Multi-Mode Indices for Effective Image Retrieval in Multimedia Systems

Guang-Ho Cha
Department of Multimedia Engineering
Tongmyong University of Information Technology
Pusan 608-711, South Korea
ghcha@tmic.tit.ac.kr

Chin-Wan Chung
Department of Computer Science
Korea Advanced Institute of Science and Technology
Taejon 305-701, South Korea
chungcw@ngis.kaist.ac.kr

Abstract

This paper presents a multi-mode indexing scheme for effective content-based image retrieval. Three types of indices are identified: visual indices for quantifiable visual information, semantic indices for non-quantifiable semantic information, keywords indices for keywords or free text. The underlying index structures are the HG-tree and the signature file. The HG-tree is one of the most promising multidimensional point index structures and the signature file is best known for handling keywords. The multi-mode indexing scheme combines and extends the HG-tree, the signature file, and the hashing technique to support a wide range of user queries in multimedia information systems. Experiments have been carried out on an image repository to demonstrate the effectiveness of the proposed content model and indexing scheme.

1. Introduction

Technology advances and application development in the area of multimedia information systems have been rapidly increasing in recent years. An important issue to be considered in the design of a multimedia information system is the content-based retrieval of multimedia data, which helps users to retrieve the desired information based on the contents of multimedia data. In such systems, multimedia data are analyzed so that the descriptions of their content, i.e., *metadata*, can be extracted and stored in the multimedia databases together with the original raw data. These descriptions are then used to search the multimedia database and to determine which multimedia data satisfy user's query selection criteria. In this paper, we focus our attention on the retrieval of images from multimedia systems.

The effectiveness of content-based image retrieval in multimedia systems depends largely on the following parameters: the contents to describe images, the representations of image contents, the types of image queries, the search strategies to process queries, and the indices to expedite the search.

The selection of the contents to describe the image is application-dependent. For example, some applications require visual information of the image (e.g., "Find images that have approximately 30% red and 15% blue colors) and some other applications need semantic (i.e., not quantifiable but perceptible by human) information (e.g., "Find images with beautiful scenery"). As more information about the image contents are specified and stored, more accurate query results might be acquired because there exist more information to discriminate among the images.

To decide the representation of image contents, the space efficiency and the search efficiency of the representation must be considered. The representations could be vectors (or tuples), strings, trees, graphs, and so on. The representations also may be affected by the application domain.

Unlike traditional database systems, the types of queries based on visual information expected in multimedia systems are largely based on similarity of the images. For example, "Find all images that are similar to a given image within some tolerance" (range query), or "Find five images that are most similar to a given montage" (nearest neighbor query). The exact-match queries and partial-match queries fall within the range queries. In addition, the queries based on some semantic attributes, keywords, or free text should also be supported to provide more effective image retrieval. Thus, the content-based image retrieval system should support efficiently not only the range and nearest neighbor queries but also the semantic and keyword-based queries to respond to a variety types of user queries.

To support efficient content-based retrieval, considering how to build indices that facilitate such a retrieval is inevitable. Indexing tabular data for exact-match search or range search in traditional databases is a well-understood problem, and the index structures like B-tree family [6] provide efficient access mechanisms. However, they are not likely to provide enough information to deal with complex image contents. Also a one-dimensional B-tree node does not usually reflect to an *n*-dimensional domain space of the image content, where *n* is the number of image features to index, and hence the representation is not particularly conducive to the *n*-dimensional image query. In addition, B-trees may not be appropriate to the similarity searching for multimedia content. For queries in which the similarity is defined as a distance metric in multidimensional feature spaces, the indexing involves clustering of objects in the multidimensional space and indexable representations of the clusters. Therefore, the traditional index structures such as B-trees are not appropriate for image data. Index structures to provide fast accesses in multidimensional feature space must be provided. Keyword-based or text-based retrievals can be managed with conventional information retrieval methods such as signature files [7, 11, 27, 32].

In summary, to support a wide range of queries the content description of multimedia data should comprise plenty of useful information which may be represented by various semantic attributes, keywords, and visual features. Moreover, appropriate index structures to index indexable image contents should be developed for efficient retrieval. These are the topics to be dealt with in this paper.

2. Images and Queries

This section describes the content representations of the image and the types of queries that are dealt with in our prototype content-based image retrieval system.

2.1. Description of an Image

An image object *I* consists of a *body B* and *a header H*. The body is a binary bitmap having a specific format such as JPG, GIF, BMP, and so on. The header is a metadata that describes the content of an image. We model the header as a triplet $H = (A_v, A_s, A_k)$:

- A_v is a set of *visual feature values*,
- A_s is a set of *semantic feature values*,
- A_k is a set of *keywords*.

A_v and A_s are represented as the fixed-sized tuples of visual and semantic attributes, respectively. A_k is represented as a variable-sized set of keywords. A_v includes the visual features which can be extracted automatically by the image interpretation subsystem. The visual features may be colors, textures, shapes, and so on. A_s includes the semantic information (i.e., non-quantifiable) of the image that should be extracted manually by the interpretation of human intermediaries. For example we can enumerate the following semantic attributes of an image:

- type: painting, scenery, portrait, and so on,
- subject: mountain, sea, animal, flower, architecture, and so on,
- title: title of image,
- perspective: aerial, ground, or close-up,
- orientation: horizontal or vertical,
- date: date when the picture is shot.

Keywords give the gist of an image. They are words or sequences of words which describes the characteristics of the image that can not be represented with simple common attributes. The maximum number of keywords allowed per image is a system parameter. Figure 1 shows an example header information that can be inserted into an image database.

A_v	*visual features*	dominant colors and textures
A_s	*type*	scenery
	subject	sea
	title	vista1
	perspective	ground
	orientation	horizontal
	date	10-17-97
A_k	*keywords*	wave, sunny, bridge

Figure 1. Sample image header

2.2. Description of a Query

The power of retrieval in multimedia systems must be increased if it can accommodate various types of queries. A *user query* is the specification of a header that closely corresponds to the information known about the image. Users can query the image databases based on the semantic attributes, keywords, and visual features. Some attribute values may be omitted, or may be given a specific values or range of values. Keywords are specified by providing a list of words that describe the image. Visual features are given by an example image, user-sketched drawings, or selected colors and texture patterns.

To summarize the different types of queries, we have the following.

- *Exact match queries* that specify a single attribute value for each possible attribute;
- *Range queries* that either explicitly specify a range of values for some of the attributes, as in $(10\% \le \text{red} \le 30\%) \wedge (30\% \le \text{green} \le 40\%) \wedge (80\% \le \text{blue} \le 90\%)$, or implicitly specify a range of values by leaving one or more attributes values unspecified.
- *Similarity* or *nearest neighbor queries* that give an example image or user-sketched drawing and require to find similar images to a given image.

3. Indices

In this section, three types of indices that index the corresponding indexable image features are proposed, and the multi-mode indexing scheme that integrates the indices are described.

3.1. Visual Indices

We assume that a set of *n* visual features have been extracted automatically or manually from each image. They may be dominant colors, textures, and shapes. When we represent a set of *n* visual features as an *n*-sized tuple, $A_v = (f_1, f_2, ..., f_n)$, where f_i, $1 \le i \le n$, is an individual visual feature, it can be mapped to a point in an *n*-dimensional visual feature space. We use the HG-tree [3] as our underlying index structure for organizing the visual feature based indices. It is an index structure to index point data in a multidimensional domain space. We select the HG-tree because it outperforms most of other multidimensional point index structures in a wide range of query performance comparisons [4].

In the HG-tree, all *n*-dimensional values are transformed into one-dimensional values using *space-filling curve*, and specifically *Hilbert curve* [15], before they can be used. Therefore all data points (i.e., tuples A_v's of *n* visual features) are represented by locations on the Hilbert curve and there is no need to consider the *n* dimensionality of the domain space. This makes the index creation and search algorithms simple.

A *space-filling curve* is a mapping that maps the unit interval onto the *n*-dimensional unit hyper-rectangle continuously. While there are other space-filling curves such as the Peano curve (also known as the Z curve) [23] and the Gray-code curve [10], it was shown that the Hilbert curve achieves better clustering than the others [12, 16]. The desirable features of the Hilbert curve are that the points close on the Hilbert curve are close in the domain space, and the points close in the domain space are likely to be close on the Hilbert curve.

The basic Hilbert curve on a 2×2 grid, denoted by H_1, and the Hilbert curve of order 2, denoted by H_2, are shown in Figure 2. The location (0,0) on the H_2 curve has a Hilbert value of 0, while the location (1,1) has a Hilbert value of 2. The Hilbert curve can be generalized for any higher dimensionality.

We could use other index structures for visual feature based indices. In fact, many other image retrieval systems have used some other index structures. The QBIC system [13] adopted the R*-tree [1] as an index structure. Petrakis and Faloutsos [24] used R-tree [14]. Mehrotra and Gary [20] used the K-D-B-tree [26]. The systems, CAFIIR [29] and STAR [30] and Zhang and Zhong [31] employed the iconic index tree based on the Self-Organizing Map (SOM) [17]. Each of these methods has not only its own advantages, but also has some limitations.

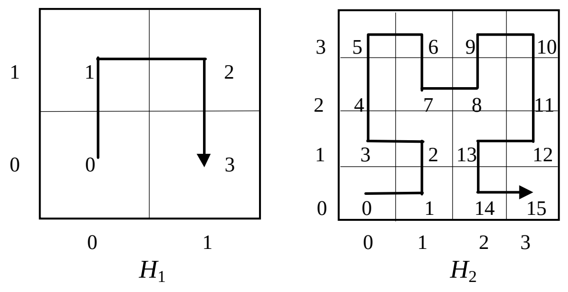

Figure 2. Hilbert Curves of order 1 and 2

Compared with the other index structures, the performance of the multidimensional spatial index structures such as R-tree and R*-tree degenerates drastically with an increase of the dimensionality of the underlying feature space, because their *fanout* decreases in inversely proportional to the dimensionality. Fanout gives the number of entries expected within an index node. All current spatial index structures suffer from this *dimensionality curse*. The improvement of fanout is a very important factor for the performance of the index structure.

The iconic index trees based on the SOM simplify the multidimensional problem by converting it to a one-dimensional clustering problem based on similarities. The major problem of these kind of index structures is that they are *static* methods. Usually the iconic index is constructed against a large data set which can represent the statistics of the data. However, if the system is to include other classes of data that change the index node characteristics, then the system must be trained again in order to provide effective clustering. Therefore, updates due to insertions and deletions are accumulated and actually performed when the amount of updates is up to a threshold. Another problem of the iconic index trees is that they are constructed only for nearest neighbor queries. Thus, it is difficult to process range queries. In fact, most of the index structures designed only for nearest neighbor queries have these problems in common. For example, the optimistic VP (vantage point)-tree [5] and the GNAT (Geometric Near-neighbor Access Tree) [2] are such kind of index structures. They precalculate some nearest neighbors of points, store the distances in a tree or graph, and use the precalculated information for a more efficient nearest neighbor search. Therefore, they have benefit in the nearest neighbor search time, but have disadvantage in update operations (insert and delete). In other words, they are static. As in the iconic index structures, they also have difficulty in processing the range queries.

The HG-tree, which is one of multidimensional point index structures, avoid all aforementioned problems. The performance degradation of the HG-tree due to the increase of the dimensionality is far less than that of the spatial index structures, because it represents each directory region covered by the data set by using only two Hilbert values. In addition, the HG-tree is completely dynamic, i.e., it supports arbitrary insertions and deletions of objects without any global reorganizations and without any loss of performance.

3.2. Semantic Indices

A fixed-set of tuple A_s to describe the semantic features is represented by $(s_1, s_2, \dots, s_k)$ such that $s_i \in D_i$ for $1 \le i \le k$. Where D_i for $1 \le i \le k$ is a domain space of the i-th semantic attribute s_i. We have k hash functions H_i: $D_i \rightarrow W_i$, for $1 \le i \le k$, where $W_i = \{0, 1, 2, ..., 2^{imax} - 1\}$, and $2^{imax} - 1$ is the maximum allowable hashed value. We construct the semantic indices with this hashed tuple $H_s = (H_1(s_1), H_2(s_2), ..., H_k(s_k))$. Unlike other image/video retrieval systems such as QBIC and Chabot [22] which use B-tree [6] as their index structure to index semantic or tabular data, we use the HG-tree as our underlying index structure for semantic indices. The B-tree is a primary indexing scheme. Thus indices are constructed on the primary attribute. If it is required to provide a fast access on other attributes, another indices constructed on that attributes are also needed. Instead of constructing multiple single-attribute (MSA) B-tree indices, we construct a single multi-attribute (SMA) index using the HG-tree.

There are some important advantages of using SMA index as compared to MSA index. First, the clustering of index pages and data pages on disk due to using single index can dramatically reduce the number of I/O operations needed for database accesses. Second, when new records are inserted into or deleted from a database, SMA index organization needs only single update for its index. MSA index, in contrast, require multiple updates since there are multiple indices. Therefore, maintaining the consistency of indices in SMA index organization is simpler than that in MSA index organization.

3.3. Keyword Indices

The *signature file* has proved to be a convenient indexing technique for text and multiattribute retrieval [7, 8, 11, 18, 25, 27, 32]. Multidimensional index structures such as K-D-B tree, grid file [21], are not appropriate for indexing text data represented by keywords. Because they assume the dimensionality of the domain space, which is the number of keywords given in user's query in the case of keyword query, is small and constant. However, the number of keywords given by users to query an image database are variable. Moreover, most of the multidimensional index structures suffer from the dimensionality curse. Therefore, we chose the signature file technique as our indexing method for keywords.

The main idea of the *signature file* is to derive properties of data objects, called *signatures*, and store them in a separate file. A collection of the derived signatures is called signature file. Signatures are hash-coded binary words of fixed length. In general, all bits of the signature are cleared to *null*, then a hash function is applied to the object's values to determine which bits are set to *one*. A lot of research has been done on the improvement of the performance of a signature file [18, 25, 27]. However, most of the researches have been performed for *static* environments where update operations are rarely occurred. Our index design scheme requires a dynamic data environment, which means that the signature file must be allowed to grow and shrink. The two representative dynamic signature files are S-tree [8] and Quick filter [32]. The main idea of the S-tree is to group adjacent signatures in pages and build a B-tree on top of them to provide direct access to the leaf signature pages. The major problem of the S-tree is that the performance is degenerated as the query signature weight becomes lower. The number of 1's in a signature is called the *signature weight*. In the Quick filter, a signature file is partitioned by a hash function and the partitions are organized by linear hashing. Therefore, it is appropriate for the dynamic environment where updates are occurred frequently and results in good performance in the queries with high signature weights. However, if the distribution of signatures is nonuniform, then similar signatures are frequently generated and therefore the overflow rate increases and the storage utilization decreases. These degenerate the performance of the Quick filter. We improve the disadvantages of the existing dynamic signature methods.

First, we divide a signature into s *frames* and select c frames from a total of s frames using one hash function h_1 as in the frame-sliced signature file [19] to tackle the problem caused by the light weight signatures. To make up the *word signature* (i.e., the signature corresponding one keyword) m bits are set to "1" in the selected c frames using the second hash function h_2. The *frame signature* is constructed by superimposing the parts belong to the corresponding frame of word signatures. At last, the *image signature* describing the image content specified by a set of keywords is constructed by concatenating the frame signatures.

Second, to solve the problem of high overflow rate and low storage utilization, we adopt the HG-tree as the underlying index structure instead of using the linear hashing scheme as in the Quick filter. The HG-tree is completely dynamic and robust in a variety of data distributions. The average storage utilization of the HG-tree is over 80% in most of the cases and the worst-case storage utilization is guaranteed to be more than 66.7% (2/3) [4].

When we construct the HG-tree with image signatures, the values of the image signatures are translated into the Hilbert values and they are inserted in the order of Hilbert values. This translation procedure corresponds to the mapping a point in *s*-dimensional space into a point in a linear Hilbert curve. The number of frames, *s*, constituting an image signature determines the dimensionality of the keyword domain space and the image signature used in a node of the HG-tree determines a directory region in the domain space. The reason of using Hilbert mapping instead of simply concatenating *s* frames is to place the similar signatures into the same disk pages.

Figure 3 and Figure 4 show the binary ordering and the Hilbert ordering in a 4×4 domain space. In general, when the similar signatures are placed in the same page, they have a high qualifying probability once the page has been designated by the search procedure. Consider the binary ordering in Figure 3. Although the two points S_1 and S_2 are far apart, i.e., the signatures of S_1 and S_2 are quite different, the probability of placing the two signatures in the same page is very high because their binary numbers are adjacent. This may result in many random accesses on disk. On the other hand, the points close on the Hilbert curve are also close in the domain space as shown in Figure 4. In other words, the image signatures placed in the same page are similar. This characteristic may achieve a better clustering of similar signatures and may avoid expensive random disk accesses.

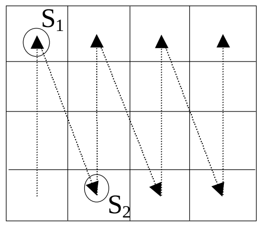

Figure 3. binary ordering

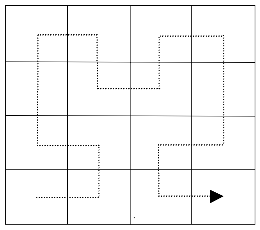
Figure 4. Hilbert ordering

3.4. Multi-Mode Indexing Scheme

The multi-mode indexing scheme that integrates the three types of indices is shown in Figure 5. User queries are transformed into three types of feature values during query processing. These values are evaluated and searched through the corresponding indices. Finally, Object IDentifiers, OIDs, are acquired and intersected to answer the user query.

4. Image Retrieval System

A content-based image retrieval system has been implemented. The architecture of the system is shown in Figure 6. The system consists of the components described in the above section. The system supports semantic attribute-based, visual feature-based, and keyword-based retrievals.

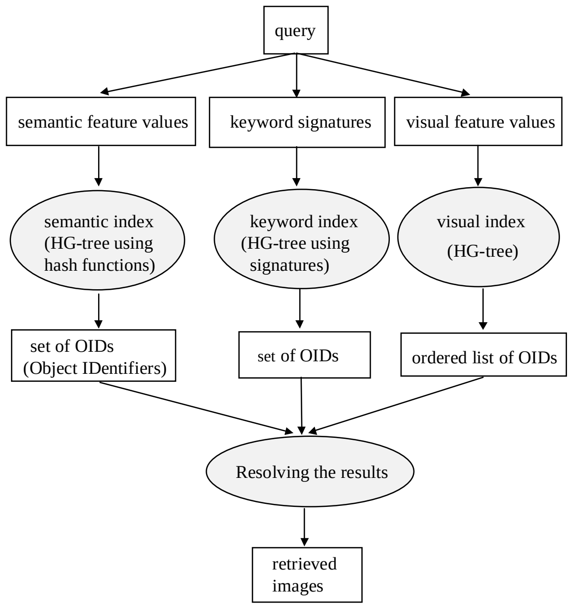

Figure 5. Multi-Mode Indexing Scheme

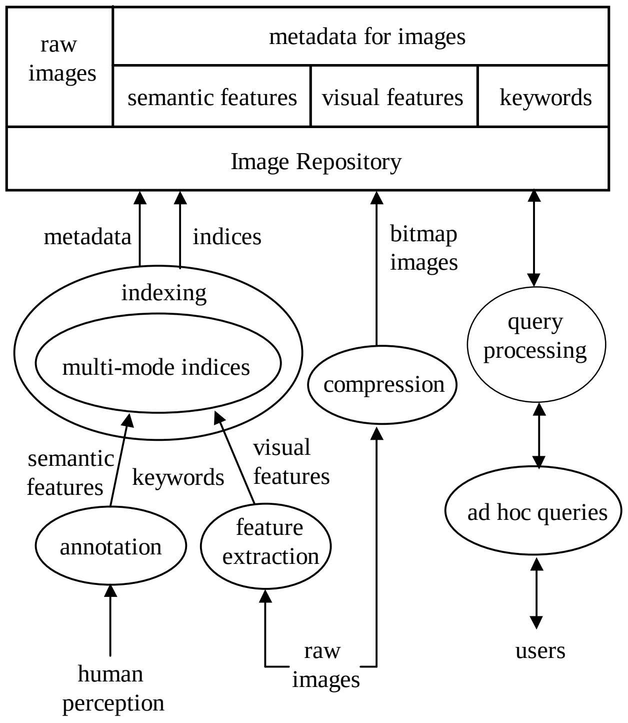

Figure 6. Architecture of content-based image retrieval system

4.1. Image Repository

To test the effectiveness of multi-mode indexing scheme for content-based image retrieval, we have constructed an image repository which has a 1,064 images. The images are 256-color bitmaps with a variety of contents. In a repository, images are stored together with three kinds of attributes, i.e., semantic attributes, visual attributes, and keywords, so that images can be retrieved from these descriptions.

4.2. Visual Feature Extraction

To acquire visual features that characterize images we used statistical color moments of the histogram of the image because color has excellent discrimination power in image retrieval system. Since most histogram bins of an image are sparsely populated and only a small number of bins have the majority of pixel counts, we used only the largest 32 bins (in terms of pixel counts) as the representative bins of the histogram. We used first two moments of the histogram as descriptors of an image:

$$\mu_i = \frac{1}{n}\sum_{j=1}^{k} f_{ij}x_{ij}\,,\sigma_i = \left(\frac{1}{n}\sum_{j=1}^{k} f_{ij}\left(x_{ij}-\mu_i\right)^2\right)^{\frac{1}{2}},$$
$$i = 1, 2, 3$$

where x_{ij} is the value of color component of the j-th bin, f_{ij} is the frequency of x_{ij}, k is the number of total bins, i.e. 32, and n is the total number of pixels in the histogram. Since we used the RGB color model, the i-th color component corresponds to one of red, green, and blue. The first moment, μ_i, defines the average intensity of each color component. The second moment, σ_i, is a measure of contrast that can be used to establish descriptors of relative smoothness.

Measures of global color statistics using only histograms suffer from the limitation that they carry no information regarding the relative position of pixels. To overcome this limitation to some extent, we divided the image into 4 sub-areas and computed 2 moments for each sub-area, resulting in a 24 (= 2 moments $\times$ 3 color components $\times$ 4 sub-areas) features for an image.

Using these 24-dimensional feature vectors, we estimate the difference, *diff*(S, T), between two color histograms S and T as follows:

$$diff\ (S,T) = \sum_{k=1}^{4}\left(\sum_{i=1}^{3}\left(\left|\mu_{ik}(S)-\mu_{ik}(T)\right|+\left|\sigma_{ik}(S)-\sigma_{ik}(T)\right|\right)\right)$$

4.3. Sample *k*-Nearest Neighbor Queries

Figure 7 and Figure 8 show respectively the results of two sample 12-nearest neighbor queries: (a) Query 1: “Find 12 images most similar to a given image tigera4.bmp” (b) Query 2: “Find 12 images most similar to a given image tigera4.bmp and whose semantic attribute *subject* has the value of *animal*”.

The image on the upper-left corner in Figure 7 and Figure 8 is a given query image ‘tigera4.bmp’ and 12 most similar images are retrieved in left-to-right and top-to-down sequence. In Figure 7, the query image tigera4.bmp is, of course, the most similar image, bench1.bmp is the second similar image, and detail14.bmp is the 12-th similar image. The result of Figure 7 is obtained only using color visual features. Obviously, all images retrieved from a real image database have similar color properties to the given query image. On the other hand, the semantic feature, *subject* = *animal* is used together with the color visual feature in the processing of Query 2. As shown in Figure 8, the more the features are specified in the query, the higher the *selectivity*, i.e., the ratio of the expected number of answers over the total number of data in the database, is increased. However, as more features are specified in the query, the search cost increases higher.

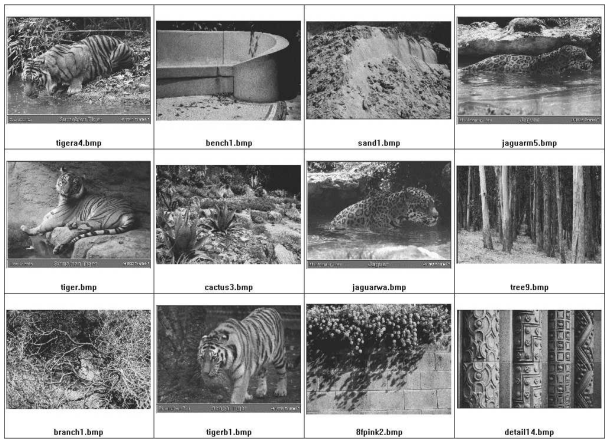

Figure 7. 12-nearest neighbor query using only color visual features

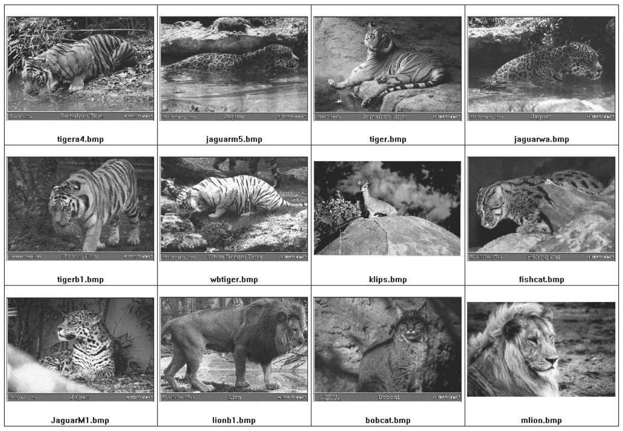

Figure 8. 12-nearest neighbor query using color visual features and semantic features

5. Conclusions

A multi-mode indexing scheme for effective image retrieval in multimedia information systems is presented in this paper. An image object consists of a body and a header. The body is a binary bitmap. The header consists of three types of metadata which characterize image content, i.e., a set of visual feature values, a set of semantic feature values, and a set of keywords. Corresponding to three types of metadata, three types of indexing methods are developed and they are based on the HG-tree, hashing functions, and the dynamic signature techniques. Using the HG-tree and the hashing functions, the proposed indexing scheme efficiently supports the multidimensional point data, i.e., visual features and semantic features. In addition, the HG-tree together with the signature technique indexes keywords dynamically and efficiently. The multi-mode indexing scheme support effectively most of the query types which can be met in content-based image retrievals.

In the future, we plan to consider how to effectively model the image content. In our current implementation, the three types of image content have no certain structures. They are simply sets. This makes easy to query for novice and infrequent users, because they need not to know the specific schema structures in the image repository. However, the database administrators or frequent users may want to query in precisely manner to restrict the search scope. Moreover, there may be some inherent relationships among the identified features. Object-oriented data model or some graph-based models may be candidates.

References

[1] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, “The R*-tree: An efficient and robust access method for points and rectangles,” *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 322-331, 1990.

[2] S. Brin, “Near Neighbor Search in Large Metric Spaces,” *Proceedings of the 21st VLDB Conference*, pp. 574-584, 1995.

[3] G.-H. Cha and C.-W. Chung, “HG-tree: An Index Structure for Multimedia Databases,” *Proceedings of the 3rd IEEE International Conference on Multimedia Computing and Systems*, pp. 499-452, June 1996.

[4] G.-H. Cha and C.-W. Chung, “A New Indexing Scheme for Content-Based Image Retrieval,” *Multimedia Tools and Applications*, Vol. 6, No. 3, May 1998. (to appear).

[5] T.-C. Chiueh, “Content-Based Image Indexing,” *Proceedings of the 20th VLDB Conference*, pp. 582-593, 1994.

[6] D. Comer, “The Ubiquitous B-tree,” *ACM Computing Surveys*, Vol. 11, No. 2, pp. 121-137, 1979

[7] S. Christodoulakis and C. Faloutsos, “Signature files: An access method for documents and its analytical performance evaluation,” *ACM Transactions on Office Information Systems*, Vol. 2, No. 4, pp. 267-288. October 1984.

[8] U. Deppisch, “S-Tree: A Dynamic Balanced Signature Index for Office Retrieval,” *Proceedings of the ACM Conferences on Research and Development in Information Retrieval*, pp. 77-87, September 1986.

[9] R. Fagin, N. Nievergelt, N. Pippenger, and H. R. Strong, “Entendible hashing - A fast access method for dynamic files,” *ACM Transactions on Database Systems*, Vol. 4, No. 3, pp. 315-344, September 1979.

[10] C. Faloutsos, “Gray codes for partial match and range queries,” *IEEE Transactions on Software Engineering*, Vol. 14, No. 10, pp. 1381-1393, 1988.

[11] C. Faloutsos and S. Christodoulakis, “Description and performance analysis of signature file methods for office filing,” *ACM Transactions of Office Information Systems*, Vol. 5, No. 3, pp. 237-257, July 1987.

[12] C. Faloutsos and S. Roseman, “Fractals for secondary key retrieval,” *Proceedings of the 8th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 247-252, 1989.

[13] M. Flickner et al., “Query by Image and Video Content: The QBIC System,” *IEEE Computer*, pp. 23-32, September 1995.

[14] A. Guttman, “R-Trees: A Dynamic Index Structure for Spatial Searching,” *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 47-57, 1984.

[15] D. Hilbert, “Uber die stetige Abbildung einer Linie auf ein Flachenstuck,” *Math. Annalen*, Vol. 38, 1891.

[16] H.V. Jagadish, “Linear Clustering of Objects with Multiple Attributes,” *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 332-342, 1990.

[17] T. Kohonen, “The Self-Organizing Map,” *Proceedings of the IEEE*, Vol. 78, No. 9, pp. 1464-1480, September 1990.

[18] D. L. Lee and C.-W. Leng, “Partitioned Signature Files: Design Issues and Performance Evaluation,” *ACM Transactions on Office Information Systems*, Vol. 7, No.2, pp. 158-180, 1989.

[19] Z. Lin and C. Faloutsos, “Frame-Sliced Signature Files,” *IEEE Transactions on Knowledge and Data Engineering*, Vol. 4, No. 3, pp. 281-289, 1992.

[20] R. Mehrotra and J.E. Gary, “Similar-Shape Retrieval in Shape Data Management,” *IEEE Computer*, pp. 57-62, September 1995.

[21] J. Nievergelt, H. Hinterberger, and K.C. Sevcik, “The grid file: an adaptable, symmetric multikey file structure,” *ACM Transactions on Database Systems*, Vol. 9, No.1, pp. 38-71, 1984.

[22] V.E. Ogle and M. Stonebraker, “Chabot: Retrieval from a Relational Database of Images,” *IEEE Computer*, pp. 40-48, September 1995.

[23] G. Peano, “Sur une courbe qui remplit toute une aire plane,” *Math. Annalen*, Vol. 36, pp. 157-160, 1890.

[24] E.G.M. Petrakis and C. Faloutsos, “Similar Searching in Large Image Databases,” Technical Report CS-TR-3388, University of Maryland, 1994.

[25] C. S. Roberts, “Partial-Match Retrieval via the Method of Superimposed Codes,” *Proceedings of the IEEE*, pp. 1624-1642, Vol. 67, No. 12, December 1979.

[26] J. T. Robinson, “The K-D-B-Tree: A Search Structure for Large Multidimensional Dynamic Indexes,” *Proceedings of*

the ACM SIGMOD International Conference on Management of Data, pp. 10-18, 1981.

[27] R. Sacks-Davis, A. Kent, and K. Ramamohanarao, “Multikey Access Methods Based on Superimposed Coding Techniques,” *ACM Transactions on Database Systems*, pp. 655-696, Vol. 12, No. 4, December 1987.

[28] K.-Y. Whang, S.-W. Kim, and G. Wiederhold, “Dynamic Maintenance of Data Distribution for Selectivity Estimation,” *VLDB Journal*, Vol. 3, No. 1, pp. 29-51, 1994.

[29] J.K. Wu, Y.H. Ang, P. Lam, H.H. Loh, and A.D. Narasimhalu, “Inference and retrieval of facial images,” *Multimedia Systems*, Vol. 2, No. 1, pp. 1-14, 1994.

[30] J.K. Wu, A.D. Narasimhalu, B.M. Mehtre, C.P. Lam, and Y.J. Gao, “CORE: a content-based retrieval engine for multimedia information systems,” *Multimedia Systems*, Vol. 3, No. 1, pp. 25-41, 1995.

[31] H.J. Zhang and D. Zhong, “A Scheme for Visual Feature based Image Indexing,” *Proceedings of the IS&T/SPIE Conference on Storage and Retrieval for Image and Video Databases III*, pp. 36-46, 1995.

[32] P. Zezula, F. Rabitti, and P. Tiberio, “Dynamic Partitioning of Signature Files,” *ACM Transactions of Information Systems*, pp. 336-369, Vol. 9, No. 4, October 1991.