Proceedings of the 6th ACM International Multimedia Conference (ACM MULTIMEDIA '98), September 1998, Bristol, England

Semantic Clustering and Querying on Heterogeneous Features for Visual data *

Gholamhosein Sheikholeslami, Wendy Chang and Aidong Zhang
Department of Computer Science
State University of New York at Buffalo
Buffalo, NY 14260, USA
gsesfah@cs.buffalo.edu, wcceec@rit.edu, azhang@cs.buffalo.edu

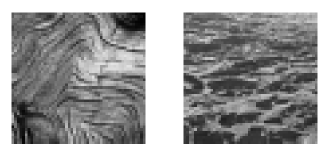
Abstract

The effectiveness of the content-based image retrieval can be enhanced using the heterogeneous features embedded in the images. However, since the features in texture, color, and shape are generated using different computation methods and thus may require different similarity measurements, the integration of the retrieval on heterogeneous features is a non-trivial task. In this paper, we present a semantics-based clustering approach, termed *SemQuery*, to support visual queries on heterogeneous features of images. Using this approach, the database images are classified based on their heterogeneous features. Each semantic image cluster contains a set of subclusters that are represented by the heterogeneous features that the images contain. A database image is included into a feature subcluster only if the image contains all the features under the same cluster. We also designed a multi-layer model to merge the results of basic queries on individual features. A visual query processing strategy is then presented to support visual queries on heterogeneous features. Experimental analysis is conducted and presented to demonstrate the effectiveness and efficiency of the proposed approach.

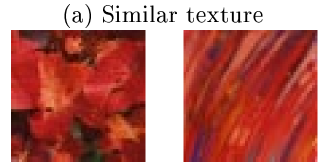
1 Introduction

Following the current research achievements [1], visual data in a database can be considered to contain feature vectors representing the content of the data. A feature vector normally represents one of the texture, color, and shape features. The similarity between two images is determined based on the distance between feature vectors of the images in the feature space. Features in texture, color, and shape are normally generated using different computational methods. For example, features in color are mostly generated using color histograms, color sets [24], or coherence color vector [17]. Features in texture can be generated using different methods such as wavelet-based feature extraction methods [23, 21], Fourier transforms [25], or fractals [29]. Thus, different features may have different similarity measurements. Because of this, the content-based retrieval process is normally performed on individual features. Several indexing methods [10, 4, 28] have been proposed to support efficient visual query retrieval based on the feature vectors.

However, the feature vectors of some semantically irrelevant images may be located very close in the feature space. Figure 1(a) presents two images of wood and water between which the distance of the texture feature vectors is very small, but these images are semantically not similar. Given a query whose feature vector is located in the neighborhood of the feature vectors of the two given images, both images are highly possible to be retrieved together in response to the query. Similarly, Figure 1(b) presents two images of leaves and painting that have close color feature vectors but are not semantically related. Thus, indexing itself based on the closeness of feature vectors in the feature space sometimes may not provide satisfactory solutions.



(a) Similar texture



(b) Similar color

Figure 1: Semantically different images with similar feature vectors.

Figure 2 (a) abstractly illustrates the above scenario in a two-dimensional feature space where two arbitrary shaped semantic clusters C_1 and C_2 exist. We may assume that C_1 and C_2 represent the images of wood and water in the texture feature space. Consider the query image q which belongs to the cluster C_1. If we only consider the closeness of feature vectors in a feature space to return relevant images, we may retrieve many images from cluster C_2 that are semantically irrelevant. To alleviate this problem, researchers have been studying the clustering of database images [27, 22]. If the database images are successfully classified into different semantic clusters, a visual query can be quickly narrowed to a specific category. Therefore, image retrieval can proceed effectively and efficiently. However,

*This research is supported by Xerox Corporation.

different semantic clusters can have arbitrary shapes and overlaps. It may be impossible to accurately obtain these shapes.

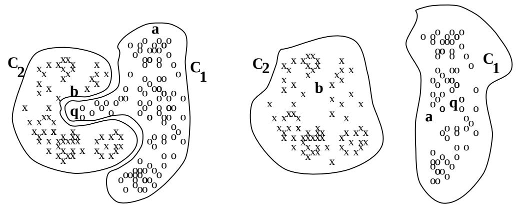


Figure 2: Views of two semantic clusters in two different feature spaces

We observe that the content-based retrieval process can be further improved using multiple features embedded in the query image. For example, clusters C_1 and C_2 in Figure 2(a) are far away in the color feature space, as shown in Figure 2(b). Given a query q, only images in C_1, which are semantically related to q, will be retrieved. Thus, to effectively retrieve relevant images for the query q, the retrieval can be based on color features. Since we cannot predict the characteristics of the query, all image features should be considered during retrieval to ensure effective retrieval. But, we must be careful on the integration of queries on heterogeneous features. Different features may play different degree of importance in making the final decision on the similarity between the query and database images. Most of the current image content-based retrieval systems use a weighted Euclidean distance to combine the similarity measurements of different feature classes [9, 3]. In these approaches, the results from different feature classes are linearly combined. However, studies have shown that various feature classes are not necessarily linearly-related. In addition, the set of weights of the feature classes should be somehow provided by the user. In FourEyes [14], a society of models is used instead of universal similarity measures or manual selection of relevant features. It provides a learning algorithm for selecting and combining groupings of data guided by example-based interaction with the user.

In this paper, we investigate an approach, termed *SemQuery*, to overcome the problems of traditional distance-based indexing and retrieval approaches. We assume that the semantics of the application image database is well defined and can be categorized by a domain expert. A hierarchy of the semantic clusters and database images is designed in which the database images are grouped based on their heterogeneous features. The features of each semantic cluster are represented by a set of templates. We then present a semantics-based clustering approach to supporting visual queries on heterogeneous features of images. In this approach, each semantic image cluster contains a set of subclusters that are represented by the heterogeneous features that the images contain. A database image is included into a feature subcluster only if the image contains all the features under the same cluster. We also designed a multi-layer neural network model to merge the results of basic queries on individual features. The input to the neural network is the set of image features and the output will be the similarity of images. A visual query processing strategy is presented to support visual queries on heterogeneous features. By narrowing the scope of the search to the semantic cluster, the experimental analysis shows that both effective and efficient retrieval can be achieved.

Section 2 presents the organization of the clusters, templates, and database images. Section 3 proposes the semantic clustering. Section 4 outlines the the query processing procedure. Section 5 presents the strategy for merging heterogeneous visual features based on a neural network model. Experiments are reported in Section 6, and concluding remarks are offered in Section 7.

2 Image Database Organization

2.1 A System Architecture for Using Heterogeneous Features in Image Retrieval

In this section, we first describe the structure of a general content-based image retrieval system, and then explain the importance and role of merging the results obtained from individual image features. Figure 3 shows the overall structure of our content-based image retrieval system, named *SemQuery*. Given a query image, the goal of content-based image retrieval is to retrieve all the images in the visual database whose content is similar to the query image. The first step of the system is to extract content of query image (in terms of different feature classes). The same type of features for database images are already extracted and stored in the database. The system then compares the features of the query image to those of database images, resulting in a group of retrieved image sets based on individual feature classes. Finally, the system merges the results obtained from individual feature classes based on their importance, to form the final set of retrieved images.

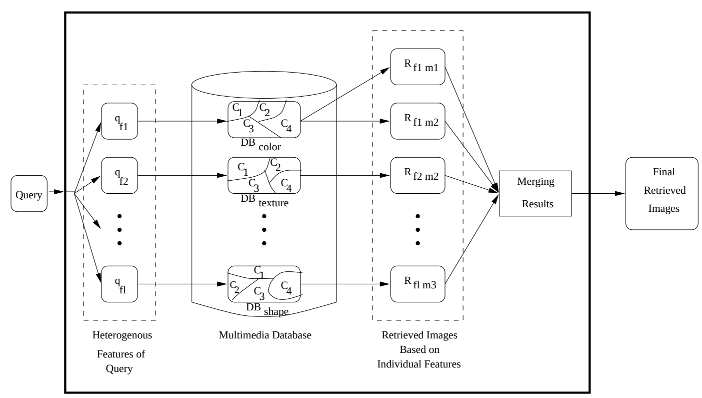


Figure 3: *SemQuery* system architecture.

The content of an image can be represented by the feature vectors in different feature classes, such as color, shape, texture, or text annotations. Different types of features can be extracted in each of these feature classes. For example, color histograms are usually used as the features for color feature class. Other methods such as color sets [24] or coherence color vector [17] are also proposed to extract color features. Features in texture can be generated using wavelet-based feature extraction methods [23, 21], Fourier transforms [25], fractals [29], random mosaic models [2], mathematical morphology [7], syntactic methods, and linear models [5]. Some of the proposed methods for shape features are in [13]. The similarity between two images is usually determined based on the distance between feature vectors of the images in the feature space. Since features in texture, color, and shape are generated using different computation methods, different features may have different similarity measurements. Because of this, the content-based retrieval process is normally performed on individual feature classes.

Thus, we assume that there exists a set of feature classes, denoted $F = \{f_1, ..., f_l\}$, where each feature class contains its own feature extraction method and its computation system on determining the similarity measurement between feature vectors within the feature class. For example, in a feature class f_i, texture features can be extracted using wavelet transform, whereas feature class f_j may use fractals to extract texture features. Both methods can use Euclidean distance to measure the similarity of feature vectors. In another feature class f_k, we can extract color features using color histograms and compare them by applying histogram intersection. A feature class may have more than one similarity measurement method. For example, we can also compare color histograms using Euclidean distance. As Figure 3 shows, given a query image q, the system first extracts its heterogeneous features q_{f_i}, where $1 \leq i \leq l$ and $f_i \in F$. Each q_{f_i} can be considered as a subquery of the query q.

At the second step, the system compares the features of the query image to the features of database images. The heterogeneous features of database images are usually extracted off-line and stored in DB_{f_i}, where $f_i \in F$. We assume that, for all feature classes $f_i, 1 \leq i \leq l$, the databases DB_{f_i} are logically separated from each other. However, they need not be physically separated and all may be in one database. The system then searches for each query feature vector q_{f_i} in the corresponding DB_{f_i} and returns the closest matches $R_{f_i m_j}$, where $R_{f_i m_j}$ is the set of retrieved images from DB_{f_i}, using measurement method m_j. The similarity measurement m_j can be Euclidean distance, Manhattan distance, histogram intersection, or any other appropriate measurement method. Note that for the same feature class, we may apply different similarity measurement methods that can result in different sets of retrieved images. Clustering techniques [27, 22] and indexing trees such as R-tree and its variants can be used to index individual feature cluster and thus speed up the search process in the databases [10, 4]. Corresponding to each retrieved image $r \in R_{f_i m_j}$, there is a numerical value s (either similarity or distance) that can be used to rank the retrieved images with respect to f_i and m_j. So the system can rank the retrieved images in each $R_{f_i m_j}$ based on individual feature classes.

Finally, the system must merge the results found by searching each individual database DB_{f_i}, to find the database images that are similar to the query image with respect to *all* the feature classes. Merging the results obtained by individual feature classes yields in the final ranked list of retrieved images. This last step is an essential and important part of retrieval based on heterogeneous features.

2.2 Clusters and Templates

Given a set of feature vectors, the feature space may not be uniformly occupied. In general, the distribution of the feature vectors of the images may be in an arbitrary shape in the feature space. Two feature vectors in two different semantic groups may be located closely in the feature space. Clustering the data identifies the sparse and the dense places, and hence discovers the overall distribution of patterns of the feature vectors.

In many existing clustering algorithms, k-medoid methods have been used, where each cluster is represented by the center of gravity of the cluster. For example, PAM (Partitioning Around Medoids) [11] was proposed to determine the most centrally located object *medoid* for each cluster. Each non-selected object is grouped with the medoid to which it is the most similar. Ng and Han introduced CLARANS (Clustering Large Applications based on Randomized Search) which is an improved k-medoid method [16]. Ester et al presented a clustering algorithm DBSCAN relying on a density-based notion of clusters which is designed to discover clusters of arbitrary shapes [8].

We thus assume that each cluster of the images can be represented by a set of feature vectors, denoted *templates*. The clusters can be further classified into subclusters, which can then be represented by their centroids. The benefit of this approach is that a hierarchical index can be built on the templates to support efficient query retrievals. Figure 4 shows an example in two dimensional space where clusters have arbitrary shapes. Multiple centroids are identified within the clusters to represent subclusters and these centroids can be used as the templates of the cluster. Note that each cluster may contain disconnected regions. There might be some objects which belong to multiple semantic clusters. Also we consider "outliers" which are the objects that do not belong to any of the semantic clusters.

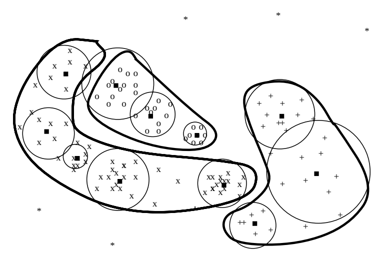


Figure 4: Arbitrary shaped clusters and templates.

2.3 Top-down Hierarchy of Clusters and Templates

In a large-scale image database, the images can be grouped into different applications, and within each application, the images naturally belong to different semantic clusters. For example, considering the airphoto images in a GIS application, four main semantic clusters, including grass, water, residence, and agriculture, are normally considered. These semantic clusters can also be divided into semantic subclusters.

Let $\mathcal{C} = \{C_1, ..., C_m\}$ be the lowest level semantic clusters. Under each C_i, most general categories of feature classes are specified, where each feature class represents a feature such as texture, color, and shape. Each feature class is further classified and is represented by a hierarchy of templates. Database images are grouped under templates, the grouping is based on criteria to be defined in the next section. Figure 5 presents a conceptual view of the hierarchical structure of the overall database organization. At the top-level, the most general categories of applications are defined. Within each application, semantic clusters are specified. For each semantic cluster, the general categories of features including texture, color and shape are specified. At the lowest level, feature templates representing the semantic cluster are defined.

Thus, both visual templates and text keywords describing the names of application, semantic cluster, and feature class are used to construct the database hierarchy. The top portion of the hierarchy (from root until feature class) can be displayed to the users through the Graphical User Interface. By displaying the top portion of the hierarchy, the names

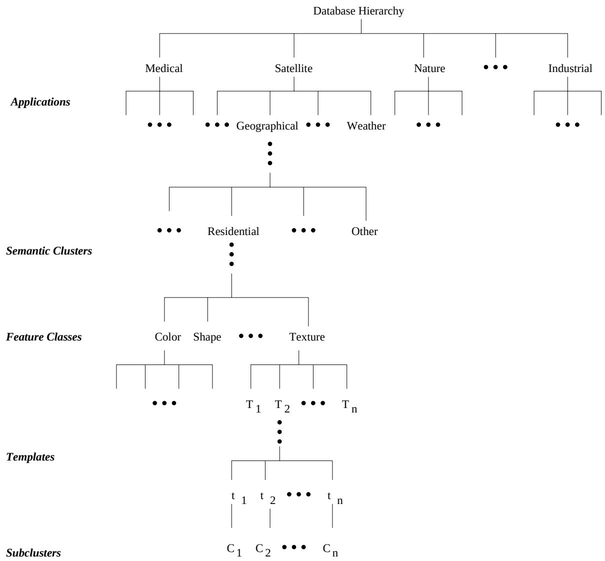


Figure 5: A conceptual view of the database hierarchy.

of application, semantic cluster, and feature class may be used as keywords to submit queries. If a query contains such keywords, the database can quickly be narrowed to the right subcluster. For example, given a query image and the name of the semantic cluster, the retrieval algorithm will only search the specified semantic cluster for relevant images. Similarly, given a query image and one specific feature class, the retrieval algorithm will only search based on that feature of the query image, all other features will not be compared. Thus, not only the effectiveness of the retrieval is greatly increased, but also flexibility in number and type of the feature classes to be used in retrieval is maintained. To support efficient query retrieval, the templates in each semantic cluster can be indexed hierarchically. A template at a higher-level represents a super cluster that contains all the subclusters represented by its child templates. A hierarchical template structure can be realized for visual databases to support efficient search of the subclusters which contain the query.

3 Clustering on Heterogeneous Features

In this section, we will discuss the clustering of the database images based on the visual templates configured for each semantic cluster. We define the scope of each semantic cluster. An image will be grouped into a semantic cluster if its heterogeneous features fall within the scope of the cluster.

Figure 6 demonstrates the intuition of our clustering approach. It shows a set representation of images of three semantic clusters. Each semantic cluster is represented by two sets, one representing images belonging to color feature class, and the other one, texture feature class. For the color feature class, the set of images in each of the semantic clusters (C_1^c, C_2^c and C_3^c) are shown by solid line. The sets drawn by dashed line (C_1^t, C_2^t and C_3^t) represent the semantic clusters based on texture feature class. A semantic cluster includes all the images which are within its scope. For each feature class, every semantic cluster is composed of a set of subclusters (not shown in the figure), and its scope is the union of scopes of the subclusters. The scope of a semantic cluster based on both color and texture feature classes would be the intersection of the scopes of the clusters of the two feature classes. For example, the scope of the semantic cluster C_1 is $C_1^c \cap C_1^t$. An image will be assigned to a semantic cluster if it falls within the scope of all the heterogeneous clusters of the semantic cluster. For example, image q in Figure 6 will be assigned to the semantic cluster C_2, because it belongs to both C_2^t and C_2^c. However, since image p is not within the scope of C_2^t, it will not be assigned to the cluster C_2.

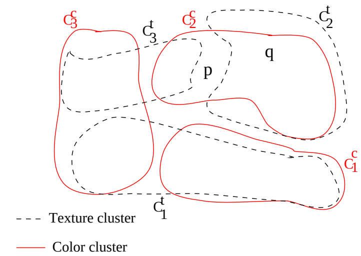


Figure 6: Set representation of image clusters based on color and texture feature classes.

We now formally present our clustering approach. Let $\mathcal{F} = \{f_1, ..., f_l\}$ be a set of feature classes, where f_i refers to texture, color, shape, etc. Let C_k be a semantic cluster which may be represented by a set of subclusters $\{c_1^{k,f_i}, ..., c_{h_k}^{k,f_i}\}$ in feature class f_i, where $f_i \in \mathcal{F}$. Given a subcluster c_j^{k,f_i}, let template t_j^{k,f_i} be the centroid of the c_j^{k,f_i}. We define the *scope of subcluster* c_j^{k,f_i}, denoted s_j^{k,f_i}, to be the part of the feature space that contains the images belonging to c_j^{k,f_i}. For example, we may define μ_j^{k,f_i} and ν_j^{k,f_i} to be the mean and variance of distances of the images to the template t_j^{k,f_i}. The *scope of subcluster* c_j^{k,f_i}, s_j^{k,f_i}, is then measured to include those images whose distance to the template t_j^{k,f_i} is less than the radius defined below:

$$\mu_j^{k,f_i} + \beta_{f_i} * \nu_j^{k,f_i}, \tag{1}$$

where parameter β_{f_i} is a factor for each feature space and must be determined by experiments. In a two dimensional space, the scope of subcluster c_j^{k,f_i} is a circle with radius $\mu_j^{k,f_i} + \beta_{f_i} * \nu_j^{k,f_i}$, whose center is the centroid of the subcluster, t_j^{k,f_i}. Other approaches to defining the scope of the subcluster can also be used.

We define the *scope of the semantic cluster* C_k in feature class f_i, denoted $S_k^{f_i}$, as the union of the scopes of all subclusters of C_k in feature class f_i:

$$S_k^{f_i} = s_1^{k,f_i} \cup s_2^{k,f_i} \cup ... \cup s_{h_k}^{k,f_i}. \tag{2}$$

The scope of the semantic cluster C_k based on the heterogeneous features in $\mathcal{F}$, denoted $\mathcal{S}_k$, is then defined as the intersection of scopes of feature classes:

$$\mathcal{S}_k = S_k^{f_1} \cap S_k^{f_2} \cap ... \cap S_k^{f_l}. \tag{3}$$

We now discuss the clustering of the database images by first precisely defining the relationship between an image and the scope $\mathcal{S}_k$ of the semantic cluster C_k. Let the feature vectors of a database image be $\mathcal{V} = \{\langle v_1 \rangle, \langle v_2 \rangle, \ldots, \langle v_l \rangle\}$, where v_j represents a feature vector in feature class f_j in $\mathcal{F}$. Let $\mathcal{S}_k^{f_i}$ consist of subclusters s_1^{k,f_i}, ..., $s_{h_k}^{k,f_i}$. Feature vector v_i in $\mathcal{V}$ is considered to be in the scope of $\mathcal{S}_k^{f_i}$ if it is in the scope of any s_1^{k,f_i}, ..., or $s_{h_k}^{k,f_i}$. Formally, we have the following definition:

Definition 1 ($v_i \in S_k^{f_i}$) *Given a semantic cluster C_k which is represented by a set of subclusters $c^{k,f_i} = \{c_1^{k,f_i}, ..., c_{h_k}^{k,f_i}\}$ in feature class f_i, where $f_i \in \mathcal{F}$. Let $s^{k,f_i} = \{s_1^{k,f_i}, ..., s_{h_k}^{k,f_i}\}$ be the set of corresponding subcluster scope of c^{k,f_i}, and $S_k^{f_i} = s_1^{k,f_i} \cup ... \cup s_{h_k}^{k,f_i}$. A feature vector v_i is within the scope of $S_k^{f_i}$ if:*

$$\exists s_j^{k,f_i} \in s^{k,f_i}, \quad v_i \in s_j^{k,f_i}.$$

Given a template t_j^{k,f_i} which is the centroid of the subcluster c_j^{k,f_i}. Let $v_{t_j^{k,f_i}}$ be the feature vector of t_j^{k,f_i} and let s_j^{k,f_i} be the scope of subcluster c_j^{k,f_i} with radius $\mu_j^{k,f_i} + \beta_{f_i} * \nu_j^{k,f_i}$. Let $d(v_p, v_t)$ denote the distance between the feature vectors $v_p(a_1, ..., a_l)$ and let $v_t(b_1, ..., b_l)$. We use the *root mean square metric* to measure the distances between the images. Following Definition 1, a feature vector v_i which is within the scope of $S_k^{f_i}$ can be defined as follows:

$$\exists s_j^{k,f_i} \in s^{k,f_i}, |d(v_i, v_{t_j^{k,f_i}}) - (\mu_j^{k,f_i} + \beta_{f_i} * \nu_j^{k,f_i})| \leq \theta_{f_i},$$

where θ_{f_i} is a given threshold for each feature class f_i. The radius of a subcluster, $\mu_j^{k,f_i} + \beta_{f_i} * \nu_j^{k,f_i}$, can be estimated based on sample images. In practice, different approaches may be designed to accommodate various scopes in applications. Using Definition 1, a database image can be assigned to subcluster c_j^{k,f_i} if it falls within the scope of the subcluster, or if its distance to the border of the subcluster is less than θ_{f_i}.

Definition 1 defines the scope of $S_k^{f_i}$ based on the locations of the feature vectors. As we have mentioned, a feature vector in feature class f_i which does not semantically belong to C_k may belong to $S_k^{f_i}$. To more precisely cluster the images under each feature class, we use heterogeneous features in determining the classification of the images in each $S_k^{f_i}$. Given a semantic cluster C_k and its scope $S_k^{f_i}$ in each feature class f_i, to cluster the database image m with feature vector $\mathcal{V}$ into cluster C_k, all feature vectors $v_1, v_2, ..., v_l$ in $\mathcal{V}$ must fall within the scopes of $S_k^{f_1}, S_k^{f_2}, ..., S_k^{f_l}$. Thus, an image will be assigned to a semantic cluster if all its features are represented in the cluster. Within each semantic cluster, the image is grouped into the subclusters represented by the templates matched by the image. If any of the feature vectors in m is not within the scope in C_k, then we do not consider that m belongs to C_k. We introduce the following definition:

Definition 2 ($\mathcal{V} \in C_k$) *Given a semantic cluster C_k and a set of feature classes $\mathcal{F} = \{f_1, ..., f_l\}$, where the scope of C_k in each feature class f_i is represented as $S_k^{f_1}, S_k^{f_2}, ..., S_k^{f_l}$, an image $\mathcal{V} = \{\langle v_1 \rangle, \langle v_2 \rangle, \ldots, \langle v_l \rangle\}$ is classified into C_k if*

$$\forall v_i \in \mathcal{V}, v_i \in S_k^{f_i}.$$

Note that images that are included in the scopes of more than one semantic clusters will be assigned to both semantic clusters.

The existence of *outliers* is also considered in the proposed clustering approach. Outliers are objects that far away from all the templates and do not have similar semantics to any of the predefined semantic clusters. Forcing to link the outliers with a cluster will cause inaccurate retrievals. In our approach, outliers are grouped into a special cluster, termed *other* cluster, denoted C_t, and is searched separately. Given an image $\mathcal{V} = \{\langle v_1 \rangle, \langle v_2 \rangle, \ldots, \langle v_l \rangle\}$, where v_j represents a feature vector in feature classes f_j in $\mathcal{F}$. An image $\mathcal{V}$ is assigned to cluster C_t if at least one of the feature vectors v_i is not assigned to any semantic cluster:

Definition 3 ($\mathcal{V} \in C_t$) *Given a semantic cluster C_k and a set of feature classes $\mathcal{F} = \{f_1, ..., f_l\}$, where the scope of C_k in each feature class f_i is represented as $S_k^{f_1}, S_k^{f_2}, ..., S_k^{f_l}$. An image $\mathcal{V}$ is assigned to the special* other *cluster C_t, if*

$$\exists v_i \in \mathcal{V}, v_i \notin S_k^{f_i}.$$

We now consider the images shown in Figure 1(a). In this example, the image water is assigned to the semantic cluster *water*, since it can be matched to at least one texture and one color templates of the water cluster. Although the image wood can be matched to at least one texture template, it is not assigned to the semantic cluster *water*, since it does not match with any color templates of the water cluster. Similarly, in Figure 1(b), the image leaves is assigned to the semantic cluster *leaves*, since it can be matched to at least one texture and one color templates of the leaves cluster. Although the image painting can be matched to at least one color template, it is not assigned to the semantic cluster *leaves*, since it does not match with any texture templates of the leaves cluster.

New semantic clusters may be added by the domain expert due to the changes in the content of the database. In such cases, new images will be collected to generate the templates representing the new semantic clusters. Thus, given a set of existing semantic clusters $\{C_1, ..., C_m\}$ and their corresponding scopes $\{\mathcal{S}_1, ..., \mathcal{S}_m\}$, let $\mathcal{S}'$ be the scope for the new semantic cluster C'. The new semantic cluster C' may be considered to be similar to the cluster C_i ($1 \leq i \leq m$) if the scope $\mathcal{S}'$ overlaps with $\mathcal{S}_i$. In such cases, the images that are grouped under cluster C_i and the images that are included in the *other* cluster may belong to the new semantic cluster C'. These images must be regrouped following the steps outlined above.

Similarly, new templates may be added into a semantic cluster due to the additions of new images. In such cases, the scope of the semantic cluster will be changed. The new scope must be compared to the scopes of the other existing semantic clusters to determine the similarity. Images that are grouped under similar semantic clusters and the images that are included in the *other* cluster must also be regrouped.

4 Query on Heterogeneous Features

The query can be a still image or a video frame. An optional keyword, which describes the feature of interest, application domain, or semantics of the query, may also be given. The keyword can be used to narrow the scope of the features and images to be searched in the retrieving process.

If a keyword is attached to the query image and the keyword matches the name of a semantic cluster of an application, only images belonging to the specified semantic cluster will be considered in the retrieving process. Similarly, if only a feature class is specified, the retrieval can be based on the specified image features. Boolean combinations of the keywords can be supported. For example, given a query image containing its texture features as well as keyword *floral*, the retrieval will be based on the texture features and only the images within the floral semantic cluster will be considered in the retrieving process. If no keyword is attached to the query image, the retrieving process will be based on the image features and it will find the matched semantic clusters for the query.

Upon receiving a query q, a set of subqueries $\{q_1, .., q_n\}$ are generated from the query, with each subquery being a feature vector corresponding to a feature class in $\mathcal{F}$. We then compare the subqueries to the templates to determine the matched templates for each subquery.

Let $\mathcal{T}_{f_i} = \{t_1, ..., t_{m_i}\}$ denote the set of templates for feature class f_i. Let $\mu_{t_j} + \beta * \nu_{t_j}$ be the radius of the corresponding scope for the template t_j, $t_j \in \mathcal{T}_{f_i}$. A set of matched templates $\mathcal{T}_{q_i}$ is selected for subquery q_i based on the following criterion:

$$\mathcal{T}_{q_i} = \{t | t \in \mathcal{T}_{f_i} \wedge |d(q_i, t) - (\mu_t + \beta * \nu_t)| \leq \theta_{f_i}\}. \quad (4)$$

Using Formula (4), a template is selected if the subquery falls within the scope of the template or if the distance between the subquery and the border of the subcluster represented by the template is less than θ_{f_i}.

Let $\mathcal{C} = \{C_1, ..., C_m\}$ be the set of semantic clusters that exists in the system, and let $\mathcal{T}_{f_i}^{C_k} = \{t_1^{k,f_i}, ..., t_i^{k,f_i}\}$ denote the set of templates representing cluster C_k $(1 \leq k \leq m)$ in the feature class f_i. We determine the set $\mathcal{C}_q$ of the matched semantic clusters for the query $q = \{q_1, ..., q_n\}$ based on the following criterion:

$$\mathcal{C}_q = \{C_k | C_k \in \mathcal{C} \wedge \forall q_i \in q (\exists f_i \in \mathcal{F}), \mathcal{T}_{q_i} \cap \mathcal{T}_{f_i}^{C_k} \neq \emptyset\}. \quad (5)$$

That is, a semantic cluster C_k is chosen if, for every subquery q_i, at least one matched template can be found in C_k. The retrieval algorithm then searches the corresponding subclusters of the matched templates within the chosen semantic clusters to retrieve a list of relevant images for the query.

Querying on single features. Let q contain a single subquery q_i and a set of chosen semantic clusters be $\mathcal{C}_q = \{C'_1, ..., C'_h\}$. Consider q_i matching with multiple independent templates $t_1^{k,q_i}, ..., t_{m_i}^{k,q_i}$ within cluster C'_k $(1 \leq k \leq h)$, and $I_{t_j}^{k,q_i}$ being the set of relevant images retrieved within the subcluster represented by template t_j^{k,q_i} $(1 \leq j \leq m_i)$. The list of relevant images $r(q_i, C'_k)$, within cluster C'_k for subquery q_i, is defined as the union of all relevant images in $I_{t_j}^{k,q_i}$, $1 \leq j \leq m_i$:

$$r(q_i, C'_k) = \bigcup_{j=1}^{m_i} I_{t_j}^{k,q_i}. \quad (6)$$

The relevant images returned for each chosen semantic cluster are then unioned to generate the final list of relevant images, denoted $\Re_q$, for the query q:

$$\Re_q = \bigcup_{j=1}^{h} r(q_i, C'_j). \quad (7)$$

Let S'_k be the scope of C'_k. Clearly, using the clustering approach defined in Definition 2, the images to be searched in $\mathcal{C}_q$ will not be larger than the scope of $\bigcup_{j=1}^{h} S_j^{\prime f_i}$, that is, $\mathcal{C}_q \subseteq \bigcup_{j=1}^{h} S_j^{\prime f_i}$. Thus, our clustering approach provides a more focused set of images to be searched within the whole database. Also, since each semantic cluster is formed based on the heterogeneous features, the SemQuery approach helps reduce the percentage of *false-positives* (images falsely included in a cluster) for querying on individual features.

Querying on heterogeneous features. Let q contain a set of subqueries $q_1, ..., q_n$ and a set of chosen semantic clusters be $\mathcal{C}_q = \{C'_1, ..., C'_h\}$. Let each q_i match with multiple independent templates $t_1^{k,q_i}, ..., t_{m_i}^{k,q_i}$ within cluster C'_k and $I_{t_j}^{k,q_i}$ be the set of relevant images retrieved from the subclusters represented by template t_j^{k,q_i}. Similar to the single feature querying, the list of relevant images $r(q_i, C'_k)$, within cluster C'_k for subquery q_i, can be defined using (6).

Since we want to find those images which contain features similar to all subqueries, the set of relevant images within cluster C'_k for query q, denoted $r(q, C'_k)$, is calculated as the intersection of $r(q_i, C'_k)$, $i = 1,...,n$:

$$r(q, C'_k) = \bigcap_{i=1}^{n} r(q_i, C'_k). \quad (8)$$

Similar to (7), the relevant images returned for each chosen semantic cluster are then unioned to become the final list of relevant images $\Re_q$,

$$\Re_q = \bigcup_{j=1}^{h} r(q, C'_j). \quad (9)$$

The retrieved images in $\Re_q$ may be further ranked using the multi-layer neural network model to be introduced in the next section. Note that, if no semantic cluster is selected for the query, the special cluster *other* will be searched. In such cases, the search will be performed sequentially on the special *other* cluster.

As we mentioned above, since each semantic cluster is formed based on the heterogeneous features, our approach can reduce the percentage of false-positives for each query based on either individual features or heterogeneous features. On the other hand, we may possibly increase a small percentage of misses for some queries. The details on this aspect will be discussed in Section 6.

5 Ranking Images Using Heterogeneous Features

Given a query image, a set of relevant images can be selected based on individual features, as discussed in the previous sections. However, a final ranked set of similar images to the query must be derived by merging the individual features of these images. In this section, we will discuss the nonlinear relationships existing in merging heterogeneous features and propose a neural network model to merge the results obtained by searching heterogeneous features. Our neural network model does not restrict the relationships between feature classes to be linear and can support nonlinearity.

5.1 Issues in Merging Heterogeneous Features

Various features may play different degrees of importance in making the decision on ranking database images. The ranking process must assign weights to the results obtained from the subqueries. In human perception, these feature classes do not have the same importance in distinguishing images. Thus, visual retrieval systems must consider the degree of importance of each feature class to determine the overall similarity of database images to the query image.

Most of the current image content-based retrieval systems such as Photobook [18], QBIC [9], Virage [3], and NETRA [12] use a weighted linear method to combine the similarity measurements of different feature classes. That is, given the similarity measurements $z_1, ..., z_l$ of a database image to a query image with respect to feature classes $f_1, ..., f_l$ and the corresponding weights $w_1, ..., w_l$, the overall similarity is calculated as $\sum_{i=1}^{l} w_i z_i$. We call such weighted linear combination of similarity measurements as "*linear merging*". In the existing systems, the *user* directly specifies the weights. For example, the user should say "retrieve images using 50% of color feature class, 30% of texture feature class, and 20% of shape feature class". But users do not naturally sort images by similarity using this kind of language. In particular, as the number of feature classes increases, intuition about how to pick relative weightings among features is lost [19]. In addition, studies have shown that various feature classes are not necessarily linearly-related. For example, the similarity measures of color and texture do not generally show a linear exchange. An important step in merging heterogeneous features is to apply a nonlinear transformation on each similarity measurement to make them more commensurate [15].

5.2 Neural Network Model to Merge Heterogeneous Features

Neural networks have been used in many areas such as pattern recognition, computer vision and control systems. We propose a multi-layer perceptron neural network, *NeuroMerge*, to merge the results obtained from the heterogeneous features. The input to the neural network is the set of measurements z_i between images $\mathcal{M}_1$ and $\mathcal{M}_2$ for all the feature classes. If all the features of images $\mathcal{M}_1$ and $\mathcal{M}_2$ are similar, we want that the output of the neural network to be close to 1. However, if $\mathcal{M}_1$ and $\mathcal{M}_2$ are not similar, the output should be close to 0. The network implements a set of functions $o_i = F_i\{z_k\}$ from input variables z_k to output o_i, where $\{z_k\}$ means $z_1, z_2, \ldots, z_I$. Cybenko has proven that using multi-layer neural network, to approximate a particular set of functions $F_i\{z_k\}$, at most two hidden layers are needed. Arbitrary accuracy is obtainable given enough units per layer. It has also been proven that only one hidden layer is enough to approximate any continuous function [6]. Consequently, our neural network model does not restrict relationship between feature classes to be linear and can support nonlinearity.

To train the neural network and find the weights, only a set of images that are visually similar (positive examples) and a set of images that are not similar (negative examples) should be provided. The system then finds the similarity (or dissimilarity) between images based on different feature classes and feeds these measurements to the neural network. Once the network is trained, the feature classes will have the proper weights, so they can be used in merging heterogeneous features. In this approach, user need not worry about assigning weights to feature classes. In the mean time, NeuroMerge assigns the weights based on human perception. Contrary to linear merging, in NeuroMerge, each feature class can be measured in terms of similarity or distance, independent of others and the combination of these similarities and distances can be directly fed into the neural network. In this respect, it makes the neural network model more flexible than the previous approaches.

We used back propagation algorithm to train a neural network with a single hidden layer [30]. In this algorithm the output error signal is propagated through the network and is used to modify the weights. Let there be I input neurons one for each feature class, J neurons in the hidden layer and K neurons in the output layer. In our proposed network, we will have only one output neuron, that is K=1. The inputs are denoted by $z_1, z_2, \ldots, z_I$, the outputs of hidden neurons by $y_1, y_2, \ldots, y_J$ and the final outputs are denoted by $o_1, o_2, \ldots, o_K$. Figure 7 shows the structure of the neural network, where v_{ij} denote the weight of connections from input unit i to hidden unit j and w_{jk} denote the weight connections between hidden unit j to output unit k. All the neurons are fully connected. The details of this approach can be found in [20].

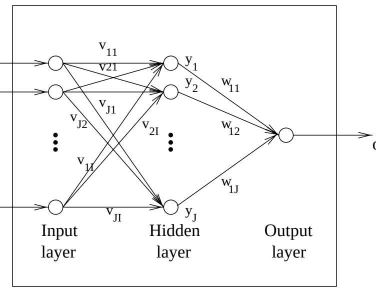


Figure 7: Structure of the proposed neural network

Given a query image q with heterogeneous features q_{f_i}, where $1 \le i \le l$ and $f_i \in F$, consider the set of retrieved images $R_{f_i m_j}$ from database DB_{f_i} using measurement m_j as defined in Section 2.1. Let us define,

$$\Re = \bigcap_{i,j} R_{f_i m_j}, \qquad 1 \le i \le l, \;\; 1 \le j \le p, \;\; f_i \in F, \;\; m_j \in M,$$

where $M = \{m_1, m_2, \ldots, m_p\}$ is the set of similarity measurement methods. Our goal is to rank the relevant images in $\Re$ with respect to all the heterogeneous features. Using the trained neural network, we can find the similarity between the query image q and each image in $\Re$ based on the heterogeneous features. In each step, the similarity measurements of heterogeneous feature classes of the query image and an image in $\Re$ are fed into the neural network. The output value of the network will be the similarity between the two images. We can sort and rank the images in $\Re$ based on the output of neural network.

Note that the trained neural network can be used to determine the similarity between the query and all the images in the database. However, such an approach will result in a linear search to all images in the database, causing inefficiency in querying. By combining the semantics-based clustering as well as querying with the neural network model, both effectiveness and efficiency can be achieved.

6 Evaluation

In the experiments, we used 29,400 texture and color feature vectors of images. We categorized the database images based on their semantics into five categories of cloud, floral, leaves, mountain, and water. For each semantic cluster, about 10% of feature vectors were chosen as training data. We then performed the hierarchical clustering approach on both texture and color features of training data to configure texture and color templates for each semantic cluster [22]. To extract the texture features, we applied wavelet transform and used its multi-resolution property as described in [22]. For the color features, the system finds the color histograms of the image, using method proposed in [26].

6.1 Performance of the Semantic Clustering

We now present the performance of our semantic clustering approach. Since the other indexing methods do not consider semantics, they do not have the common ground to be directly compared to our approach. In our experiments, the images are assigned to a semantic cluster following the steps outlined in Section 3. Given a visual database and a set of clusters $\mathcal{C}$, let A_c be the set of images assigned to the cluster c ($c \in \mathcal{C}$) by the clustering approach and let I_c be the ideal set of images that have been visually inspected to be semantically related to cluster c. We measure the precision P_c and recall R_c of the clustering approach by comparing the set of images assigned to a cluster with the ideal set of images. The precision P_c and recall R_c of the clustering for the cluster c are calculated as follows:

$$P_c = \frac{|I_c \bigcap A_c|}{|A_c|}, \quad R_c = \frac{|I_c \bigcap A_c|}{|I_c|}. \tag{10}$$

The threshold θ_{f_i} used to assign a database image to the cluster *other* is set at 150. The factor β_{f_i} used to adjust the scope of a subcluster is set at 0 for both texture and color features. The system first clusters images based on texture features only. The images are then clustered based only on color features. Finally, the images are clustered based on both heterogeneous color and texture features. The experimental results are reported in Table 1.

Cluster c	Both		Color		Texture	
	R_c	P_c	R_c	P_c	R_c	P_c
Cloud	1.00	0.79	1.00	0.79	1.00	0.23
Floral	1.00	0.93	1.00	0.61	1.00	0.55
Leaves	0.91	0.90	0.91	0.52	0.97	0.69
Mountain	0.96	0.82	1.00	0.77	0.96	0.36
Water	0.71	0.31	0.86	0.31	0.81	0.10

Table 1: Precision and recall of semantic clustering.

Table 1 shows that the precision of the semantic clustering approach is significantly higher when both features are used than that of the cases where only the texture features or only the color features are used. Using heterogeneous features, we can reduce the *false-positives* (images falsely included in a cluster), so A_c is smaller and contains more relevant images. Thus P_c is higher. Also, since A_c is the joint result from all heterogeneous features, some *positives* (images that should be included in cluster) may be missed, and thus R_c may become smaller. In Table 1, if we compare P_c and R_c of clusters based only on texture to those of both color and texture, we observe an average 0.36 increase in P_c and 0.03 decrease in R_c. Similarly, comparison of P_c and R_c values of clusters based only on color to the ones using both color and texture shows an average 0.16 increase in P_c and 0.04 decrease in R_c. These results show that when applying both color and texture features in clustering, the increase in precision of clustering P_c (or reduction of false-positives) is much higher than the decrease in recall of clustering R_c (or increase in missing positives). Thus, using heterogeneous features in SemQuery provides a better overall performance than using individual features.

Using either individual or heterogeneous features, we have lower performance for the cluster water. After examining the training images and the templates, we observed that the cluster water, even after considering its heterogeneous features, still overlaps with some of the other clusters. The reason is that the current extracted texture and color features are not effective enough to correctly distinguish images of water from others. This issue can be improved by further enhancement of feature extraction methods.

6.2 Performance of the Image Retrieval

We examine the effectiveness of semantic clustering on the retrieval based on heterogeneous features. In this context, we consider all the existing methods that retrieve the closest images in the feature space to the query image as *nearest neighbor retrieval*. They include retrieval using serial search, index trees such as R-tree and its variants, and traditional template-based clustered database. When the neighboring images to the query image have the same semantics as that of the query image, both nearest neighbor retrieval and the retrieval based on proposed semantic clustering are roughly equivalent. But when the query image is located near the boundary of semantic clusters, some of its nearest neighbors are not semantically relevant to it. Semantic-based clustering helps avoid retrieving images in irrelevant semantic clusters. So, in such cases it will be more effective than the nearest neighbor retrievals.

We chose 19 query images for our experiments. The ideal set of relevant images of each query q, denoted I_q, was determined by visual inspection. Each query image is decomposed into a texture subquery and a color subquery. The system then finds the matched templates for each subquery and determines the matched semantic clusters for the query.

To retrieve relevant images for query q, the system first retrieved the top n closest images within the matched semantic cluster with respect to the texture subquery. The same process was repeated for the color subquery. The intersection of the two closest image sets is the set of relevant images, denoted $\Re_q^n$. The images in $\Re_q^n$ are further ranked using the multi-layer neural network. Note that the number of images in $\Re_q^n$ may be less than n. We use precision P_q and recall R_q to measure the performance of the retrieval on query q:

$$P_q = \frac{|I_q \bigcap \Re_q^n|}{|\Re_q^n|}, \quad R_q = \frac{|I_q \bigcap \Re_q^n|}{|I_q|}. \tag{11}$$

We also compare the retrieval from semantic clustered database to the nearest neighbor retrieval approach. For the nearest neighbor retrieval approach, the system first retrieved the top n closest images among all database images with respect to the texture subquery. The same process was repeated for the color subquery. The intersection of the two closest image sets is the set of retrieved images, denoted $\aleph_q^n$. Note that the number of images in $\aleph_q^n$ may be less than n. Since number of images in $\aleph_q^n$ and $\Re_q^n$ may be different, the precision of both approaches cannot be directly compared using Formula (11). We define the *retrieval rate* E_n as the performance measurement:

$$E_n = \frac{|I_q \bigcap L^n|}{n}, \tag{12}$$

where L^n is $\Re_q^n$ for the semantic clustered database or $\aleph_q^n$ for the nearest neighbor retrieval. E_n can be used as the measurement to compare the effectiveness of SemQuery to the nearest neighbor approaches, where the same number of images (n) selected based on individual features are intersected, but different number of images may be eventually retrieved.

Precision and recall of all queries are shown in Table 2 when $n = 20$. The average number of relevant images in I_q for all queries is 20.9. The average number of images in $\Re_q^n$ of all queries is 9.4. The average precision P_q value of 0.85 indicates that among the average 9.4 images retrieved, 8 are relevant to the query. A recall R_q value of 0.45 indicates that only 9.4 of the average 20.9 relevant images are retrieved. This is due to the small number of intersected images ($\Re_q^n$). The R_q value is likely to improve when more than 20 images were retrieved for each feature class, thus increases the number of images in $\Re_q^n$. Table 2 also shows the retrieval rate E_n for every query. The retrieval rates may seem to have low values, but the corresponding precisions show the high effectiveness of the retrieval. Retrieval rate may not properly reflect the absolute effectiveness of retrieval, but it is an appropriate measurement to compare the relative effectiveness of our approach with the nearest neighbor approaches where both retrieve different number of images based on intersection of the same number of images.

Query	R_q	P_q	E_n
c1	0.67	1.00	0.50
c2	0.40	1.00	0.30
c3	0.32	1.00	0.50
c4	0.32	0.91	0.50
c5	0.29	0.75	0.45
f1	0.42	1.00	0.50
f2	0.33	0.50	0.20
f3	0.33	0.83	0.25
f4	0.38	1.00	0.45
f5	0.29	1.00	0.35
l1	0.26	1.00	0.45
l2	1.00	0.50	0.25
l3	0.26	1.00	0.50
l4	1.00	0.71	0.25
m1	0.42	1.00	0.50
m2	0.39	0.78	0.35
m3	0.46	1.00	0.55
w1	0.46	0.46	0.30
w2	0.62	0.73	0.40
Average	0.45	0.85	0.40

Table 2: Recall R_q, precision P_q, and retrieval rate E_n of semantic clustered database ($n = 20$).

Table 3 summarizes retrieval rate E_n and recall R_q with respect to different values of n. In all four tests, the semantic clustering outperformed the nearest neighbor approach. Table 3 also shows increasing the value of n increases the recall of retrieval but results in slight reduction in retrieval rate E_n. That is, more relevant images will be retrieved but some irrelevant ones will also be retrieved.

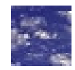
n	Semantic Clustering		Nearest Neighbor	
	R_q	E_n	R_q	E_n
10	0.27	0.42	0.17	0.28
15	0.37	0.41	0.25	0.28
20	0.45	0.40	0.32	0.28
25	0.51	0.38	0.37	0.27

Table 3: Recall R_q and retrieval rate E_n.

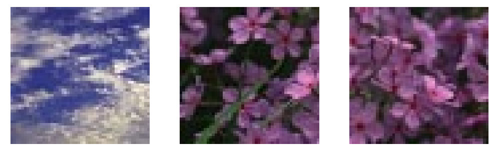
Table 4 shows the average recall and retrieval rate of $n = 20$ with respect to different query types. The semantic clustering outperformed the nearest neighbor approach in both recall and retrieval rate of all clusters.

Query Type	Semantic Clustering		Nearest Neighbor	
	R_q	E_n	R_q	E_n
Cloud	0.40	0.45	0.28	0.29
Floral	0.35	0.35	0.31	0.31
Leaves	0.63	0.36	0.43	0.24
Mountain	0.42	0.47	0.27	0.30
Water	0.54	0.35	0.35	0.23

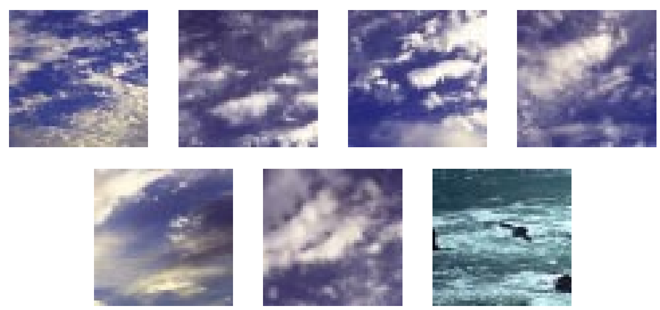
Table 4: Recall R_q and retrieval rate E_n ($n = 20$).



(a)



(b)



(c)

Figure 8: (a) Query image; (b) Retrieved images using nearest neighbor retrieval ($\aleph_q^n$); (c) Retrieved images from semantically clustered database($\Re_q^n$).

Figure 8 presents an example of the retrieval results with $n = 20$. The query image is taken from one of cloud images. Both nearest neighbor retrieval and SemQuery correctly returned the image that the query is taken from (See the first image in lists (b) and (c)). However, $\aleph_q^n$ contains only three images and two of the three images are floral images which are semantically different from the query image. Although the color feature vectors of other cloud images are very close to the color feature vector of the query, their texture feature vectors are not as close as those of floral images. Thus nearest neighbor retrieval approach first retrieves floral images. During retrieval based on texture, SemQuery only searches the images in the cloud semantic cluster which does not include any pink images (such as floral images). Thus it does not retrieve the floral images even though their texture is more similar to that of query image. As Figure 8(c) shows, 7 images are included in $\Re_q^n$ and all but 1 of them are clouds.

7 Conclusion

In this paper, we have proposed an approach, termed SemQuery, to supporting visual queries on heterogeneous features of images. We have designed a semantics-based clustering approach for the classification of database images. This clustering mechanism can categorize the images into different clusters based on their heterogeneous features. We have also designed a multi-layer model to merge features of the images. This model has been used to successfully rank the images retrieved based on individual features. A comprehensive visual query processing strategy is then presented to support visual queries on heterogeneous features. By successfully combining the semantics-based and template clustering both effectiveness and efficiency have been achieved. Experimental analysis has demonstrated the effectiveness and efficiency of the proposed approach.

References

[1] Special Issue on Content-Based Image Retrieval Systems, Editors V. N. Gudivada and V. V. Raghavan. *IEEE Computer*, 28(9), 1995.

[2] N. Ahuja and A. Rosenfeld. Mosaic models for texture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(1):1–11, 1981.

[3] J.R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Jain, and C.F. Shu. The virage image search engine: An open framework for image management. In *Proceedings of SPIE, Storage and Retrieval for Still Image and Video Databases IV*, pages 76–87, San Jose, CA, USA, February 1996.

[4] 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.

[5] G.R. Cross and A.K. Jain. Markov random field texture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(1):25–39, 1983.

[6] G. Cybenko. Approximation by superimposing of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

[7] E.R. Dougherty and J.B. Pelz. Texture-based segmentation by morphological granulometrics. In *Advanced Printing of Paper Summaries, Electronic Imaging '89*, volume 1, pages 408–414, Boston, Massachusetts, October 1989.

[8] M. Ester, H. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of 2nd International Conference on KDD*, 1996.

[9] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, and B. Dom et al. Query by Image and Video Content: The QBIC System. *IEEE Computer*, 28(9), 1995.

[10] A. Guttman. R-Trees: A Dynamic Index for Geometric Data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1984.

[11] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[12] W. Y. Ma and B. S. Manjunath. NETRA: A toolbox for navigating large image databases. In *IEEE International Conference on Image Processing*, 1997.

[13] Rajiv Mehrotra and James E. Gary. Similar-shape retrieval in shape data management. *IEEE Computer*, 28(9):57–62, September 1995.

[14] T. P. Minka and R. W. Picard. Interactive learning using a "society of models". Technical Report 349, M.I.T. Media Laboratory Perceptual Computing Section, 1995.

[15] Thomas Minka. An image database browser that learns from user interaction. Master's thesis, MIT, 1996.

[16] R. T. Ng and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. In *Proceedings of the 20th VLDB Conference*, pages 144–155, Santiago, Chile, 1994.

[17] Greg Pass, Ramin Zabih, and Justin Miller. Comparing images using color coherence vectors. In *Proceedings of ACM Multimedia 96*, pages 65–73, Boston MA USA, 1996.

[18] A. Pentland, R. Picard, and S. Sclaroff. Photobook: Tools for Content-based Manipulation of Image Databases. In *Proceedings of the SPIE Conference on Storage and Retrieval of Image and Video Databases II*, pages 34–47, 1994.

[19] Rosalind Picard. A society of models for video and image libraries. Technical Report 360, MIT Media Laboratory Perceptual Computing, 1996.

[20] G. Sheikholeslami, S. Chatterjee, and A. Zhang. NeuroMerge: An Approach for Merging Heterogeneous Features in Content-based Image Retrieval Systems. In *Proceedings of the 1998 International Workshop on Multi-media Database Management Systems*, Dayton, Ohio, August 1998.

[21] G. Sheikholeslami and A. Zhang. An Approach to Clustering Large Visual Databases Using Wavelet Transform. In *Proceedings of the SPIE Conference on Visual Data Exploration and Analysis IV*, San Jose, February 1997.

[22] G. Sheikholeslami, A. Zhang, and L. Bian. Geographical Data Classification and Retrieval. In *Proceedings of the 5th ACM International Workshop on Geographic Information Systems*, pages 58–61, Las Vegas, Nevada, November 1997.

[23] J. R. Smith and S. Chang. Transform Features For Texture Classification and Discrimination in Large Image Databases. In *Proceedings of the IEEE International Conference on Image Processing*, pages 407–411, 1994.

[24] John R. Smith and Shih-Fu Chang. Visualseek: a fully automated content-based image query system. In *Proceedings of ACM Multimedia 96*, pages 87–98, Boston MA USA, 1996.

[25] W.D. Stromberg and T.G. Farr. A Fourier-based textural feature extraction procedure. *IEEE Transactions on Geoscience and Remote Sensing*, 24(5):722–732, 1986.

[26] J. Wang, W. Yang, and R. Acharya. Color Clustering Techniques for Color-Content-Based Image Retrieval. In *the Fourth IEEE International Conference on Multimedia Computing and Systems (ICMCS'97)*, pages 442–449, Ottawa, Canada, June 1997.

[27] Wei Wang, Jiong Yang, and Richard Muntz. STING: A statistical information grid approach to spatial data mining. In *Proceedings of the 23rd VLDB Conference*, pages 186–195, Athens, Greece, 1997.

[28] D.A. White and R. Jain. Algorithms and strategies for similarity retrieval. Technical Report VCL-96-101, Visual Computing Laboratory, University of California, San Diego, 1996.

[29] A. Zhang, B. Cheng, and R. 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.

[30] Jacek M. Zurada. *Introduction to Artificial Neural Systems*. West Publishing Company, 1992.