World Wide Web (2006) 9: 343–364
DOI 10.1007/s11280-006-8560-4

Indexing and Integrating Multiple Features for WWW Images

Heng Tao Shen · Xiaofang Zhou · Bin Cui

Received: 12 March 2005 / Revised: 19 August 2005 /
Accepted: 3 February 2006 / Published online: 8 June 2006

Abstract In this paper, we present a novel indexing technique called *Multi-scale Similarity Indexing* (MSI) to index image's multi-features into a single one-dimensional structure. Both for text and visual feature spaces, the similarity between a point and a local partition's center in individual space is used as the indexing key, where similarity values in different features are distinguished by different scale. Then a single indexing tree can be built on these keys. Based on the property that relevant images have similar similarity values from the center of the same local partition in any feature space, certain number of irrelevant images can be fast pruned based on the triangle inequity on indexing keys. To remove the "dimensionality curse" existing in high dimensional structure, we propose a new technique called *Local Bit Stream* (LBS). LBS transforms image's text and visual feature representations into simple, uniform and effective bit stream (BS) representations based on local partition's center. Such BS representations are small in size and fast for comparison since only bit operation are involved. By comparing common bits existing in two BSs, most of irrelevant images can be immediately filtered. To effectively integrate multi-features, we also investigated the following evidence combination techniques—*Certainty Factor, Dempster Shafer Theory, Compound Probability,* and *Linear Combination*. Our extensive experiment showed that single one-dimensional index on multi-features improves multi-indices on multi-features greatly. Our LBS method outperforms sequential scan on high dimensional space by an order of magnitude. And Certainty Factor and Dempster Shafer Theory perform best in combining multiple similarities from corresponding multiple features.

Categories and Subject Descriptors H.3.1 [Content Analysis and Indexing]: Indexing methods.

H. T. Shen (✉) · X. Zhou · B. Cui
School of Information Technology and Electrical Engineering,
The University of Queensland, Brisbane QLD 4072, Australia
e-mail: shenht@itee.uq.edu.au

X. Zhou
e-mail: zxf@itee.uq.edu.au

Keywords indexing · clustering · WWW · multi-features · image retrieval

1. Introduction

WWW provides a super big pool for interesting images. Recently, WWW image retrieval has been a very challenging research area. Such images are typically described by both high-level (text) and low-level (visual) features. To retrieve relevant images from large image database, two issues are essential: effectiveness and efficiency. However, most known research results [15] are on retrieval effectiveness. There is no clearly known research achievement on how to index this particular type of large image database described by multi-features for efficient retrieval, especially on indexing completely different representations: text and visual features. Current method is to build one structure for every single feature. Given an image query, it has to access all individual structures and integrate results from each index to get the final results. Furthermore, these known indexing structures suffer from "dimensionality curse." When the dimensionality of data space reaches 20 or greater, indexing techniques fail to outperform sequential scan [25] for nearest neighbor search.

In this paper, we propose a new method called Multi-scale Similarity Indexing (MSI) that can index WWW images' multi-features in a single structure. MSI exhibits a means of indexing different features in different representations in a single tree, such as image text and visual features, where text feature is in Weighted Lexical Chain model [18] or others, and visual feature is in standard high dimensional data space. MSI first partitions each feature space into clusters. Then the similarity of each image in each feature space to its corresponding cluster's center is computed as the indexing key. By a simple mapping function, we can keep the keys for each cluster in different feature space distinct in different scale level. Thus a standard B+ tree can be easily built on these indexing keys.

However, like other existing indexing technique, MSI also suffers from "dimensionality curse." To release MSI from such curse, we propose a novel technique called Local Bit Stream (LBS). LBS exhibits a way to transform the high dimensional feature representation (big size) into a uniform, accurate and compact representation (small size). Given clusters in each feature space, LBS encodes each point in different feature space into a uniformly dimensional bit stream (SB). The BS of a point in a feature space is generated by comparing this point and its cluster's center. Thus such transformation is localized at cluster level. The effectiveness of LBS depends on how to generate the BS for each feature point. Due to the completely different nature of text and visual features, both are encoded in different schemes. We present different encoding strategies for text and visual features. However, both encoding strategies can produce the same uniformly BS representations for text and visual feature points. BS is an approximate representation of original data. It's compact, much smaller in size, and accurate for similarity measures. Furthermore, BS comparisons involve bit operations only. Thus it is much faster in terms of efficiency.

Since multiple features are considered in the similarity measurement, in this paper we also investigated several integration methods to improve the retrieval effectiveness. Four techniques are tested in our experiments: *Certainty Factor*, *Dempster Shafer Theory, Compound Probability,* and *Linear Combination.* Each

method combines multi-similarities from multi-features by considering different factors to handle the imprecision.

We implement our indexing techniques on top of MYSQL server. An extensive performance study is conducted to evaluate our methods. Our results show that single indexing structure is supreme to multi-indexing structures, and LBS breaks the dimensionality curse by improving the response time faster than sequential scan and iDistance [26] by an order of magnitude without sacrificing the retrieval precision. Moreover, Certainty Factor and Dempster Shafer Theory outperform other evidence combination techniques.

A preliminary version of this paper appeared in [19]. There we addressed LBS. In this paper, we extend the work in several ways. First, we study various evidence combination techniques to improve the retrieval accuracy. Second, we investigate the effect of dimensionality reduction on the performance of LBS. Third, more experiments, including results on combination methods and dimensionality reduction method, are reported to further confirm the effectiveness of our proposed methods.

The rest of paper is organized as follows. In Section 2, we review some related works. In Section 3, some preliminary work on image features and similarity measures are introduced. In Section 4, we present the single one-dimensional indexing structure—MSI, and in Section 5 we propose the LBS and its encoding schemes. In Section 6, we examine various evidence integration techniques. An extensive performance study is presented in Section 7. Finally, we conclude our paper in Section 8.

2. Related work

Our related works cover several research areas: WWW image retrieval, evidence integration, high dimensional indexing, and multi-feature query processing.

Several WWW image retrieval systems have been proposed in literature. Existing known systems, such as AMORE [12], ImageRover [17], and WebSeek [22], allow the WWW image retrieval on combination of multi-features, like keywords, color, shape and texture. Recently, the high-level features of WWW images were explored by a Weight ChainNet model [18] since low level features cannot represent the high level semantic meanings for WWW images. More recently, the textual and Hyperlink information are extracted from blocks of Web pages to improve the accuracy [3]. And relevance feedback techniques are also applied in WWW image retrieval [6, 27]. However, most of systems focus on retrieval accuracy only.

To integrate multi-features together, most of systems used linear combination by assuming that text and visual features are linearly important. Recently, Dempster Shafer Theory, one technique to handle uncertainty, has been also employed on indexing of face retrieval on the web [2]. In this paper, we examine more techniques, including Certainty Factor and Compound probability.

Recently, Nearest Neighbor (NN) search in high dimensional spaces has been a very active research topic. Several indexing structures [5, 8, 14, 16, 25, 26] have been proposed. However, all these techniques are for indexing an individual feature space purpose, and they all suffer from known "dimensionality curse." Their performances degrade rapidly as dimensionality increases. As dimensionality reaches high (>20), they even fail to outperform sequential scan. Most existing systems build one index for every feature space. Given a query, each index has to be accessed. ImageRover

[17] tried to combine multi-features by first performing dimensionality reduction on each feature then used existing indexing structure to index *concatenated feature vector* from every reduced feature space. Anne et al. [13] applied non-linear neural network techniques with dimensionality reduction method, then used the similar way to index reduced multi-visual features by existing indexing structure. However, both have the following drawbacks. First the dimensionality curse still remains. Their techniques reduce the spaces into a level where the retrieval accuracy is reasonably affected. Image features spaces are typically in dimensionality of a range of tens to hundreds. For images with multi-features, it is usually not practical to reduce the total dimensionality of all reduced feature spaces to be less than 20 while remaining high retrieval accuracy. Second, there was no clear report on their indexing efficiency. Third, neural network is tedious and hard for training, especially for WWW image database with text features. In this paper, we aim to index multi text and visual features in a one-dimensional single index and leave the dimensionality curse to the past.

Another category of our related work is on processing multi-feature queries. Such problem appears obviously on multi-features images database. Given a query image, the typical steps are first to compute the similarity among the same feature space, then combine the score from all feature spaces, and finally rank them based on the final score. Some optimization jobs can be done to reduce the overall cost [7, 9]. We will not present multi-feature query processing problem in this paper, but on indexing issues.

3. Preliminary

In this section, we briefly present the features we used to describe the WWW images and respective similarity measures.

3.1. WWW image features

Without losing the generality, we use text feature and one visual feature as the descriptors of WWW images.

3.1.1. Text feature

Text descriptions of WWW image carry high-level semantic meanings. We choose a recently proposed representation model called Weighted ChainNet Model [18] as the text feature. Weighted ChainNet constructs a Lexical Chain (or sentence) network given the WWW image's surrounding text in its embedded web page, by assigning different weight for different type of Lexical Chain (LC). And there are six types of lexical chains were introduced: Title Lexical Chain Alt Lexical Chain, Page Lexical Chain, Sentence Lexical Chain, Reconstructed Sentence Lexical Chain, and Caption Lexical Chain. The first three types are constructed by image's title, image's alternate text, and web page's title, respectively, and the last three are constructed by image caption. To simplify the problem and illustration, here we summarize the chain network into a single weighted lexical chain by summing all the

weight in each type of lexical china for each word in the network. The following formula is used to compute the total weight for each word.

$$Word_{weight} = \sum_{i=1}^{6} Word^{i}_{weight}$$

Where $Word^{i}_{weight}$ is the weight of *Word* in type *i* lexical chain, and *i* ranges from 1 to 6.

Thus all the weighted words form a single *weighted lexical chain*, which is used as our WWW image's text feature. For simplicity, we denote image's text feature as *T*.

3.1.2. Visual feature

Wavelet transform is a useful tool in effectively generating compact representation that exploits the structure of visual features of images. By using wavelet sub band decomposition, and remain the most important sub bands (largest coefficients), we can get fixed size dimensional feature vectors independent of resolution and scaling. Wavelets produce the wavelet coefficients for an image as its description. And such coefficients construct a coarse overall approximation of image's visual feature. This approximation captures image's shape, texture and location information in a single signature. We use daubechies' wavelets [24] to generate WWW image's visual features. In this paper, we truncate the 64 most dominating coefficients as our image's visual feature. Thus our WWW image's visual feature is in 64-dimensional feature vector. For simplicity, we denote image's visual feature as *V*.

3.2. Image similarity measurements

For text feature, we employ the cosine formula as follows:

$$Sim^{text}(T_i, T_j) = \frac{T_i \bullet T_j}{\|T_i\| * \|T_j\|}$$

where T_i and T_j is image *i*'s and image *j*'s text feature, respectively.

For visual feature, the similarity between two images is computed as follows based on Manhattan Distance:

$$Sim^{visual}(V_i, V_j) = 1 - \frac{\left(\sum_{d=1}^{D} |V_{i.d\cdot} - V_{j.d}|\right)}{D}$$

where $V_{i.d}$ and $V_{i.j}$ is the *d*th dimensional value for image *i*'s and image *j*'s visual feature, respectively. *D* is the dimensionality of visual feature space.

4. Indexing multi-scale similarities

4.1. Building indexing structure

In this section, we present the one-dimensional single indexing technique for image's multi-features, called Multi-scale Similarity Indexing (MSI). MSI is mainly

inspired from the following observations. First, in the same cluster, relevant images have close similarities to the cluster's center. And this property is hold for both text feature space and visual feature space. Second, based on the similarities to the cluster center, images can be ordered within that cluster. Third, similarities are one-dimensional values. If we can map each image into corresponding similarity value and each cluster in each feature space can be scaled into different interval, a single one-dimensional index like B+-tree can be easily built on these similarities. Thus in MSI, high dimensional features spaces are transformed into one-dimensional space. Certain amount of irrelevant images can be fast pruned based on these one-dimensional values' comparisons.

To build MSI, we need first to cluster each feature space into partitions and compute their centers. Let's assume that there are m clusters in text feature space and n clusters in visual feature space. In text space, each cluster is assigned with a cluster ID from 1 to m, and similarly to visual feature space with cluster ID from 1 to n. Given an image with feature T and V, its indexing keys in different feature space are computed as follows:

$$key^{text} = T\_SCALE + i*C + Sim^{text}\left(T, O_i^T\right)$$
$$key^{visual} = V\_SCALE + j*C + Sim^{visual}\left(V, O_j^V\right)$$

where key^{text} and key^{visual} are the indexing keys, i and j are cluster Ids for its T and V in text feature space and visual feature space, with reference points O_i^T and O_j^V, respectively. The reference points are typically selected as the cluster centers. T_SCALE and V_SCALE are two constant scales with a large gap to distinguish text and visual spaces. C is a constant to stretch the similarities range so that features in different cluster have different range. Thus features in different clusters can be distinguished easily. For example, an image with feature T and V, T is in cluster i in text feature space, and V is in cluster j in visual feature space, then its two indexing keys will be transformed into the ranges $[T\_SCALE + i * C, T\_SCALE + (i+1) * C]$ and $[V\_SCALE + j * C, V\_SCALE + (j+1) * C]$, respectively.

A single B+-tree can be used to index the similarity keys for fast retrieval. And an additional auxiliary array is used to store the clusters centers and their minimum and maximum radii/similarity values that define the cluster's data space, where the minimum and maximum radii are used to facilitate searching. When there is only one feature, MSI is similar to iDistance [26], except the keys are computed based on the similarity, rather than distance values.

4.2. Query processing

Given a query image Q to search for the K top relevant images (K nearest neighbors), the searching algorithm works as follows. For each feature space, the query is first divided into two sub queries, Q_T and Q_V, respectively, where Q_T and Q_V are image's text and visual features. Then nearest neighbor searching is performed to get K_T and K_V top ranked image Ids from text and visual feature space such that the intersection of both set of Ids has at least K Ids in common. Evidence combination methods are then applied to compute the final list of results.

For each sub query, the searching starts with a query sphere by a *relatively* small radius R around Q_T and Q_V, respectively. To find the desired number of most

relevant images, the searching radius cannot be predetermined. Hence an iterative method that examines the *increasing larger* query sphere in each iteration has to be used. Searching in MSI begins with scanning the auxiliary array to determine which cluster whose data space overlaps with the searching sphere of Q_T and Q_V. This can be determined by the following triangle inequality property:

$$Sim^{text}(Q_T - O^T) \leq Sim^{text}(Q_T - P) + R \quad \text{or}$$
$$Sim^{visual}(Q_V - O^V) \leq Sim^{visual}(Q_V - P) + R$$

where P is a feature point in either text or visual feature space.

Figure 1 shows the searching spaces for two queries Q_1 and Q_2 corresponding to a cluster O which covers a space defined by its minimum and maximum radii. From the above triangle inequality property, for Q_1, cluster O can be directly pruned since the similarity between Q_1 and O is greater than cluster's maximum radius/similarity plus query searching radius R. The same situation occurs when the similarity between Q_1 and O is less than the cluster's minimum radius/similarity minus query searching radius R. And this pruning situation is common to both text and visual feature spaces.

On the other hand, if both query sphere and cluster's data space intersect, such as Q_2's searching sphere in Figure 1, range searching has to be performed in MSI. Given an image with Q_T and Q_V intersect with cluster O_i^T and O_j^V in text and visual space, respectively, their ranges searched in MSI are:

$$\begin{bmatrix} T\_SCALE + i^*C + Sim^{text}(Q_T, O_i^T) - R, \\ T\_SCALE + i^*C + Sim^{text}(Q_T, O_i^T) + R \end{bmatrix} \quad \text{and}$$
$$\begin{bmatrix} V\_SCALE + j^*C + Sim^{visual}\left(Q_V, O_j^V\right) - R, \\ V\_SCALE + j^*C + Sim^{visual}\left(Q_V, O_j^V\right) + R \end{bmatrix}$$

Note that query sphere R is an increasing parameter with number of iterations. Searching for both sun queries is concurrent. It stops when there are at least K

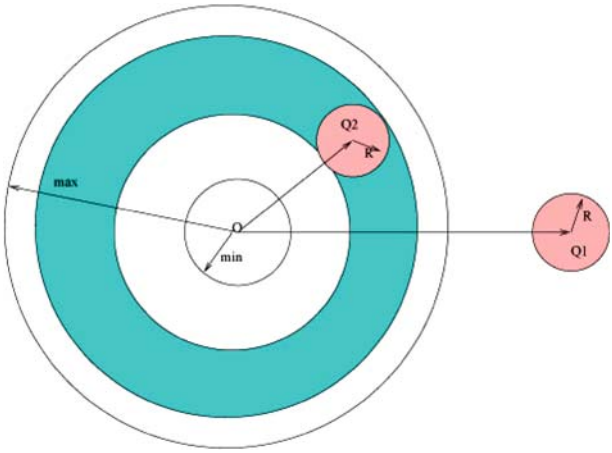

Figure 1 Searching spaces for two queries.

common image Ids are discovered in two sets of results searched from two sub queries. Thus MSI can provide approximate K nearest neighbors quickly using one dimensional data comparisons.

So far, we have built a single one-dimensional indexing for WWW image's multi-features. However, similarity-indexing key mapping function is lossy in nature. Searching the data whose similarities to cluster's center are close to query point may introduce certain number of 'false positives.' For instance, in Figure 1, although certain amount of points that are out of searching range can be pruned immediately (white area), the candidates for data access still include a number of points far away from query (green area, named as 'false positive'). It will be perfect if we can remain only the points inside of query searching sphere (pink area). In next section, we propose Local Digital Coding to effectively filter most of these 'false positives.'

4.3. Clustering techniques

As mentioned earlier, the first step for building MSI is to partition each feature space. For WWW image text feature, every image is in weighted lexical chain model. We observed that WWW images are usually categorized by different topics. Here we propose a method called Topic-driven Clustering, to partition the text space into clusters.

Topic-driven clustering algorithm:

1. select the top K hottest keywords, and each keyword is assigned as a center.
2. assign images into these K clusters based on similarities to each center.
3. reconstruct each cluster's center by summarizing its images' weighted lexical chains.
4. reassign images into K clusters based on similarities to each new center.
5. merge K clusters into a desirable number of clusters.

In this algorithm, images are initially clustered into K clusters based on the most frequent keywords appearing in the images (step 1 and 2). Obviously, not all of the images can be assigned into clusters since K keywords cannot cover all images. A reclustering process is performed. In step 2, we expand each cluster's center to be a weighted lexical chain by summarizing all images' lexical chains in the same way as summarizing the image's representation from a lexical chain network. By doing so, the center of cluster can be properly adjusted and represent the cluster's complete information. Then images are partitioned again corresponding to these new centers. Finally we merge closely related clusters into one, until we get a desirable number of cluster we want. After we apply Topic-driven clustering algorithm, the image's text feature are clustered into partitions, each of which has a weighted lexical chain as the center.

To partition the high dimensional data space, such as image's visual feature, several clustering methods [1, 4, 10] have been proposed in literature. In this paper we use the elliptical K-means [23] method to detect the *correlated* clusters in the space. The main purpose of finding correlated cluster is to perform the dimensionality reduction on these clusters locally. Recently research [11] has shown that dimensionality reduction on local correlated/elliptical cluster achieved much better effectiveness compared to reduce the dimensionality on the uncorrelated dataset. This is because dimensionality reduction methods such as Principle Component Analysis (PCA) are effectively only when the data space is well

correlated. Otherwise, the retrieval accuracy should be affected greatly. In our experiments, we also showed the effectiveness of dimensionality reduction on correlated clusters [11] integrating with our indexing methods.

4.4. Updating issues

Notice that transforming a high-dimensional feature vector to a single indexing value causes information lost. The indexing keys in MSI are computed based on the cluster IDs and the reference points. Consequently, the number of clusters and the selection of reference points will affect the effectiveness of MSI. If there are too few clusters, most cluster spaces are expected to be very large and highly overlapped. Hence most clusters are probably accessed and searched. For the extreme case, when there is only cluster, On the other hand, if too many clusters are generated, most cluster spaces are expected to be very small. However, comparing with too many clusters to check if they overlap with the query sphere introduces extensive overhead before accessing the B+-tree. For the extreme case, when each point is treated as a cluster, the whole dataset is scanned eventually. We will see the effect of number of clusters on MSI in the experiments for more details. An optimal reference point aims to minimize the inter-distance information lost caused by transformation. Transformation by an optimal reference point leads to the maximal variance of keys, hence to more effective indexing by B+-tree. As shown in [20], for a cluster, its optimal reference points lie on the direction identified by the first principle component of the data in the cluster.

WWW images are quickly evolving. As the web crawler keeps crawling, more and more images are expected to be inserted into the indexing structure. In MSI, dynamic insertion can be easily handled for indexing. The outline of the dynamic insertion in MSI is shown below:

Dynamic insertion:

1. Extract image features;
2. Find the closest cluster;
3. Generate the indexing keys for all feature spaces;
4. Insert keys into B+-tree;
5. Monitor the change of the first principle component of the cluster;
6. If the change is greater than θ
   - 6.1 Remove the cluster's indexing keys from the B+-tree;
   - 6.2 Partition the cluster into two smaller clusters;
   - 6.3 Increase the number of cluster by 1;
   - 6.4 Computed the indexing keys for the points in two new clusters;
   - 6.5 Insert two smaller clusters into the B+-tree;

When a new image is added in the dataset, its features are first extracted, followed by the closest cluster's identification (steps 1–2). To measure the closeness of a cluster to the new image, we use the distance between the new image and the cluster's center. The cluster with the smallest distance is regarded as the closest cluster. The one dimensional key value is then generated for each feature space by an optimal reference point of the closest cluster, followed by the standard insertion operation in B+-tree by inserting the key into the B+-tree (steps 3–4).

Naturally, as more images are inserted into the clusters, the clusters are growing in size and their shapes/orientations may be changed too. To be robust to the data size, MSI monitors the change of a cluster and decides when to split it into two smaller ones. To do so, MSI checks the orientation of the cluster (i.e., its first principle component) upon each insertion to the cluster (step 5). If the change of its first principle component from the original one is greater than a user defined threshold, there is a significant change in the cluster. For a cluster, the change of its first principle component is measured by the angel between the updated one and the original one. If the angel is greater than a predefined threshold θ, it is expected that the cluster's shape has been changed significantly. It is necessary to recompute the indexing keys for more effective indexing in MSI. To do so, the indexing keys of the points in the cluster are first removed from the tree (step 6.1). Since the cluster size is very big, it is also necessary to split the cluster into smaller ones so that the number of cluster also increases reasonably as the data size keeps growing. We use the elliptical k-means algorithm [23] to partition the cluster into two smaller ones (step 6.2), followed by increasing the number of clusters by 1 (step 6.3). The indexing keys of the points in the new clusters are then computed and inserted into the B+-tree (steps 4–5).

To insert a new image, the major cost is to insert its keys into the B+-tree (step 4). If a split of a cluster is triggered, the major cost consists of three parts: partition the cluster into two smaller clusters, compute the indexing keys for the smaller clusters, and insert the indexing keys of two clusters into B+-tree. However, we claim that our method is acceptable since the partition is much less frequent than the insertion of new images (controlled by θ) and only affect a single cluster instead of the whole dataset. The dynamic deletion can be handled in the similar way.

5. Local bit stream

In this section, we introduce our new indexing technique applied in MSI to break the dimensionality curse by filtering the irrelevant images greatly.

LBS is inspired by the following observations. First, digital bit (0 or 1) is the smallest data representation. If each dimension of feature space can be represented by digital bit, the memory space can be reduced dramatically. Second, bit operation is always fastest. However, in high dimensional space, the similarity computation on original data is very expensive.

Realize that for WWW images, its text feature and visual feature are in different representation models. Here, we use weighted lexical chain to represent text feature and standard high dimensional point to represent visual feature. Both have the following main differences. First, text feature is discrete in nature since each dimension of lexical chain is a word basically, while visual feature is continuous value along each dimension. Second, the dimensionality of text feature is dynamic, while that of visual feature is fixed. Different images may have different number of words to describe its semantics. To generate the uniform bits representation (we name it as Bit Stream, or BS) for both features, different encoding scheme has to be used.

Except the uniform BS representation, how to produce an effective BS for each image feature is a challenging task. Here we associate the generation of BS with the cluster center where an image belongs. That is, for an image's feature, we first allocate its cluster, and then compare it with its cluster center to generate its BS. Thus we

localize the BS generation at cluster level, rather than the whole database level. Next, we present the two encoding algorithms for text and visual features to produce a D-bit long uniform BS representation, where D is the dimensionality of feature space.

5.1. BS Generation for text feature

In text feature space, due to its discrete nature, two preliminary steps are needed before we start encoding by using the property of ordered data. We first order every image's lexical chain and the cluster center's lexical chain based on alphabet order of the words. Second, every word in each image lexical chain is labeled with its position index in the center's lexical chain. Since the center contains all the words appearing in all the images within the cluster, we use the center as the axis to generate the BS representation for images inside of cluster. The encoding algorithm for an image text feature T in a cluster O^T is shown below.

Text Encoding:

1. BS = 0;
2. range = O^T.size()/D + 1;
3. for every word in T
4. pos = word.index/range
5. BS | = 1 << pos;
6. end for

The BS is initialized to be 0. For an ordered center O^T, we divide it into D intervals (line 2). For each word in a text feature T, since we know its corresponding index in O^T, we first compute which interval it lies in (line 4), then update the bit value at that position counted from left to be 1 (line 5). For example, if T contains two words "ACM" and "Multimedia" and their respective position index in the cluster center is 10 and 100. The center contains 1,000 words. We want to construct a 64-bit BS. Based on the above encoding method, the interval range is 16. That is, the first interval is [0, 16), second is [16, 32), and so on. Words "ACM" and "Multimedia" are in interval 0 and 6. Thus the BS for T is $2^{\wedge 6} + 2^{\wedge 0} = 65$. If more bits are needed than integer, multi-integer or character can be used.

Due to text feature's discrete property, if two are similar, the AND (&) operation on two BSs must be greater than 0. By checking this result, lots of irrelevant images can be pruned immediately. This encoding algorithm can make sure the retrieval precision is exactly the same as sequential scan. Further more, the space occupied by BS is fixed in D bits. However, each original text feature generally takes hundreds of bytes.

5.2. BS generation for visual feature

Different from text feature, visual features are in high dimensional uniform. And along each dimension, the data value is continuous. In text feature space, the fixed number of intervals divided from cluster center is used to produce a uniform D-bit BS representation since the dimensionality of text feature for each image is different. However, in visual feature space, the dimensionality is fixed. Meanwhile, the similarity measurements between both spaces are also different. Thus we present a different encoding algorithm for continuous and fixed dimensionality feature spaces.

Visual encoding:

1. BS = 0;
2. for $i = 0$ to $D-1$
3. if ($V[i] > O^V[i]$)
4. BS |= 1 << i;
5. end for

Given a D-dimensional feature space, the above algorithm encodes each feature V into a D-bit BS representation given its cluster center O^V. Initially, the BS is set to be 0 (line 1). For each dimension, if its value is greater than the center value, we set the bit value to be 1 at that dimension for BS (line 3–4), else remain 0. Thus in visual space, the BS is a coarse approximation of original data.

BS for visual feature is derived from comparing its cluster's center. If two images are similar, their BSs should also be similar. To decide whether two BSs are similar, we use a threshold parameter—φ to indicate minimum number of *common bits* that two similar BSs should have. Along a dimension, if both BSs have same bit value, either 0 or 1, we say both BSs have one *common bit*. That is, along a dimension, if two visual features are both greater than the center or both less than the center, then their BSs have one common bit. Clearly, BS representation for a visual feature reflects its approximate trend/signal around its local cluster center. If both BSs have more than ϕ number of common bits, we say two BSs are similar. Given a 64-dimensional feature space, usually the values are float type—4 bytes long. Thus for each feature, it occupies 64 * 4 bytes space. However, a BS occupies 64/8 bytes. There are 32 times differences. Again, by performing bit operations on BSs, we can fast prune irrelevant images before we access the original data. Since φ is a threshold parameter, it has certain side effects. If it is too small, the pruning may not be very effective. On the other hand, if it is too big, some relevant images may be filtered. In the experiments, we will see that while we keep the same accuracy as sequential scan, the retrieval speed can still be faster than sequential scan by times.

So far, we have looked the encoding method to produce BSs for text and visual features. And both encoding algorithm use the *local cluster's center* as a basis. The final outputs from both algorithms have the same representation model—BS.

LBS builds a new simple feature representation called BS for each image in respective feature space. BS is small in size, and fast comparison since it's only involved bit operations. BSs can be embedded into MSI lower than the indexing keys level and upper than the original data level. An example tree structure is shown in Figure 2. Thus, after the first level pruning in MSI, a second level pruning by comparing BSs is performed to filter most of 'false positives' included in the first level pruning. The images whose BSs are similar to query BS are then accessed at data level. Experiments showed that while keeping the accuracy high, 90% more 'false positives' could be effectively pruned.

6. Evidence integration

In this section, we examine four evidence combination techniques: Certainty Factor, Dempster Shafer Theory, Compound Probability, and Linear Combination, to

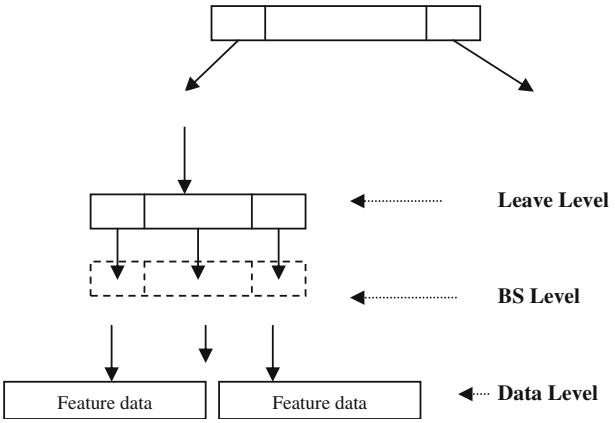

Figure 2 Overall Structure of MSI with LBS.

combine the similarities computed from text and visual spaces. We denote the text similarity as Sim^{visual} and visual similarity as Sim^{text} like before.

6.1. Certainty factor

MYCIN expert system [21] first introduced the concept of *certainty factor (*CF) to deal with uncertainty. Certainty factor is a way of combining belief and disbelief for an evidence into a single number. It can be used to rank the hypothesis in order of importance/similarity. Definition of CF can be defined as following:

$$CF = \frac{MB - MD}{1 - \min(MB, MD)}$$

where MB is the measure of belief and MD is the measure of disbelief. In our image retrieval system, we treat the belief of text feature evidence as the similarity between two image text features Sim^{text}, and the disbelief as $1-Sim^{text}$. Based on the above formula, the CF for text feature evidence is:

$$CF = \frac{Sim^{text} - (1 - Sim^{text})}{1 - \min(Sim^{text}, 1 - Sim^{text})}$$

This formula is also applied to visual features. That is, the CF for visual space evidence is simply to replace Sim^{text} with Sim^{visual} in the above formula.

To combine two CFs from two evidences, the formula goes as follows if both CFs > = 0:

$$\begin{aligned} CF_{combine} &= CF_1 + CF_2(1 - CF_1) \quad \text{if both CFs are} > 0 \\ &= \frac{CF_1 + CF_2}{1 - \min(|CF_1|, |CF_2|)} \quad \text{if one of CFs is} < 0 \\ &= CF_1 + CF_2(1 + CF_1) \quad \text{if both CFs are} < 0 \end{aligned}$$

To compute the final similarity between two images, we use the above formula based on two CFs computed from text and visual evidences.

6.2. Dempster Shafer Theory

Dempster Shafer Theory provides a way to combine independent evidences [2]. A fundamental difference between Dempster Shafer Theory and probability theory is the treatment of *ignorance*. When there is no evidence to a statement, probability theory has to assign a possibility; however, Dempster Shafer Theory does not enforce a belief to be assigned to ignorance. For example, if to an evidence, its belief assigned to x is $m(x)$, then the rest belief (uncertainty) on x is left to the environment and denoted as $m(\Theta) = 1-m(x)$. By using the Dempster's rule of combination, the combined evidence for a hypothesis z is computed as:

$$m_1 \oplus m_2(z) = \sum_{x \cap y = z} m_1(x) m_2(y)$$

where $m_1 \oplus m_2(z)$ is the combined probability for set z. $m_1(x)$ and $m_2(y)$ are the beliefs assigned to set x and y by two sources of evidences.

In our system, we treat Sim^{text} and Sim^{visual} as the beliefs assigned to text and visual features by two sources computed by respective similarity measurement. We can use the following table to show how to compute the combined similarity from two evidences by assigning the similarities for text and visual as 0.8 and 0.6 (Table 1).

The final similarity by Dempster's rule of combination for our system is:

$$Sim = m_{text}(text)*m_{visual}(visual) + m_{text}(\Theta)*m_{visual}(visual) + m_{visual}(\Theta)*m_{text}(text)$$

or simply

$$Sim = 1 - m_{text}(\Theta)*m_{visual}(\Theta)$$

Thus for example given, we can get the final similarity value as 0.8*0.6 + 0.6*0.2 + 0.4 * 0.8 = 1−(0.4 * 02) = 0.92.

6.3. Compound probability

Compound Probability assumes that two evidences are completely independent. The final probability is simply computed as:

$$Sim = Sim^{test} * Sim^{visual}$$

Table 1 Dempster's rule of combination in real system.

	$m_{text}(text) = Sim^{text} = 0.8$	$m_{text}(\Theta) = 1\text{-}Sim^{text} = 0.2$
$m_{visual}(visual) = Sim^{visual} = 0.6$	0.8 * 0.6	0.6 * 0.2
$m_{visual}(\Theta) = 1\text{-}Sim^{visual} = 0.4$	0.4 * 0.8	0.4 * 0.2

6.4. Linear combination

This is the most popular combination method used in most existing systems. By assuming that two evidences are linearly important, the final similarity is computed as:

$$Sim = \alpha * Sim^{test} + (1 - \alpha) * Sim^{visual}$$

where α is the weight assigned to text feature, and it can be adjusted when necessary.

7. Performance study

In section, we present our experiments results on our proposals. We compare our indexing technique with sequential scan and multi-indices built by iDistance method [26], where the indexing keys are computed by similarity rather than distance. In the following, we refer our MSI with LBS as LSB only.

7.1. Experiments set up

Our database contains 100,000 WWW images downloaded by our web crawler randomly from around 40,000 websites, which are mainly from National University of Singapore, BBC and World Travel Guide web sites. And we use the weighted lexical chain and wavelet descriptors as image's text and visual features as explained in Section 3. Wavelet visual features are truncated in dimensionality of 64. We also perform dimensionality reduction [11] in visual feature space to generate corresponding set of 30-dimensional subspaces of visual feature for comparative study. We manipulated these databases in MYSQL server. We implement our method in the environment of Ultra-10 SunOS 5.7 processor (333 MHz CPU and 256 MB RAM).

Two parameters are used as measurements: Precision and Efficiency. Since our database is large, it's impossible to compute the retrieval recall. Here we use fixed KNN (K nearest neighbor) to compute the precision. Obviously, within this K results, if the precision is higher, recall is higher also. The results are manually judged by users. We set K = 20, and test 20 image queries which are randomly selected from the database for each experiment. Efficiency is measured by the *Total Response Time* (TRT), which includes the communication time to the MYSQL server.

7.2. Tuning φ

In our LBS method, the important parameter φ which is used to measure the similarity between two BSs has to be tuned. For text feature's BS, due to discrete nature, we have to access every feature point whose BS AND (bit operation &) query's BS is greater than 0. So we need tune φ for visual features only.

In this experiment, we use image's visual features only—the 64 and 30-dimensional data to see the changing of φ for different dimensional data space. Here we the *relative precision* by comparing LBS with sequential scan. The relative

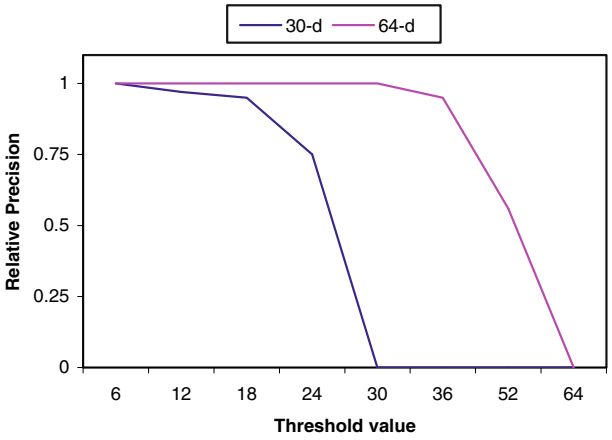

Figure 3 φ Effects on precision.

precision is defined as precision by LBS divided by precision by sequential scan. The following Figures 3 and 4 show the effect of different φ values on relative precision and efficiency for two sets of visual features.

From Figure 3, we can see that for 64-dimendional original data, LBS can remain the same precision as sequential scan when φ is less than 36. When φ is greater than 36, there is rapid decreasing on precision. This is reasonable. When φ is larger, more points can be pruned. As φ becomes too large, there may be only less than K candidates remained. When φ becomes 64, only the query image is the candidate to access the original data. Similar situation happens to 30-dimensional reduced data space also. When φ is less than 18, there is no obvious difference between LBS and sequential scan. The slightly less precision is most probably due to the information lost during the dimensionality reduction. Thus a φ value which is a bit larger than the half size of the dimensionality of data space can remain the precision high.

Figure 4 shows the φ effects on the total response time. TRT for sequential scan is the result when φ = 0. For 64-dimensional space, when φ is less than 24, there is no

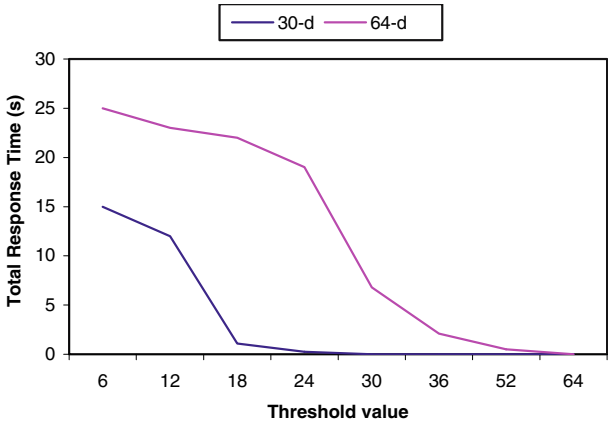

Figure 4 φ Effects on total response time.

obvious reduction on TRT. Similar situation happens to 30-dimensional reduced space when φ is <12. This is clear that in high dimensional space, most of points have few common bits along few dimensions with query point, while they may not be necessary to be the top K nearest neighbors to the query point. However, when φ increases to be larger than the half size of the dimensionality of data space (36 for 64-dimensional space and 18 for 30-dimensional space), the TRT is reduced dramatically. This indicates that most of the irrelevant points can be distinguished when φ reaches around the half size of the dimensionality. As ϕ becomes too large, more points can be filtered, but precision may be affected as shown in Figure 3. Thus there is a tradeoff between precision and TRT. From Figures 3 and 4, we can see that good values for φ could be a bit larger than the half size of the dimensionality. At these values, TRT can be reduced greatly, while precision is still high. In our later experiments, we chose φ as 36 and 18 for 64- and 3-dimensional spaces, respectively.

As a step further, we also test the sensitivity of our method to K, the number of nearest neighbors. We use the 64-dimendional original data and set φ to be 36. As shown from Figure 5, as K increases, the precision increases correspondingly. It is reasonable since when K is small, a single miss of results affects precision more significantly. This confirms the effectiveness of our method with respect to various K values.

7.3. Effect of the number of clusters

As we mentioned in Section 4.4, the performance of MSI is affected by the number of clusters. In this experiment, we investigate the effect of the number of clusters.

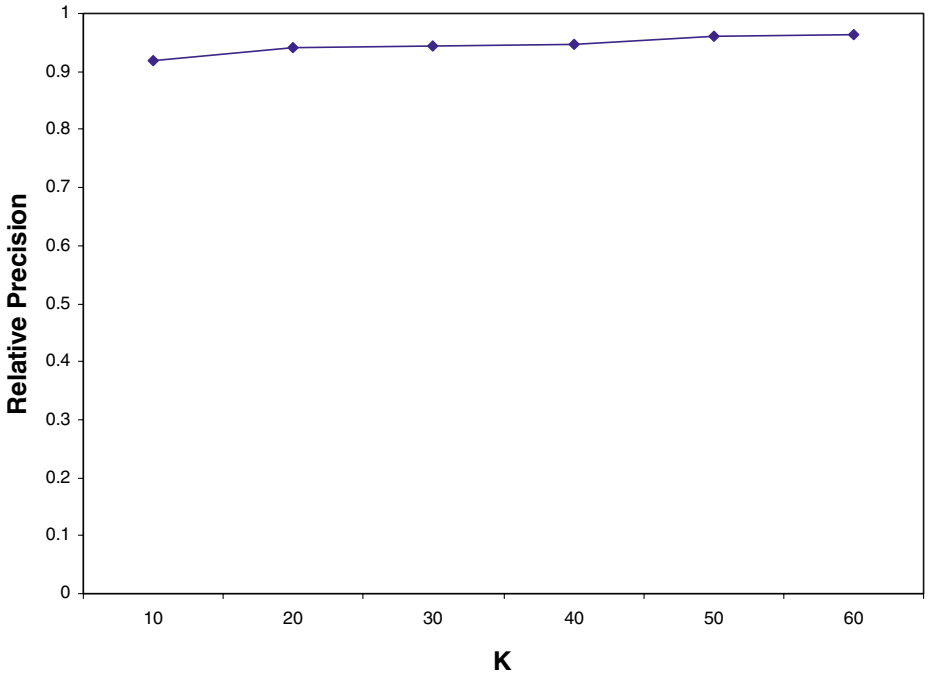

Figure 5 Effect of K on precision.

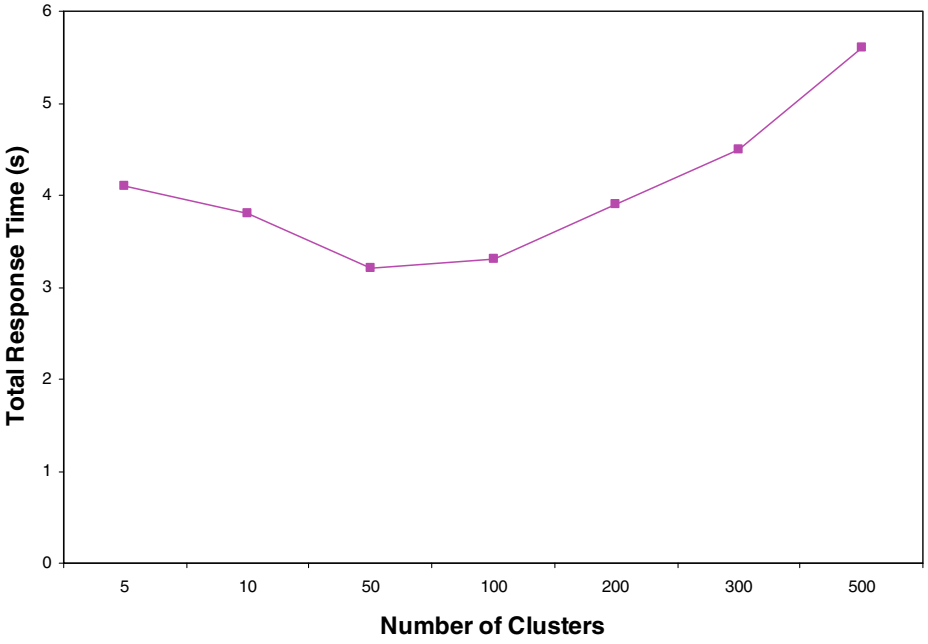

Figure 6 Effect of the number of clusters on total response time.

Figure 6 shows the performance of our method with respect to the number of clusters for both text and visual features, where visual feature is 64-dimensional space. As we can observe from Figure 6, our method does not perform best when the number of clusters is either too small, or too big. When there are too few clusters, each cluster spans a large space. And most cluster spaces are expected to be highly overlapped. As a result, given a query, a very large portion of a cluster space is possibly to be accessed and most clusters are probably searched. On the contrary, when there are too many clusters, most cluster spaces are expected to be very small. However, comparing with too many clusters to check if they overlap with the query sphere introduces extensive overhead. Figure 6 suggests that for our dataset, partitioning the dataset into 50 clusters achieves the best performance. Hence in the following experiments, we set the number of clusters to be 50.

7.4. Comparative study

Now we want to compare our indexing method LBS with sequential scan and multi-indices by iDistance [26], based on the retrieval speed. We also test the performance of dimensionality reduction (DR) [11] on visual feature (i.e., the 30-dimensional reduced space) together with LBS. This experiment tested three datasets: text feature only, visual feature only, and text combined with visual features. The following table shows their differences in terms of total response time (s).

From Table 2, we can see that multi-indices method built by iDistance performs even worse than sequential scan. There are two possible reasons. First, accessing and searching multi-indices take more time. Second, dimensionality curse resists in such purely similarity based one-dimensional index because too many 'false positives' are

Table 2 Comparative study on retrieval speed.

	Text feature	Visual feature	Combination
LBS with DR		1.1	2.3
LBS	0.8	2.1	3.2
Sequential scan	3.5	21	33
Multi-indices by iDistance	5.7	32	41

searched. By employing single index structure and LBS, the performance is improved to a different level. LBS is more than an order of magnitude better than sequential scan. Clearly, our LBS is an effective method to filter those irrelevant points. By applying dimensionality reduction, we can further speed up the retrieval process. Obviously, reduced space takes less space. This will reduce the accessing time. And the computation cost can also be reduced since less dimensionality is involved.

7.5. Dynamic insertion

In this experiment, we study the effectiveness of our methods in dynamic environments. We constructed the index by using the first 20,000 images, and then we inserted 20,000 images at a time. After each insertion, we performed KNN search. We set the θ to be $\pi/16$. To see the performance of our dynamic insertion method, we compare with one-off construction (i.e., index rebuilt upon each

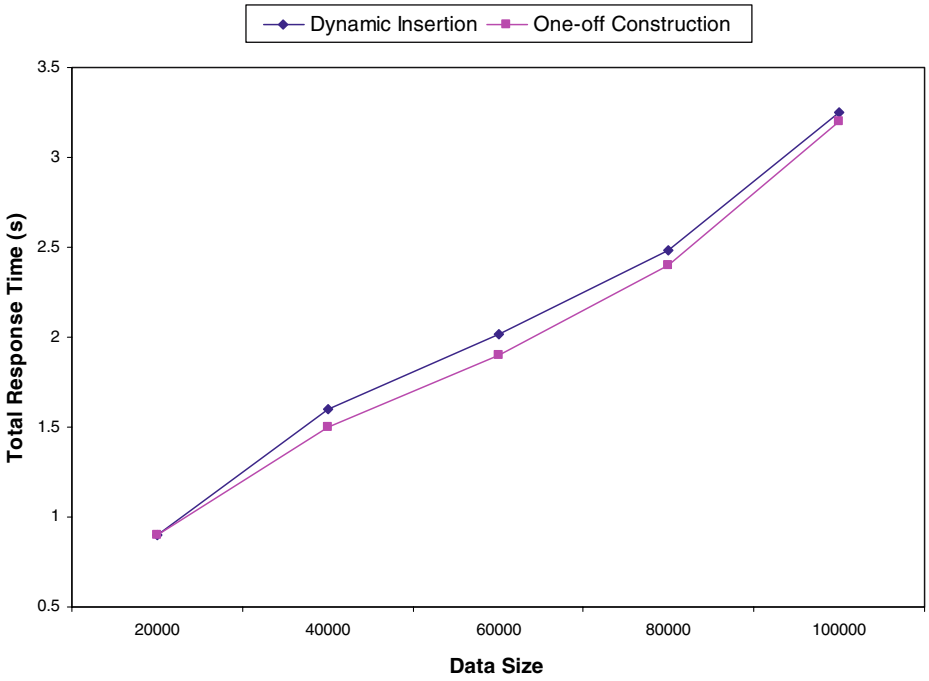

Figure 7 Performance of dynamic insertion.

Table 3 Effect of different weightages for linear combination.

Text weightage	0	0.1	0.3	0.5	0.7	0.9	1
Precision	0.28	0.34	0.58	0.80	0.83	0.82	0.81

insertion). Figure 7 depicts the performance of our method, comparing with the one-off construction.

Figure 7 shows that the response time increases as the data size grows. Compared to the performance of dynamic insertion with one-off construction, we notice that the performance of dynamic insertion slightly degrades. In MSI, the reference points are selected based on the clusters' first principle components. Naturally, as more images are inserted into the clusters, the first principle components of the clusters may shift away from the original ones. Obviously, the further the first principle components move away, there is a greater negative effect on the performance of our method. To preserve the effectiveness of our method, we set a threshold on the changing angels of the first principle components. Whenever the change is greater than a threshold, we partition the cluster into smaller ones and rebuild a small part of the structure. By doing so, rebuilding the whole tree is avoided, while the effectiveness of the method can be preserved. The neglected increase on the response time in Figure 7 confirms the effectiveness of our dynamic insertion.

7.6. Effectiveness of evidence combination

In this experiment, we investigated four evidence combination methods in order to choose the best technique. The average precision we tested (manually judged by a user on the top 20 results for 20 queries) on text feature and visual feature is 0.81 and 0.28, respectively. First, we tune the optimal weightages of text and visual features for linear combination, where the sum of the weightages is 1.

From Table 3, we can see that as the text weightage increases (i.e., the visual weightage decreases), the precision increase and reaches a climax when the text weightage reaches 0.7. When the visual feature dominates the overall similarity (text weightage ranges from 0 to 0.3), the precision is very low. On the other hand, when the text feature dominates the overall similarity (text weightage ranges from 0.5 to 1), the precision is very high. This indicates that text feature is much more powerful in capturing the image's semantics. On the other hand, visual feature cannot capture the high level semantic meaning of images, like text does. However, integrating text feature with visual feature does help to improve the precision slightly by setting text feature with a higher weightage. As shown in Table 3, we assign the tuned optimal weightages for text and visual features to be 0.7 and 0.3, respectively.

The following Table 4 indicates the respective precision of four methods. From Table 4, we can see that Certainty Factor and Dempster Shafer Theory perform best. Certainty factor utilizes the importance of an evidence by combined the belief and disbelief factors in each hypothesis. And Dempster Shafer Theory can significantly improve the precision if two evidences are far from optimum.

Table 4 Effectiveness of combination methods.

Certain factor	Dempster Shafer Theory	Compound probability	Linear combination
0.88	0.87	0.51	0.83

Obviously, for WWW images, the precision retrieved in their visual features is far away from optimum since they cannot capture the high level semantic meaning of images. Linear Combination has marginal improvement on text-based retrieval.

However, Compound Probability even degrades the text-based retrieval. This is because if two evidences have large difference, its combined similarity will be very small. However, in our system, text and visual feature indicates different level of semantics.

8. Conclusion

In this paper, we presented a novel indexing technique called MSI to index WWW image's multi-features in a single one-dimensional structure. Combined with Local Bit Stream, our method can outperform sequential scan and multi-indices by iDistance significantly without degrading the retrieval precision. Meanwhile, dimensionality reduction method can help to further improve the performance. We also examine four evidence combination techniques and experiments showed that Certainty Factor and Dempster Shafer Theory perform best.

References

1. Aggrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications., Proceedings of the ACM SIGMOD Conference, pp. 94–105 (1998)
2. Aggrawal, C.C., Wolf, Joel L., Yu, P.S., Procopiuc, C., Park, J.S.: Fast algorithms for projected clustering, Proceedings of the ACM SIGMOD Conference, pp. 61–72 (1999)
3. Aslandogan, Y.A., Yu, C.T.: Evaluating strategies and systems for content based indexing of person images on the Web., ACM Multimedia, pp. 313–321 (2000)
4. Cai, D., He, X., Li, Z., Ma, W.-Y., Wen, J.-R.: Hierarchical clustering of WWW image search results using visual, textual and link analysis. ACM Multimedia (2004)
5. Chakrabart, K., Mehrotra, S.: The hybrid tree: An index structure for high dimensional feature spaces., International Conference on Data Engineering, pp. 322–331 (1999)
6. Chen, Z., Wenyin, L., Hu, C., Li, M., Zhang, H.: iFind: A web image search engine., SIGIR (2001)
7. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware., PODS (2001)
8. Gaede, V., Gunther, O.: Multidimensional access methods. ACM Comput. Surv. **30**(2), 170–231 (1998)
9. Guntzer, U., Balke, W-T., Kiessling, W.: Optimizing multi-feature queries for image databases., VLDB, pp. 261–281 (2000)
10. Hinneburg, A., Keim, D.A.: An optimal grid-clustering: towards breaking the curse of diminsionality in high dimensional clustering., VLDB (1999)
11. Jin, H., Ooi, B.C., Shen, H.T., Yu, C., Zhou, A.: An adaptive and efficient dimensionality reduction algorithm for high-dimensional indexing, ICDE, pp. 87–98 (2003)
12. Mukherjea, S., Hirata, K., Hara, Y.: Amore: A World Wide Web image retrieval engine. The WWW Journal **2**(3), 115–132 (1999)
13. Ngu, A.H.H., Sheng, Q.Z., Huynh, D.Q., Lei, R.: Combining multi-visual features for efficient indexing in a large image database. VLDB J. **9**(4), 279–293 (2001)
14. Ooi, B.C., Tan, K.L., Yu, C., Bressan, S.: Indexing the edges—a simple and yet efficient approach to high-dimensional indexing., PODS, pp. 166–174 (2000)
15. A Review of Content-Based Image Retrieval Systems, http://www.jtap.ac.uk/reports/htm/jtap-054.html
16. Sakurai, Y., Yoshikawa, M., Uemura, S., Kojima, H.: The A-tree: An index structure for high-dimensional spaces using relative approximation., VLDB, pp. 516–526 (2000)

17. Sclaro, S., Taycher, L., La Cascia, M.: Imagerover: A content-based image browser for the World Wide Web. In Proc. IEEE Workshop on content-based access of image and video libraries (1997)
18. Shen, H.T, Ooi, B.C., Tan, K.L.: Giving meanings to WWW images. In Proc. of 8th ACM Multimedia Conference, pp. 39–47 (2000)
19. Shen, H.T., Zhou, X., Cui, B.: Indexing text and visual features for WWW images. To appear in Proceedings of 7th Asia Pacific Web Conference (APWEB 2005)
20. Shen, H.T., OOi, B.C., Zhou, X., Huang, Z.: Towards effective indexing for very large video sequence database. SIGMOD, pp. 730–741 (2005)
21. Shortliffe, E.H.: Computer-based medical consultation: MYCIN. Elsevier North-Holland, New York
22. Smith, J.R., Chang, S.-F.: An Image and video search engine for the World-Wide Web., Proceedings, IS&T/SPIE Symposium on Electronic Imaging: Science and Technology (EI'97)—Storage and Retrieval for Image and Video Databases V (1997)
23. Sung, K.K., Poggio, T.: Example-based learning for view-based human face detection. PAMI **20**(1), 39–51 (1998)
24. Wang, J.Z., Wiederhold, G., Firscheinvv, O., Wei, S.X.: Content-based image indexing and searching using Daubechies' wavelets. Int. J. Digit. Libr. **1**(4), 311–328 (1998)
25. Weber, R., Schek, H., Blott, S.: A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces., VLDB, pp. 194–205 (1998)
26. Yu, C., Ooi, B.C., Tan, K.L., Jagadish, H.V.: Indexing the distance: An efficient method to KNN processing. VLDB, pp. 421–430 (2001)
27. Yu, S., Cai, D., Wen, J.-R., Ma, W.-Y.: Improving pseudo-relevance feedback in web information retrieval using web page segmentation., World Wide Web (2003)