ABSTRACT

This paper proposes a novel content-based image retrieval technique, which integrates block-based visual features and user’s query concept-based semantic features. It also facilitates short-term and long-term learning processes by integrating users’ historical relevance feedback information. The history is compactly stored in a semantic feature matrix and efficiently represented as semantic features of the images. The short-term relevance feedback technique can benefit from long-term learning. The high-level semantic features are dynamically updated based on users’ query concept and therefore represent the image’s semantic meaning more accurately. Our extensive experimental results demonstrate that the proposed system outperforms its seven state-of-the-art peer systems in terms of retrieval precision and storage space.

Index Terms— Content-based image retrieval, relevance feedback, semantic feature matrix

1. INTRODUCTION

Content-based image retrieval (CBIR) is a popular topic in recent research in computer vision, pattern recognition, and information retrieval due to the increasing number of available digital images. It is a method that searches images with similar concepts based on their visual content such as color, shape, texture, etc. The challenging issue in CBIR research is the semantic gap. When people see an image, they can easily tell its content and meaning. But the computer can only understand the low-level visual features of an image. Therefore, the semantic gap is the discrepancy between low-level visual features and high-level semantic concepts. This is still an open problem in the CBIR field.

To bridge the semantic gap, relevance feedback (RF) techniques [1] have been widely used in CBIR systems to improve the retrieval performance. RF is an interactive process in which the user labels correctly retrieved images as relevant to the query and others as irrelevant to the query. This feedback reveals semantic relations among images and therefore can be used to refine the query in a retrieval session. However, most existing RF techniques use short-term (intra-query) learning to handle query formulation in a single retrieval session. Recently, long-term (inter-query) learning [2-9] extends short-term learning to derive semantic meaning of database images by studying the feedback history collected from multiple users in different retrieval sessions. Long-term learning has mainly been studied in two ways. First, it is used to establish the relationship between the current and previous query sessions by analyzing retrieval patterns across multiple sessions. Images with similar retrieval patterns as query are returned as the retrieval results. Second, it is used to bring feature vectors of similar images close to each other by a weighting or transformation scheme. Images with similar feature vectors as query are returned as the retrieval results.

The remainder of this paper is organized as follows. Section 2 briefly reviews the related work. Section 3 presents low-level visual features together with their distance measures. Section 4 presents the process to construct the compact semantic feature matrix and extract and update high-level semantic features using users’ RF. Section 5 illustrates the experimental results of comparison with seven state-of-the-art peer systems on different databases. Section 6 concludes the paper with a brief discussion of our approach and future work.

2. RELATED WORK

Long-term learning techniques generally use users’ log information in past query sessions to help retrieval in the current retrieval session. The information in the log is usually aggregated into a matrix, which allows the CBIR system to discover extra knowledge (i.e., the semantic relevance of a database image to the current query). Here, we briefly review several long-term techniques that are closely related to the proposed approach.

He et al. [2] construct a semantic matrix to store retrieval patterns of all query sessions and apply dot product to find semantically similar images. However, they only store information for positive feedback images and ignore negative feedback images. They also do not propagate learned information to other related images. Hoi et al. [3] apply the statistical correlation on the retrieval log to analyze the relationship between current and past retrieval
sessions. They save both positive and negative feedback information in the retrieval log with values of 1, -1, and 0 representing positive, negative, and no learned relationship, respectively. Han et al. [4] use the feedback log to compute the ratio of co-positive-feedback frequency and co-feedback frequency for analyzing the relationship among query sessions. They further apply the memory learning technique to store semantic information and learn semantic relations. Yin et al. [5] design a virtual-features-based technique to record all the concepts digested from long-term feedback history and update virtual features of the image to estimate the semantic relevance between images. Qi et al. [6] apply the dynamic semantic clustering technique on the feedback log to cluster images into several semantic homogenous clusters for efficient retrieval. Xiao et al. [7] use a dynamic semantic matrix to store and update users’ feedback information in multiple sessions to capture more accurate semantic features. They then derive the semantic relevance among images and combine the semantic similarity with the visual similarity to find top images that are similar to the query image. Chang et al. [8, 9] use RF to group similar images into semantic clusters and further use semantic clusters based information to construct a weighted manifold structure in two ways to propagate the ranking scores of labeled images. Although these learning techniques achieve impressive retrieval results, they usually require a large matrix to store historical feedback information to learn the relationships among all images. This large matrix prevents some CBIR systems from searching a large-scale image database. For example, manifold ranking systems [8, 9] require several coexisting large matrices to propagate learned labels. They cannot run on a computer if the number of images is over 10,000 due to the large consumption of the memory space. In addition, the large matrix may be sparse if queries fall into a few semantic categories, which may deteriorate the learning performance. 

To address large storage issues in long-term learning, we propose a novel block-based long-term learning scheme for CBIR by dynamically merging similar semantic concepts. First, we divide the original image into 9 non-overlapping blocks and apply the minimum distance matching approach to compute the block-based image similarity. Second, we build an adaptive semantic feature matrix in long-term learning to store retrieval patterns of historical query sessions. Third, we apply a merging approach to combine similar columns in the semantic feature matrix to make it more compact. Fourth, we update semantic features of positive images labeled in each RF iteration of the query session to ensure all positive images share similar semantic concept. Fifth, we integrate the low-level block visual feature-based and high-level semantic feature-based image similarity to estimate the semantic relevance among images. The proposed CBIR system is different from the other long-term CBIR systems from the three perspectives: 1) It uses block-based low-level features to measure the image visual similarity. 2) It uses adaptive high-level semantic features extracted from the semantic feature matrix to measure the image semantic similarity. 3) The semantic feature matrix constructed from the user’s RF is compact and consumes a significant less storage.

3. LOW-LEVEL VISUAL FEATURES AND THEIR DISTANCE MEASURE

Most CBIR systems use global features to represent an image due to their computational simplicity. However, there are some drawbacks of using the global feature representation. One is that some spatial information is lost. The other issue is that the user is often interested in a certain region of one image. Therefore, using global features to represent an image will result in limited description power. To overcome the limitations of global features based CBIR methods, some researchers propose region-based image retrieval (RBIR) [10, 11]. In RBIR, an image is segmented into several regions. Low-level features are then extracted for each region. In [10], a unified feature matching (UFM) approach is used to measure the similarity among regions. In [11], a vector quantization method is employed to achieve compact and sparse regional representation. The earth mover’s distance (EMD) is further applied to measure the similarity of images. RBIR is proven to be more effective than global features-based CBIR. But it has higher time complexity and memory usage. The segmentation and local feature extraction processes take a lot of time. Matching those regions also takes more time and makes the retrieval process slower.

To simplify the segmentation process, we propose to use the blocking scheme to quickly obtain several blocks. This choice is also supported by the findings in image annotation [12], where the block-based scheme has been proven to achieve comparable annotation results as the complicated image segmentation scheme. In our system, we divide each image into 9 blocks with the same size as illustrated in Fig. 1. For each block, we extract three kinds of features, e.g., color, edge, and texture features. These 54-dimensional features include 9 color, 36 edge, and 9 texture components. Specifically, we compute the first three moments in each HSV color channel to represent color. We compute the 36-bin edge direction histogram to represent edges in the grayscale image. We compute the entropy of each of nine detail wavelet subbands to represent texture in the grayscale image.

![Fig. 1. Illustration of image blocks](image-url)
We use the Euclidean distance of the normalized low-level visual features to measure the block similarity. We further propose a minimum distance matching approach to compute the block-based similarity between two images $I_i$ and $I_j$. by:

$$BD_{distSim}(I_i, I_j) = \frac{1}{9} \sum_{k=1}^{9} \min(d(I_i^k, I_j^k), ..., d(I_i^9, I_j^9))$$

where $I_i^k$ represents the $k^{th}$ block of the $i^{th}$ image and $d(I_i^k, I_j^k)$ is the Euclidean distance between $I_i^k$ and $I_j^k$. This matching strategy works as follows: For each block in image $I_i$, we compute its Euclidean distance to all blocks in image $I_j$ and keep the smallest distance value. We then average the smallest distance value for all blocks in $I_i$. This average distance is the similarity between two images $I_i$ and $I_j$. The smaller the distance, the higher the similarity is.

4. COMPACT SEMANTIC FEATURE MATRIX AND SEMANTIC FEATURE-BASED LEARNING

The proposed long-term learning system aims to use a compact semantic feature matrix to store users' retrieval patterns so each image can be more accurately represented by multiple features: the block-based visual features and the user's query concept-based semantic features.

4.1. Formulate semantic feature matrix

We use a semantic feature matrix to dynamically store and update historical retrieval and feedback experiences from multiple users to capture more accurate semantic features. The size of this semantic feature matrix is initially set to be $N \times 50$, where $N$ is the total number of images in the database and 50 is an initial estimate of the number of semantic concepts in the database. This estimate may be doubled or tripled when the number of semantic concepts in the database exceeds the estimate. Fig. 2 shows a sample semantic feature matrix, where each row represents the semantic feature for one image, and each column represents one semantic concept corresponding to the query. For example, the first row represents the semantic feature of the 1st database image (img1) and the 100th row represents the semantic feature of the 100th database image (img100). The 1st column $C_1$ represents the semantic concept (e.g., sky) of the 1st query and the 50th column $C_{50}$ represents the semantic concept (e.g., flower) of the 50th query. The value in each cell means the relevance of a database image to the corresponding semantic concept. A positive value means that the current image has the similar meaning as the corresponding semantic concept. A larger positive value indicates the current image likely to possess such a semantic concept. A negative value means the current image does not have the corresponding semantic concept. A negative negative value indicates the current image unlikely to possess such a semantic concept. A zero value means that there is no established relationship between the current image and the corresponding semantic concept since the current image has not been retrieved for the query.

<table>
<thead>
<tr>
<th></th>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$C_3$</th>
<th>$C_4$</th>
<th>$C_5$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>img1</td>
<td>-0.01</td>
<td>3.85</td>
<td>-0.05</td>
<td>-0.1</td>
<td>-0.04</td>
<td>...</td>
</tr>
<tr>
<td>img2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>img3</td>
<td>-0.06</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>img4</td>
<td>-0.12</td>
<td>-0.02</td>
<td>3.34</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>img100</td>
<td>-0.12</td>
<td>-0.02</td>
<td>3.34</td>
<td>0</td>
<td>-0.03</td>
<td>...</td>
</tr>
</tbody>
</table>

Fig. 2. Illustration of the semantic feature matrix

At the beginning of the retrieval process, this matrix is empty (contains all 0's) since no knowledge has been learned. After the first query iteration, we simply mark the cells corresponding to all returned positive images at the 1st column as 1's, and mark the cells corresponding to all returned negative images at the 1st column as -1's.

To effectively use users’ RF information, we update semantic features of all positively labeled images after each RF iteration. The update strategy is guided by the observation that all positively labeled images should have similar semantic concepts. To this end, we perform the average operation on the semantic features of all positive images labeled in a RF iteration and use this average to update semantic features of each of positively labeled images. This operation ensures to propagate the learned semantic features of positively labeled images to other positively labeled images, especially for those whose semantic features are empty (not learned yet).

Fig. 3 shows an example of the update process. To facilitate discussion, we assume that the semantic feature matrix has been properly updated after performing three query sessions. Fig. 3(a) shows the snapshot of the semantic feature matrix after three query sessions. This snapshot contains only pertinent information related to the next query (e.g., the 4th query). For example, after one RF iteration of the 4th query, images 3, 5, and 42 are labeled as positive images, and image 210 is labeled as a negative image. Among these three positive images, img42 has empty semantic feature. So the update process propagates learned features of img3 and img5 (e.g., $(2, 1, 0)$ and $(0, -1, 1)$) to img42 by assigning the average semantic features of all three positive images to each positive image. In other words, all three positive images have the same semantic feature $(2/3, 0, 1/3)$, which contains the average value of the semantic features of three positive images. For negative images, we cannot propagate the learned semantic features...
since they may not necessarily share the similar concepts at the current session. Fig. 3(b) shows the snapshot of the updated semantic feature matrix. It clearly shows that semantic features of img42 are no longer empty and the semantic features of all three positive images are the same. In addition, we mark the cells corresponding to positive images at the 4th column as 1’s, and mark the cells corresponding to negative images at the 4th columns as -1’s.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>img3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>img5</td>
<td>0</td>
<td>-1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>img42</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>img210</td>
<td>0.5</td>
<td>-1</td>
<td>0</td>
<td>-1</td>
</tr>
</tbody>
</table>

Semantic feature matrix (a) after 3rd query (b) after 4th query

Fig. 3. Illustration of updating semantic feature matrix

4.2. Merge similar semantic concepts

To keep the semantic feature matrix compact, we propose a merging process. The CBIR system learns a new concept after each query session and tries to merge it with existing concepts based on a merging rule. As we mentioned earlier, the values in each column of the semantic feature matrix represent the relevance of each database image to the semantic concept of the query associated with that column. Given a query session, if one of returned positive images has been marked as a previous query image, we know the current query image must share similar semantic meaning with this previous query and therefore merge the corresponding columns into one new column. Similarly, if two or more returned images have been used as query images in the previous sessions, we know that current query image must share similar semantic meaning with these previous query images. We then merge these corresponding columns into one column by assigning average values of these merged columns. We perform this merging process at the end of each query session. That is, we decide whether the column of the current query can be merged with the previous columns in the semantic feature matrix by checking the above merging rule. If the merging condition is satisfied, we perform the merging operation and decide whether the newly merged column can be merged with the remaining columns in the semantic feature matrix. This iterative process stops when there is no merging operation occurs. Fig. 4 shows one example of this merging process.

4.3. Combine visual features and semantic features

To speed up the comparison process, we use the dot product to calculate the similarity of semantic features of two images. The higher value indicates the two images are more similar. At the 1st iteration of a query session, we directly calculate the dot product of query image and the other database images. In the rest iterations, we use the average semantic features of positive images and the average semantic features of negative images to perform the dot product, as summarized in Eq. (3). The idea is as follows: if an image is similar to the query image, its semantic features should be close to the semantic features of positive images labeled in the query session, and far away from the semantic features of negative images labeled in the query session.

\[
HighSim(q, I_j) = \begin{cases} 
SF(q) \cdot SF(I_j) & \text{1st iteration} \\
SF(AvgP) \cdot SF(I_j) - SF(AvgN) \cdot SF(I_j) & \text{otherwise} 
\end{cases}
\]

where \(SF(q) \cdot SF(I_j)\) represents the dot product between semantic features of query image \(q\) and a database image \(I_j\), \(SF(AvgP)\) represents the average semantic features of \(q\)’s positive images, and \(SF(AvgN)\) represents the average semantic features of \(q\)’s negative images.

Finally, we compute the overall similarity between the query image \(q\) and the database image \(I_j\), by:

\[
Sim(q, I_j) = w_q \times HighSim(q, I_j) - w_l \times BDissim(q, I_j)
\]

Here, \(w_q\) and \(w_l\) control the weight of block-based local visual feature-based similarity and the query concept-based semantic feature-based similarity, respectively. We set \(w_l\) to be 0.4 and \(w_q\) to be 0.6 due to the importance of the learned semantic features. The higher the overall similarity value, the more similar the two images are.

5. EXPERIMENTAL RESULTS

We conduct a set of carefully designed experiments to evaluate the performance of our proposed block-based
CBIR system on 2000 images, 6000 images, 8000 images, and 12000 images.

5.1. Dataset

To simplify the retrieval process, we manually organize database images into semantic classes. Hence, the image relevance is automatically determined by checking whether returned images belong to the same class as the query. It should be noted that the ground truth is used to evaluate the retrieval performance and is not used to provide additional class-related information for the proposed system. To this end, we collect the following images for the experiments:

- 6000 COREL images: These images are chosen from the COREL database and cover 60 scenes such as cars, buildings, etc. Each scene contains 100 images.

- 2000 Flickr images: These images are downloaded from http://www.flickr.com and cover 20 categories such as flag, boat, etc. Each category contains manually picked 100 images.

- 4000 online images: These images are downloaded from two websites: http://images.google.com and http://picasa.google.com and cover 40 categories. Each category contains manually picked 100 images.

We then build four image databases as follows: 1) the 2000-image database containing 2000 Flickr images; 2) the 6000-image database containing 6000 COREL images; 3) the 8000-image database containing 6000 COREL and 2000 Flickr images; 4) the 12000-image database containing 6000 COREL, 2000 Flickr, and 4000 online images.

5.2. Results and discussions

For all experiments, we randomly choose 10% database images as query images and use the rest images for testing. For each test, we calculate the average retrieval precision by evaluating the top 25 retrieved results in each RF iteration.

For the 1st experiment, we compare the performance of the proposed system using different combined features on three databases. Specifically, we test the following combined features: block-based local visual features and semantic features (i.e., region only), global visual features and semantic features (i.e., global only), 10% global and 90% local visual features and semantic features (i.e., 0.1 global+0.9 region), 50% global and 50% local visual features and semantic features (i.e., 0.5 global+0.5 region), and 70% global and 30% local features and semantic features (i.e., 0.7 global+0.3 region). Figures 5 through 7 compare the average retrieval precision of the proposed system using the aforementioned combined features on 6000-, 8000-, and 12000-image databases, respectively. They clearly show the retrieval precision increases when using more block-based local features. In other words, the proposed block-based long-term CBIR system achieves the best retrieval performance and is therefore compared with the peer systems in the other experiments.

For the 2nd experiment, we compare the proposed system with seven long-term-based CBIR systems: Hoi’s log-based [3], Han’s memory learning [4], Yin’s virtual-feature-based [5], Qi’s dynamic semantic clustering (DSC)-based [6], Xiao’s long-term cross-session learning [7], Chang’s semantic clusters (SC)-based [8], and Chang’s semantic manifold-based [9] CBIR systems. Figures 8 through 11 show the average retrieval precision of eight CBIR systems on 2000-, 6000-, 8000-, and 12000-image databases, respectively. It should be noted that the two manifold-based ranking systems [8, 9] cannot run on our computer for the 12000-image database due to its requirement of several 12000×12000 matrices. It’s clear that the proposed system achieves comparable retrieval results as the peer systems for the 2000-image database and outperforms all the other peer systems for the 6000-image, 8000-image, and the 12000-image databases. Its impressive performance clearly stands out when the size of image database increases. As shown in Fig. 11, the proposed system yields retrieval precision of 77.38% by the last iteration and improves the 2nd best CBIR system [7] by 9.73%. It should be noted that the proposed system only requires a small fraction of storage space as required by the other systems. This compact storage is obtained by the merging operation performed after each query session.
6. CONCLUSIONS

We propose a block-based long-term CBIR system which combines block-based visual features and users’ query concept-based semantic features to measure the image similarity for efficient retrieval. The major contributions are: 1) Combine two complementary features, local visual features and semantic features, to achieve impressive retrieval performance after two RF iterations; 2) Use an effective merging method to construct a compact semantic feature matrix to store retrieval patterns of historical query sessions; 3) Update semantic features of positive images labeled in each RF iteration to ensure all positive images share similar semantic concept. Our extensive experimental results show the proposed system outperforms seven peer systems on 6000-, 8000-, and 12000-image databases and achieves comparable results on the 2000-image database.

In the future, we will investigate appropriate methods to reduce the dimension of low-level features to better represent each block. We will also investigate a more effective way to keep the semantic feature matrix compact. Deploying the proposed system in web image environment will be another direction.

7. REFERENCES