Image Indexing and Similarity Retrieval Based on Key Objects

C. H. Lee
Department of Information Management
Chaoyang University of Technology

P.W. Huang
Department of Computer Science
National Chung-Hsing University

Abstract

Recently, many applications increasingly demand content-based image retrieval. To speed up the search process, a signature file containing the signatures associated with database images is frequently used as a filter to prune off non-promising images at the early stage of query processing. In this paper, we generate the signature for each picture with respect to “key” objects and propose a novel structure for organizing signatures. By using this new indexing structure, the number of signatures to be examined per query is reduced significantly. In particular, the number of bits to be examined per query is fewer as compared to the signature tree method proposed by Chen [1].

1 Introduction

The increasing availability of image markedly impacts image file and database systems. Image data are not well-defined keywords such as traditional text data used in searching and retrieval functions. Consequently, various indexing and retrieving methodologies must be defined based on the characteristic of image database. Spatial relationships represent an important feature of objects in an image. In designing image database systems, many powerful spatial knowledge structures for image representation have been proposed such as 2D C-string [4], 2D-PIR [5], 2D Be-string [6], etc. For these approaches, retrieving images similar to a query picture is done by iconic indexing based on a pre-selected spatial knowledge representation and the problem of similarity retrieval is reduced to a task of string subsequence matching which is still a very time-consuming process.

One way of improving the efficiency of similarity retrieval is to use a signature file as the spatial filter to prune unqualified images at the early stage of searching [3]. Since the purpose of the signature file is to reduce the search space in the image database, a sequential organization was assumed in most of the analytical works on signature extraction method [2]. However, searching the signature file itself may still be inefficient. There are many techniques to avoid sequential search: the bit-slice approach, the two-level approach, partition approach [7], and signature tree [1]. In this paper, we assume that the signatures in a signature file can be loaded and stored in the main memory. Based on this assumption, we proposed a novel structure for organizing a signature file. With this new indexing structure, the number of signatures to be examined per query is the minimum. Thus, the process of searching for qualified signatures becomes very efficient. The remainder of this article is organized as follows. Structure of image signature and image retrieval are given in Section 2. A novel structure for organizing signatures is presented in Section 3. Experimental results and discussions are presented in Section 4.

2 Structure of image signature and image retrieval

Our structure of image signature is based on the concept of key objects in the image database [3], where certain numbers of objects are designed as “keys”. A key object selection is proposed based on the frequency of object occurrence in the database. The idea is to capture the knowledge about these key objects as well as the spatial relationships between the key objects and other objects of the same picture. Such abstractions for a picture are embedded in a data structure called ”image signature”. There are two fields in an image signature: the ”key objects field” and the ”relation field”. Suppose there are k key objects in the database and only k’ of them appear in a given picture, then there will be k bits in the key object field with k’ bits of them set to 1s and the rest set to 0s. The signature has 5 × k bits in the relation field which is divided into k sections with 5 bits in each section. The ith section indicates the relations of all objects with the ith key object. The 5 bits in a section indicate whether the key object corresponding to that relation section has a DISJOINT, JOINT, PART, OVALP, CONTAIN, or BELONG relation with some other objects of the same picture. For example, Fig. 1 shows an example of an image signature. Suppose the 4 key objects are A, B, C, and D. Since the key object field
"1010", objects A and C must be in the corresponding picture and objects B and D are not. The first section has value "11010", which implies that key object A has at least one DISJOINT relation, one JOIN relation, and one CONTAIN relation with some other objects of the same picture. The third section has value "10100", which means that key object C has at least one DISJOINT relation and one PART.OVLP relation with some other objects.

Since a query picture can also be abstracted as an image signature, we can retrieve images from the database by comparing the image signature of the query with the image signatures. Assume that QS is the image signature associated with a query picture, DS is the image signature associated with a database picture, QSk is the key object field in QS, DSk is the key object field in DS, QSk ∩ DSk = QSk. The second type of query is to retrieve database pictures with QSk ∩ DSk = QSk. The first type of query is to retrieve database pictures such that QSk ∩ DSk = QSk.

3 Hierarchical relation graph

A Hierarchical Relation (HR) Graph is a directed graph with the following two properties: (1) A node contains a signature; (2) If Si and Sj are two signatures contained in nodes A and B, respectively, and node A is an immediate predecessor of node B, then Si ∩ Sj = Si with only one bit difference between Si and Sj.

A node of an HR graph may contain a real or virtual signature. Only the real signatures are associated with database pictures. Virtual signatures have no corresponding images in the database. An example of HR graph is shown in Fig. 2.

3.1 Constructing an HR graph

The following notations are used in the HR graph construction algorithm:

- \( S = (b_1, b_2, \ldots, b_w) \) is a \( w \)-bit signature, where \( b_k = 0 \) or \( 1 \) for \( 1 \leq k \leq w \).
- \( Z_a(S) = (z_1, z_2, \ldots, z_w) \) is a signature modified from S with \( z_k = 0 \) if \( k = a \), otherwise \( z_k = b_k \).
- \( N(S) \) is a node containing signature S.

Algorithm: Constructing an HR graph.

Input: A set of signatures \( S = \{S_1, S_2, \ldots, S_n\} \)

Output: An HR graph \( G = (V, E) \) for S.

1. \( V = \emptyset; E = \emptyset \)
2. For each \( S_i \in S \) do
   a. Generate node \( N(S_i) \) to contain \( S_i \) and mark \( N(S_i) \) as a "real signature" node.
   b. If \( N(S_i) \notin V \), then call Insert\_node\((N(S_i))\).
3. Return \( G = (V, E) \).

Function: Insert\_node\((N(S))\).

1. Add the node \( N(S) \) to V.
2. \( c = \sum_{k=1}^{w} b_k \), where \( S = (b_1 \cdots b_w) \). Let \( b_{a_1} = b_{a_2} = \ldots = b_{a_x} = 1 \).
3. For \( k = 1 \) to \( c \) do
   a. Generate node \( N(Z_{a_k}(S)) \) and mark it as a "virtual signature" node.
   b. Add the edge \((N(Z_{a_k}(S)), N(S))\) to E.
   c. Insert\_node\((N(Z_{a_k}(S)))\).

To see how the above graph construction algorithm works, let us look at example shown in Fig. 2. Assume that signature 1011 will be inserted into an empty HR graph. Because the node \( N(1011) \notin V \), we add this node to the HR graph. Since the signature "1011" has three non-zero bits at positions 1, 3, and 4, respectively, the three predecessors of node \( N(1011) \) can be obtained by changing "1" to "0" at these positions one at a time. Thus, \( N(0011), N(1001), \) and \( N(1010) \) will be added into the HR graph subsequently. Similarly, the nodes \( N(0001), N(0010) \) and \( N(1000) \) are added into the graph. Finally, the root \( N(000000) \) is added into the graph.

3.2 Implementing the HR graph

An HR graph captures the relationships of signatures in a signature file to facilitate signature matching. Assume that a signature is \( w \)-bit long. An HR graph can be implemented by an array \( A \) of size \( 2^w \). Each element \( A[i] \) of the array consists of two fields denoted by \( A[i].\text{string} \) and \( A[i].\text{tag} \), respectively. The content of \( A[i].\text{string} \) is either null or a string coded by three symbols 0, 1, and x. A query signature \( b = (b_1, b_2, \ldots, b_w) \) can be used to index the array. \( A[b].\text{string} \) records all immediate successors of node \( N(b_1, b_2, \ldots, b_w) \) in the corresponding HR graph. We replace an "x" by an "1" one at a time to find an immediate successor of the signature \( (b_1, b_2, \ldots, b_w) \). The content of \( A[b].\text{tag} \) is either "0" or "1". An "1" indicates that \( N(b_1, b_2, \ldots, b_w) \) is a real signature node in the HR graph. Otherwise, it is a virtual signature.
node. For example, assume that $A[0001].string = xxx1$. Then, we can obtain $N(1001)$, $N(0101)$, and $N(0011)$ as the children nodes of $N(0001)$ in the corresponding HR graph. The array structure corresponding to a HR graph is called the adjacency-coded representation of a signature file. Figure 3 shows an HR graph and its corresponding adjacency-coded representation.

3.3 Access method

Assume that a database contains $m$ pictures. Each picture $p_i$ (1 ≤ $i$ ≤ $m$) is associated with a signature $p^s_i$. Let $q^s$ be the signature corresponding to query picture $q$. Our goal is to find all signatures $p^s_i$ such that $q^s \cap p^s_i \equiv q^s$. We present the algorithm of searching for all qualified signatures in an HR graph implemented by the adjacency-coded representation as follows.

**Algorithm**: Signature matching based on HR graph with adjacency-coded representation.

**Input**: A query signature $q^s = (q_1, q_2, \ldots, q_w)$ and a signature file $A$ in adjacency-coded representation.

**Output**: The set $R$ of all qualified signatures.

1. $R = \emptyset$; $Uninspected.Q = \emptyset$.
2. If $A[q^s].tag = 1$, then $R = R \cup \{q^s\}$.
3. Let $C = A[q^s].string$.
4. For each symbol "x" in $C$
   a. Replace this "x" by "1" and all other "x"s by "0" to get a new string $s$.
   b. If $s \notin Uninspected.Q$, then add $s$ to $Uninspected.Q$.
5. If $Uninspected.Q = \emptyset$, then return $R$.
   Otherwise, remove an item $l$ from $Uninspected.Q$.
6. If $A[l].tag = 1$, then $R = R \cup \{l\}$.
   If $C = null$, then Goto 5.
   Otherwise, Goto 4.

Assume that we have three picture signatures $\{0100, 1100, 1001\}$ in the database. The HR graph for organizing these signatures, as well as its adjacency-coded representation $A$ for this graph is shown in Fig. 3. Let $q^s = 1000$ be a given query signature. Since $A[1000].tag = 0$, so 1000 is not a database signature. Because $A[1000].string = 1x0x$ and by elaborating "1x0x", we have $Uninspected.Q = \{1001, 1100\}$. Since $A[1001].string = null$ and $A[1100].string = null$, no more signatures will be added to the $Uninspected.Q$. When we examine the signatures in $Uninspected.Q$, we can see $A[1001].tag = 1$, so 1001 is a matched signature. Furthermore, $A[1100].tag = 1$, so 1100 is also a matched signature. After removing the two signatures 1001 and 1100 from $Uninspected.Q$, the queue becomes empty and $\{1001, 1100\}$ is returned as the set of matched signatures.

4 Experimental results

In this section, we demonstrate the efficiency of our signature file indexing method as compared to the signature tree method proposed by Yangjun Chen [1]. There are two experiments for two types of queries, respectively. In the first experiment, we simulate first type of query, which is to retrieve database pictures with $QS_k \cap DS_k = QS_k$. There are 900 pictures in the database. Thirteen popular objects are identified as "key" objects. The number of "key" objects in each picture is in the range from 5 to 9 objects. We generated 900 signatures for key object fields associated with each
picture. Five groups of queries are randomly generated. The query signatures for key object field in each group contains 3-5, 4-6, 5-7, and 6-8 bits set to 1, respectively. There are 30 query signatures in each group. Then, we calculated the average number of bits to be examined per query. Let $r_1$ and $r_2$ be the number of bits examined by Chen’s and our method, respectively. The reduction rate is defined as $\frac{r_1 - r_2}{r_1}$. The rate represents the improvement of our method over Chen’s method in terms of the total number of bits examined in order to find all matched signatures from a signature file.

Table 1 shows the experimental results of our method as compared to Chen’s method. For example, in the case of [4,6] (i.e. the number of 1’s in a query signature is between 4 and 6), the average number of bits examined per query by Chen’s method is 1100.87 while that in our method is 130.5. So the reduction rate is 88.15%. The last row shows that the average reduction rate for all five cases is 89.78%. Table 1 reveals one fact that the reduction rate is getting higher as the number of 1’s in query signatures become larger.

In the second experiment, we simulate second type of query, which is to retrieve database pictures with $QS_R \cap DS_R = QS_R$. There are four cases for different image databases. Each image database contains 1000 pictures. Thirteen popular objects are identified as the basic component patterns for composing the iconic pictures associated with the original pictures. Since we only have 13 objects in all iconic pictures, we may choose 3 of them as the key objects. The frequency of key objects occurrence is getting higher from case 1 to case 4. Thus, the number of 1’s in signatures is getting larger from database 1 to database 4. Then, we generate signature for relation field associated with each picture. The query pictures are selected 100 pictures randomly from the original database. Table 2 shows the average number of bits examined per query by using our method as compared to Chen’s method. For example, in the database 1, the average number of bits examined per query by Chen’s method is 962.36 while that in our method is 459.7. So the reduction rate is 52.23%. The last row shows that the average reduction rate for all four cases is 68.25%. Table 2 reveals one fact that the reduction rate is getting higher as the number of 1’s in database signatures become larger.

In this paper, we proposed a novel indexing structure for organizing signatures to improve the efficiency of using a signature file as a search filter. Algorithms of generating and using this indexing structure were discussed in details in this paper. Although Chen’s method is better than BSSF [1]. Our experimental results show that the number of bits examined per query by using our method is much less than that of using Chen’s method.

### References


