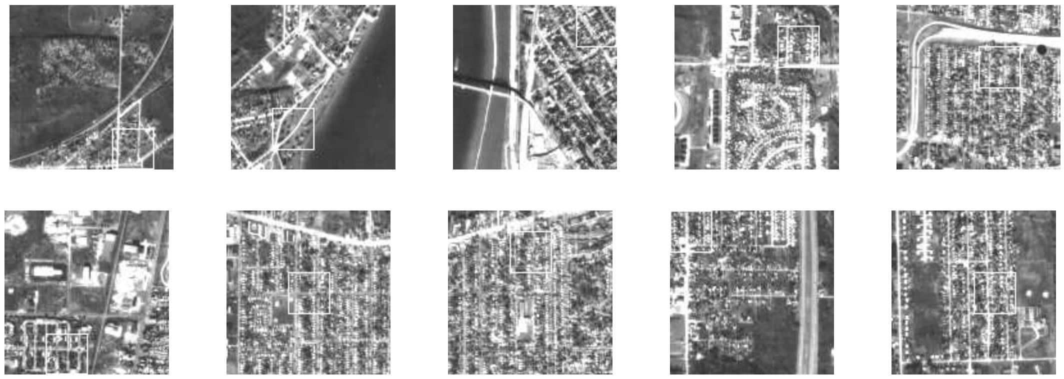
Figure 7 shows the 10 closest retrieved images for a sample query icon, using clustering, and without using clustering. The retrieved section of each image is shown inside the white box. For the query icon in Figure 7, the first 10 retrieved images by clustering and without clustering approaches are the same. This shows that, in general, the retrieval results when using clustering and without clustering are consistent and comparable. Hence, the two methods have comparable effectiveness.

5 CONCLUSION

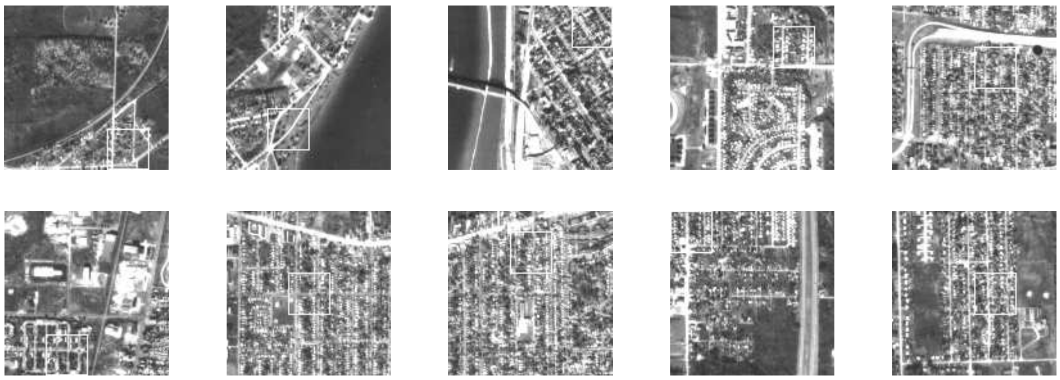
In this paper, we proposed an approach to clustering large visual databases. Database images are clustered based on their similarity to cluster icons. Clusters are application dependent and are



a) query icon



b) without clustering



c) with clustering

Figure 7: Retrieved images a) query icon, b) without clustering, c) with clustering.

determined by application experts. In our approach, cluster icons were found hierarchically using training data sets. We used wavelet transform to extract texture features of images at different scales. Only the coarse scale features of images have been used in clustering. We compared 4 different feature sets and the results are reported in the paper. Our experiments showed that for good feature sets, retrieval from a clustered database can be as effective as retrieval from the whole database, but it is more efficient.

References

[1] Manish Arya, William Cody, Christos Faloutsos, Joel Richardson, and Arthur Toga. QBISM: A Prototype 3-D Medical Image Database System. *IEEE Data Engineering Bulletin*, 16(1):38–42, 1993.

[2] J.R. Bach, S. Paul, and R. Jain. A Visual Information Management System for the Interactive Retrieval of Faces. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):619–628, 1993.

[3] N. Beckmann, H.P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and Robust Access Method for Points and Rectangles. In *Proceedings of ACM-SIGMOD International Conference on Management of Data*, pages 322–331, Atlantic City, NJ, May 1990.

[4] S.K. Chang, C.W. Yan, Donald C. Dimitroff, and Timothy Arndt. An Intelligent Image Database System. *IEEE Transaction on Software Engineering*, 14(5):681–688, May 1988.

[5] T. Gevers and A.W.M. Smeulders. *An Approach to Image Retrieval for Image Databases*, volume 720. Lecture Notes in Computer Science, Springer-Verlag, Berlin.

[6] A. D. Gordon. *Classification.* Chapman and Hall Ltd, New York, 1981.

[7] Tai-Yuan Hou, Peiya Liu, Arding Hsu, and Ming-Yee Chiu. Medical Image Retrieval by Spatial Features. In *IEEE Conference on Systems, Man, and Cybernetics*, 1992.

[8] D.W. Jacobs. Grouping for recognition. Technical report, Artificial Intelligence Lab, M.I.T., AI-Memo-1177, 1989.

[9] K.-I. Lin, H.V. Jagadish, and C. Faloutsos. The TV-Tree: an Index Structure for High-Dimentional Data. *The VLDB Journal*, 3(4):517–542, October 1994.

[10] D. G. Lowe. *Perceptual Organization and Visual Recognition.* Boston: Kluwer Academic, 1985.

[11] S. Mallat. Multiresolution approximation and wavelet orthonormal bases of $l^2(r)$. *Transactions of American Mathematical Society*, 315:69–87, September 1989.

[12] F. Rabitti and P. Savino. Automatic Image Indexation and Retrieval. In *Conference on Intelligent Text and Image Handling*, 1991.

[13] E. Remias, G. Sheikholeslami, Aidong Zhang, and Tanveer F. Syeda-Mahmood. Supporting Content-Based Retrieval in Large Image Database Systems. *The International Journal on Multimedia Tools and Applications*, 1996. (to appear).

[14] M. Flickner H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by Image and Video Content: The QBIC System. *IEEE Computer*, 28(9), 1995.

[15] T. Sellis, N. Roussopoulos, and C. Faloutsos. The R+-tree: A Dynamic Index of Multidimensional Objects. In *Proceedings of the 18th VLDB conference*, 1987.

[16] G. Sheikholeslami, E. Remias, J. Guo, and Aidong Zhang. Image decomposition and representation in large image database systems. *The Journal of Visual Communication and Image Representation.* (scheduled to appear in Vol. 8, No. 2, June 1997).

[17] John R. Smith and Shih-Fu Chang. Quad-Tree Segmentation for Texture-Based Image Query. In *Proceedings of ACM Multimedia 94*, pages 279–286, San Francisco, California, October 1994.

[18] A. Turtur, F. Prampolini, M. Fantini, R. Guarda, and M.A. Imperato. IDB: An Image Database System. *IBM Journal of Research and Development*, 35(1):88–96, January 1991.

[19] Jian Kang Wu and Arcot Desai Narasimhalu. Identifying Faces Using Mutiple Retrievals. *IEEE Multimedia*, 1(2):27–38, 1994.

[20] Aidong Zhang, Biao Cheng, and Raj Acharya. An Approach to Query-by-texture in Image Database Systems. In *Proceedings of the SPIE Conference on Digital Image Storage and Archiving Systems*, pages 338–349, Philadelphia, October 1995.

[21] Aidong Zhang, Biao Cheng, and Raj Acharya. A Fractal-Based Clustering Approach in Large Visual Database Systems. *The International Journal on Multimedia Tools and Applications*, 3(3):225–244, November 1996.

[22] Aidong Zhang, Biao Cheng, Raj Acharya, and Raghu Menon. Comparison of Wavelet Transforms and Fractal Coding in Texture-based Image Retrieval. In *Proceedings of the SPIE Conference on Visual Data Exploration and Analysis III*, pages 116–125, San Jose, January 1996.