Abstract

Recently many algorithms have been proposed to solve the similarity retrieval problem within an image database system. Some of these algorithms may utilize schemes like superimposed code, matrix comparison, and hashing, etc.

However, in this paper, we propose an entirely different method based upon module operation to answer a similar query. First, the method transfers each image picture into a positive integer value \( V_i \). Whenever a similar query \( q \) is being issued, the algorithm automatically converts this query pattern into a positive integer value \( V_q \). Then the elements in the answer set will be the pictures \( f_j \)'s such that the remainder of each \( V_{f_j} / V_q \) equals to 0, i.e., \( V_{f_j} \mod V_q = 0 \). The total time we needed for answering a query by our retrieval algorithm is \( O(n^2) \), where \( n \) is the total number of objects of a query \( q \).

Keywords: prime number, similarity retrieval, module operation, perfect hashing.

1. Introduction

Image database is somewhat different from the conventional database, whose primary work has not only to handle the alphanumerical data but also has to process the object-oriented data associated with their spatial relationship. For example, a query may demand all the pictures associated with a tree to the left of a house or a house above a lake, etc.

Let \( S \) be a set of pictures in an image database and \( q \) be a query picture. The similarity retrieval between \( S \) and \( q \) is to filter out picture subset \( S' \) of \( S \) such that each picture of \( S' \) has some same objects with \( q \) and the spatial relationships among these same objects are preserved. In many application fields, such as computer-aided design, robotics and remote sensing of earth resources, we can find that the similar retrievals always play as an important role.

In the past few years, with the help of new devised algorithms, the response time of similarity retrieval in an image database has been speeded up, and the implementation efforts based on these algorithms turn more and more convenient.

To simplify the similarity retrieval management, S. K. Chang et al. [4] first introduces the concept of iconic index. An iconic image contains icons whose coordinates are extracted from the feature points of the objects of a real image picture and each icon is assigned an unique symbolic name. Furthermore, the spatial relationships among real objects in a picture are still kept. In other words, an iconic image is a symbolic representation of a real image that can be regarded as this logical view of a physical image. For example, picture \( f \) of Fig. 1 stands for a symbolic picture which contains four distinct objects \{house, tree, sun, lake\}. These four objects can be replaced as four icons A, B, C and D. Note that, in the following, whenever a picture is mentioned, it is regarded as a symbolic icon image implicitly. Besides, we restrict ourselves to images that have at most one icon of each type.

![Fig. 1 A symbolic picture instance.](image)

Previous approaches in solving the similarity retrieval problem all use simple table look-up in concept. The Intelligent Image Database System (IIDS) [5] first utilizes the promising 2D-string data structure [4] to decompose a picture \( P \) into a set of 2D-string rules that preserve the spatial relationship among objects of original image. Each 2D-string rule involves the use of \( <, = \) and \( > \) operators to compare the \( X \)- and \( Y \)-coordinates of the icons. To retrieve the similar pictures, the 2D-string rules matching between \( S \) and \( q \) have to be carried out.

In order to fasten the response time, Lee and Shan [7] tries to combine the superimposed coding technique with 2D-string which has wildly used among text databases. The basic idea of superimposed code method is to convert image pictures and query picture into binary signatures. Their algorithm first
prunes the images whose picture signatures do not contain the same "1"s with the query picture signature. Exhaustive search follows its work along the remaining subset.

To achieve good performance, the superimposed coding technique is applied to represent an image as a binary signature. Better performance can be expected, if distinct '1's of signatures are uniformly distributed. Chang and Lin [3] adopts this concept to develop a more fast superimposed coding technique based upon the 9DLT (9 Direction Lower Triangular) matrix [1]. In [1], C. C. Chang induces that there only exist nine spatial relationships among all possible pairs of icons. And each spatial relationship is characterized by a positive integer value. Since the spatial relation between $O_j$ and $O_i$ can be determined from the spatial relation between $O_j$ and $O_i$, therefore, only the lower left half of the 9DLT matrix contains integer values. Cluster all iconic pairs that have the same assigned integer value together, then we will have nine groups. Each group then yields their only superimposed codes. Hence, one can get better performance instead of unifying the overall iconic pairs into a unique group [7].

Later, Chang and Lee [2] combines 9DLT matrix and hashing techniques to determine similar matching with less effort. By hashing, we mean a key to address transformation that has desirable retrieval velocity. Given a key $K$, one can calculate the address that accommodates more information associated with the key $K$ by a predefined function.

In this paper, we propose an entirely different approach using "module operation" to determine the similar pictures satisfying with queries. In our method, each picture $f_i$ is transferred into a positive integer value $V_{f_i}$, and query picture $q$ is transferred into a positive integer value $V_q$, too. (Note that integer values appeared in this paper are all restricted to positive integers.) Then the elements in the answering set will be the pictures $V_{f_j}$'s such that the remainder of $V_{f_j}/V_q = 0$, i.e., $V_{f_j} \mod V_q = 0$. Further, suppose that there are $N$ pictures in an image database. In order to locate the desired pictures for a given query, we only have to repeat $N$ divide operations at most.

This paper is organized as follows. In Section 2, we introduce the detail of 9DLT matrix. Section 3 explains the concept of our approach. Sections 4 and 5 illustrate the prime numbers assignment and its refinement. The algorithm's layout is represented in Section 6. Finally, the last section states some conclusions.

2. 9DLT (Nine Direction Lower Triangular) Matrix

To transfer each real picture into a symbolic picture, the following procedures should be followed. First, by image processing and pattern recognition techniques, the objects of the original picture are recognized. Next, we enclose each object by a minimum bounding rectangle with boundaries parallel to the X-axis and Y-axis. Afterwards, a symbolic name is substituted for each object. The query picture is also regarded in the similar manner. As we have mentioned, the spatial relationships among real objects can be preserved in symbolic picture.

According to the different levels of demand in searching, type-2, type-1 and type-0 similar matching are defined [4]. Informally, picture $f$ is said to be a type-2 subpicture of picture $f$ if all the objects of $f$ indeed occur in $f$, furthermore, the spatial relationship and the distance measure among objects are preserved. $f$ is said to be type-1 subpicture of $f$ if $f$ is generated by deleting some rows and some columns of the symbolic picture $f$. $f$ is said to be type-0 subpicture of $f$ if some adjacent rows and columns of $f$ are merged together. In this paper, we only concentrate on type-1 and type-2 similar matching retrieval, since we believe these two types of matching are more useful than type-0 similar matching retrieval.

For example, in Fig. 2, $f_2$, $f_3$ and $f_4$ are all type-0 subpictures of $f_1$, $f_2$ and $f_3$ are type-1 subpictures of $f_1$, and only $f_2$ is a type-2 subpicture of $f_1$.

Given a picture, C. C. Chang [1] induced that there totally have nine different spatial relationships among objects (see Fig. 3). In Fig. 3, $R$ denotes the referenced object/icon. Code 0 represents "at the same spatial location as $R$", code 1 represents "north of $R$", code 2 represents "northwest of $R$", code 4 represents "southwest of $R$", and so on. Let $C$ be the set of nine direction codes, i.e., $C$={0, 1, 2, 3, 4, 5, 6, 7, 8}. Here each code represents the spatial relationship between two icons. A 9DLT matrix $M$ is an $n \times n$ matrix over $C$ in which $t_{ij}$, the element of $M$, is the direction code of icon $O_i$ to the icon $O_j$ for $i > j$ and $t_{ij}$ is undefined if $i \leq j$.

Reconsidering Fig. 2, the corresponding 9DLT matrix of $f_1$, $f_2$, $f_3$ and $f_4$ are depicted in Fig. 4, where matrices $M_1$, $M_2$, $M_3$ and $M_4$ correspond to pictures $f_1$, $f_2$, $f_3$ and $f_4$, respectively. Each picture corresponds to a 9DLT matrix.

Fig. 2 A symbolic picture $f_1$ and three queries $f_2$, $f_3$ and $f_4$. 
Let a triple \((O_i, O_j, r_{ij})\) denote the spatial relationship between objects \(O_i\) and \(O_j\), where \(0 \leq r_{ij} \leq 8\). The triple \((O_i, O_j, r_{ij})\) implies \((O_j, O_i, r'_{ij})\) where \(r'_{ij}\) is the inverse of \(r_{ij}\). In other words, the inverse of "north of" is "south of", the inverse of "south-west of" is "north-east of", and so on. 

Triple \((O_i, O_j, r_{ij})\) is said to be an ordered triple if \(O_i \leq O_j\) in lexical ordering. Hence, the 9DLT matrix \(M_1\) of \(f_1\) will correspond to a set \\{(A, B, 6), (A, C, 8), (A, D, 8), (B, C, 8), (B, D, 8), (C, D, 6)\} of ordered triples, and \(M_4\) corresponds to \\{(A, B, 5), (A, D, 7), (B, D, 8)\}. For simplicity, let the set of ordered triples corresponds to the 9DLT matrix \(M_i\) be \(T_i\). Since pictures can be expressed as a set of ordered triples, the similarity retrieval problem reduces to check whether the set of ordered triples of a query picture is a subset of the ordered triples of a symbolic picture in an image database. Hence, the similarity retrieval problem can be regarded as the triple matching problem. From Fig. 2, in fact, \(f_2\) is a type-1 subpicture of \(f_1\). It can be verified from Fig. 4 that the corresponding elements and their \(t_{ij}\)'s are identical between matrices \(M_2\) and \(M_1\). In other words, the ordered triple set \(T_2 = \{(A, B, 6)\}\) of \(M_2\) is the subset of \(T_1 = \{(A, B, 6), (A, C, 8), (A, D, 8), (B, C, 8), (B, D, 8), (C, D, 6)\}\) of \(M_1\). Therefore, \(f_2\) is a type-1 subpicture of \(f_1\). Consider the matrices \(M_1\) and \(M_4\), i.e., \(f_1\) and \(f_4\). We find that the corresponding \(t_{ij}\)'s are not identical, hence, \(f_4\) is not a similar subpicture of \(f_1\).

### 3. Basic Concept of Our Scheme

The main concept of this paper is to transfer each image picture \(f_i\) into a positive integer value \(V_{f_i}\). Whenever a query picture \(q\) is issued, \(q\) is transferred into a positive integer value \(V_q\), too. Therewith, employ the module operation between \(V_{f_i}\) and \(V_q\), if the remainder of \(V_{f_i} / V_q = 0\), i.e., \(V_{f_i} \mod V_q = 0\), then we make sure that \(q\) is a type-1 subpicture of \(f_i\), in other words, \(f_i\) satisfies with \(q\).

For example, four symbolic pictures \(f_1, f_2, f_3\) and \(f_4\) and a query picture \(q\) are depicted in Fig. 5. After being processed by a "black box", \(f_1, f_2, f_3\) and \(f_4\) become 286, 17, 105 and 2, respectively. At this moment, let us blind and convince ourselves that this black box enables to do this job. Query \(q\) is also fed into the black box, and we get one positive integer value 105. Employ the divide operation to calculate the remainders between \(V_{f_i}\) and \(V_q\), we then obtain

- \(V_{f_1} \mod V_q = 286 \mod 105 = 72\),
- \(V_{f_2} \mod V_q = 17 \mod 105 = 17\),
- \(V_{f_3} \mod V_q = 105 \mod 105 = 0\),
- \(V_{f_4} \mod V_q = 2 \mod 105 = 2\).

Because only the third remainder equals to 0, we predicate that the picture \(f_3\) has something to do with \(q\). Examine \(f_3\) and \(q\) more carefully, we conclude that \(f_3\) and \(q\) are indeed type-1 matching.

The readers may wonder what the black box looks like and how does it accomplish to transfer a picture into an integer value. In the following, we will disclose the cover of this black box step by step.
In fact, the major duty of the black box is to transfer all ordered triples that deduced from a 9DLT matrix into a set of distinct positive prime numbers. Since each picture $f_i$ is associated with an ordered triple set $T_i$, we will get a set of distinct prime numbers by feeding this ordered triple set into black box. Next, let us calculate the lowest common multiple of these distinct prime numbers, and address this value as the standing value $SV_i$ of $f_i$. This gives rise to the standing values 286, 17, 105 and 2. Furthermore, it is easy to verify that 286, 17, 105 and 2 are indeed constituted from the lowest common multiple of a set of distinct prime numbers. Decompose these standing values, we have:

\[
286 = 2 \times 11 \times 13, \\
17 = 17, \\
105 = 3 \times 5 \times 7, \\
2 = 2.
\]

Here 2, 3, 5, 11, 13 and 17 are prime numbers which associate to triples $(B, C, 8), (A, B, 8), (A, C, 7), (A, B, 6), (A, C, 6)$ and $(A, C, 8)$, respectively. This verifies the capability of black box. In the next section, we will introduce how to assign prime number to each ordered triple.

4. A Prime Number Assignment Method

Section 3 illustrates that each picture is transferred into a standing value that resulting from the lowest common multiple of a set prime numbers. And the prime numbers are generated from the corresponding ordered triples of pictures. However, the ordered triples are originated from the 9DLT matrices. Fig. 6 depicts a prime number assignment of the ordered triples of four image pictures of Fig. 5.

<table>
<thead>
<tr>
<th>$f_1$</th>
<th>$f_2$</th>
<th>$f_3$</th>
<th>$f_4$</th>
</tr>
</thead>
</table>

$286 = 11 \times 13 \times 2$, $17 = 17$, $105 = 3 \times 5 \times 7$, $2 = 2$.

Fig. 6 Prime numbers assignment.

Examining Fig. 6 and Fig. 7, we discover that the maximum standing value in Fig. 6 is 286, however, the maximum standing value in Fig. 7 is 182. From the practical point of view, we would rather have the maximum standing value as small as possible. Thus, the assignment of Fig. 7 is preferable to that of Fig. 6. Since the standing values are generated from the multiplication of prime numbers, without cautious assignment, some standing values may become very large while some others are very small. Furthermore, if the standing values are too large, then we may face integer overflow problem. To avoid this undesirable condition, we should mindfully assign some prime numbers to the ordered triples.

Before going further, let us define some terminologies that will make the representation more clearly. Let $P(O_i, O_j, r_{ij})$ be the frequency of the ordered triple $(O_i, O_j, r_{ij})$, that indicates the number of occurrences of this triple appears in an image database. Let $PN(O_i, O_j, r_{ij})$ be the prime number being assigned to the ordered triple $(O_i, O_j, r_{ij})$. The standing value $SV_i$ of a picture $f_i$ is the multiplication of all $PN(O_i, O_j, r_{ij})$, where $(O_i, O_j, r_{ij})$s belong to the picture $f_i$. In the following, we give Algorithm PS to assign a prime to an ordered triple of image database:

**Algorithm PS**

**Input:** A set of ordered triples.

**Output:** A set of distinct prime numbers associated to the set of ordered triples, respectively.

**Step 1:** Count the frequencies of ordered triples and sort them on decreasing order. Set all standing values of all pictures to 1.

**Step 2:** While there are unassigned ordered triples whose frequencies are greater than 1, do Step 3 to Step 6; otherwise, go to Step 7.

**Step 3:** If there are more than one different ordered triples that have the same largest frequency greater than one, then go to Step 4; otherwise, go to Step 5.

Fig. 7 New prime numbers assignment.
Step 4: Select the ordered triple belonging to one picture with the largest current standing value and assign the candidate prime number to it. Go to Step 6. Here a candidate prime number means an integer value which has not been assigned to ordered triple but its predecessor has already been assigned to an ordered triple.

Step 5: Assign the candidate prime number to the ordered triple which has the largest frequency value.

Step 6: Calculate the lowest common multiple between this new assigned prime number and the current standing value that is associated to the picture containing this new assigned ordered triple. Go to Step 2.

Step 7: While there exists unassigned ordered triple whose corresponding frequency equals to 1, do Step 8 to Step 9. Otherwise, stop executing.

Step 8: If there are more than one picture that have the same largest unassigned ordered triple, select the picture that has the largest standing value up to now, and assign the candidate prime number to the ordered triple; otherwise, assign the candidate prime number to an arbitrarily selected ordered triple of a picture that has the largest number of unassigned ordered triples.

Step 9: Calculate the lowest common multiple between this new assigned prime number and the contemporary standing value associated to the selected picture. Go to Step 7.

For instance, pictures $f_1$, $f_2$, $f_3$ and $f_4$ of Fig. 5 have eight ordered triples totally, but only triple $(B, C, 8)$ appears in both pictures $f_1$ and $f_4$. Therefore, according to the Step 2 of Algorithm PS, we assign 2 to $(B, C, 8)$. That is $PN(B, C, 8)=2$. After assigning $(B, C, 8)$, frequencies of the remaining triples are all equal to 1. Hence, according to Step 8, we select an ordered triple of $f_3$, say $(A, B, 8)$, and set $PN(A, B, 8)=3$.

Now, $SV_1 = 2$, $SV_2 = 1$, $SV_3 = 3$ and $SV_4 = 2$. The next candidate prime number is 5, according to Step 8, 5 is assigned to the ordered triple $(A, C, 7)$ of $f_3$. At this moment, $SV_1 = 2$, $SV_2 = 1$, $SV_3 = 3 \times 5$ and $SV_4 = 2$. Now, prime number 7 is ready, according to Step 8, $(A, B, 6)$ is selected. Since $SV_3 = 3 \times 5$ holds the largest standing value among $SV_1 = 2 \times 7$, $SV_7 = 1$, $SV_8 = 3 \times 5$ and $SV_4 = 2$, next candidate prime number 11 is dedicated to picture $f_3$. All the ordered triples of $f_3$ are assigned, thus $f_3$ is raced out the assignment. Candidate prime number 13 is bounded for $(A, C, 6)$ of $f_1$ according to Step 8. Afterwards, only the ordered triple $(A, C, 8)$ is left, and candidate prime number 17 is destined to be its associated prime number. At last, we have $SV_1 = 2 \times 7 \times 13$, $SV_2 = 17$, $SV_3 = 3 \times 5 \times 11$ and $SV_4 = 2$.

The spirit of Algorithm PS is to reduce the magnitude of the largest standing value. If some ordered triple has largest frequency, it should have the first priority to be assigned candidate prime number. Because this triple has occurred in several pictures, the final standing values associated to these pictures will grow larger if we assign larger prime number to it. Hence, to decrease the standing values, we should give a small prime number to this kind of ordered triples.

### 5. Refine the Prime Number Assignment

In the last section, we presented a prime number assignment that converts each image picture into a standing value. Each ordered triple is assigned a distinct prime number. Calculate the lowest common multiple of the distinct prime numbers that assigned to ordered triples, the standing value is generated. Since Algorithm PS assigns distinct prime number to each distinct ordered triple, the standing value of each picture is equal to the multiplication of these distinct prime numbers.

However, when the total number of objects of an image picture becomes large, the corresponding standing value will grow large, too. In order to reduce the standing values, we inevitably have to lessen the standing value. In this section, we introduce a Canonical Assignment that is distinct from Algorithm PS such that the distinct triples of a picture may assign powers of the same prime number. The standing value of an image picture remains the lowest common multiple of the prime numbers associated with the ordered triples belonging to this image picture.

Let there be four pictures whose ordered triple sets are $T_1 = \{(A, B, 8), (A, C, 8), (A, D, 7), (B, C, 8), (B, D, 6), (C, D, 5)\}$, $T_2 = \{(A, B, 6), (A, C, 8), (B, C, 1)\}$, $T_3 = \{(A, B, 4), (A, E, 6), (B, E, 8)\}$ and $T_4 = \{(A, B, 8), (A, C, 6), (A, D, 6), (B, C, 5), (B, D, 6), (C, D, 7)\}$. Among these pictures, triples $(A, B, 8), (A, C, 8)$ and $(B, D, 6)$ appear at more than once. $(A, B, 8)$ and $(B, D, 6)$ occur in both $f_1$ and $f_4$, while $(A, C, 8)$ appears in both $f_1$ and $f_2$.

According to Algorithm PS, we have the standing values $SV_1 = 331890$, $SV_2 = 7285$, $SV_3 = 21199$ and $SV_4 = 359898$ as in Fig. 8.

In Algorithm PS, we initially assign a candidate prime number to the ordered triples whose frequencies are greater than 1. In the Canonical Assignment, however, we assign prime numbers to the ordered triples with frequencies greater than 1 later. For all the image pictures $f_1$'s, we temporarily ignore the ordered triples with frequencies greater than 1. Start to assign $2^1, 2^2, \ldots, 2^m_1$ to the $m_1$ ordered triples with frequencies equal to 1 in a picture $f_1$, then powers of $3^1, 3^2, \ldots$ and $3^m_2$ are assigned to the $m_2$ ordered triples with frequencies equal to 1 of the second selected picture $f_j$, and etc.
The values reveal one important fact that smaller powers of prime number should be assigned to the ordered triples of one selected picture with bigger $e^\#_i$. According to these values, we determine that $f_1$ has to be assigned the smaller powers of prime number than $f_4$, and $f_2$ has also to be assigned the smaller powers of prime number than $f_3$. By this way, standing values will become less. In Fig. 9, we assign the powers of 2, 5, 7 and 3 to $f_1$, $f_2$, $f_3$ and $f_4$.

Subsequently, 11, 13 and 17 become the candidate prime numbers to $(A, B, 8)$, $(A, C, 8)$ and $(B, D, 6)$ after determining the ordered triples with frequencies equal to 1. Following the heuristic criteria, we determine the destination of prime numbers 11, 13 and 17:

1. Calculate $TN(O_i, O_j, r_{ij})$, where $TN(O_i, O_j, r_{ij})$ is the total number of unassigned ordered triples of a picture $f$ such that $f$ contains $(O_i, O_j, r_{ij})$.
2. Sort these summation values in descending order.
3. Assign candidate prime numbers to ordered triples of this sorted sequence from the beginning to the end.

Since $(A, B, 8)$ and $(B, D, 6)$ appear in $f_1$ and $f_4$, $(A, C, 8)$ appears in both $f_1$ and $f_2$, we have $TN(A, B, 8)=12 (=6+6)$, $TN(B, D, 6)=12 (=6+6)$ and $TN(A, C, 8)=9 (=6+3)$. Hence, 11, 13 and 17 are assigned to $(A, B, 8)$, $(A, C, 8)$ and $(B, D, 6)$, respectively. Prime number 17 should multiply the other five prime numbers in pictures $f_1$ and $f_4$ if $(A, B, 8)$ is assigned with 17. This value will be larger than to multiply another the other five prime numbers with 11 or 13. On the contrary, $(A, C, 8)$ whose corresponding prime number only has to multiply two prime numbers, the standing value will not affect too much if we assign 17 to $(A, C, 8)$. Thus the final standing values of $f_1$, $f_2$, $f_3$ and $f_4$ are 19448, 425, 343 and 11583.

Comparing Fig. 8 to Fig. 9, we can find that the standing values of Fig. 9 is smaller than those in Fig. 8. Amazing enough that the largest decrease is $f_3$ that we can reduce about 61 times.

In fact, each ordered triple in Fig. 9 still has distinct values but does not have distinct prime number. If object number increases, then we can use less space to store these standing values and the division operations can be carried more efficiently than Algorithm PS. Algorithm of Canonical Assignment is outlined as follows.
Algorithm CA

Input: An image database.
Output: Each ordered triple of image database is associated with a prime number.

Step 1: Compute $e_i$ and $G_i$ of each image picture.
Step 2: Cluster pictures into groups while each group contains pictures that have the same number of ordered triples.
Step 3: Sort the pictures of each group into an ascending sequence according to $e_i$.
Step 4: Select one of the unselected picture $f$ of the sorted ascending sequence of the nonempty group that has the largest $(e_i + G_i)$ value.
Step 5: Delete this selected picture $f$ from this nonempty group.
Step 6: Assign the powers of candidate prime number to the ordered triples of $f$ whose frequencies equal to 1.
Step 7: Repeat Step 4 until there is no nonempty group.

Step 8: Compute the summation values $T'(O_i, O_j, r_j)$ of the remaining $W$ ordered triples, where each $F(O_i, O_j, r_j)$ is greater than 1.
Step 9: Sort the summation values of Step 8 into descending order. Let $i = 1$
Step 10: Assign the candidate prime number to the $i$-th ordered triple in the sorted sequence.
Step 11: Let $i = i + 1$.
Step 12: Repeat Step 10 until $i > W$.

Note that the final standing value of each picture is the lowest common multiple of the corresponding prime numbers of ordered triples of this picture.

6. Algorithm

At the last section, we explore the mystery of the black box. In this section, we will examine the black box further and compute the complexity of this algorithm.

Let there be $N$ pictures in the image database and each picture has $m_i$ objects. Let query picture $q$ has $n$ objects and the total ordered triples of these $N$ pictures be $SO$.

$$SO = \sum_{i=1}^{N} (m_i - 1) \cdot m_i$$

Sort these $SO$ ordered triples of $N$ pictures, we can create a sorted table in lexical ordering. Each item consists of two entries, the first entry contains the ordered triple, and the second entry holds the prime number corresponding to the ordered triple of the first entry. Given a query picture, there are $O(n^2)$ ordered triples of query picture. To determine the prime numbers of the ordered triple set of $q$, we need $O(n^2 \log SO)$ time using binary search to fix the prime number in the predetermined static sorted table. To synthesize the standing value of query picture, we need $O(n^2)$ multiply operations. To determine which picture has the similar matching with query picture, we should need $O(N)$ divide operations. Since each picture has at least one object, $O(N)$ is usually less than $O(n^2 \log SO)$. Thus the overall time complexity of this algorithm is $O(n^2 \log SO)$.

The reader may not be difficult to find the bottleneck of our algorithm is to determine the corresponding prime number of query's ordered triples. In the following, we will utilize the perfect hashing scheme to decrease the time complexity from $O(n^2 \log SO)$ down to $O(n^2)$, where $n$ is the number of query objects/icons.

Hashing is a key-to-address transformation that is an useful data structure in determining key's location immediately. The extreme difference between binary search and hashing is that binary search usually takes $O(\log SO)$ time to determine the desired item, but hashing usually renders $O(1)$ retrieval facility. However, if hashing function has not been configured carefully, several keys may map into the same location, i.e., keys collision. In this way, the data structure may grow intricately.

If hashing function is one-to-one mapping from the key space to address space, it is a perfect hashing function. To accommodate perfect hashing scheme into similarity retrieval, each ordered triple $(O_i, O_j, r_j)$ should be hashed into a unique position in address space. In this paper, we utilize the scheme proposed by Cook and Oldehoeft [6] and Chang and Lee [2] to design a perfect hashing scheme. The major idea of this perfect hashing function is $h(O_i, O_j, r_j) = r_j + \text{associated value of } O_i + \text{associated value of } O_j$. Let appearance of ordered triple $(O_i, O_j, r_j)$ be the occurrences of $(O_i, O_j, r_j)$ within the image database.

Algorithm Perfect Hashing

Input: A set of ordered triples $(O_i, O_j, r_j)$.
Output: Associated values of query's objects.

Step 1: Compute the appearances of the symbolic icons.
Step 2: Sort the objects/icons in decreasing order according to their appearances.
Step 3: For each ordered triple $(O_i, O_j, r_j)$, interchange $O_i$ and $O_j$ if $O_j$ precedes $O_i$ in the sorted sequence of Step 2.
Step 4: Let the second object in each ordered triple be a key, sort the list of the ordered triples generated from Step 3 in descending order.
Step 5: Assign values to objects from the first ordered triple. For each cluster of triples with the same second object, determine a value assigning to the second object. (Note that for this cluster, either the first object is equal to the
second object or the first symbol precedes the second object in the object ordering. The first symbol must have been determined, thus we can map triples of group into address space.

Step 3 forces the objects with larger appearance values be put to front and the objects with smaller appearance values be put to rear. Implicitly, ordered triple will be clustered to several groups such that each group has the same second object. Hence, the value assignment in the Step 5 will assign values to objects with larger appearance values at first. Objects with smaller appearance values may have larger freedom to select a value such that collision can be avoided. Whenever we succeed to find a value assigned to the second object of some group, the addresses of the ordered triples of this group are determined subsequently. If we can not find a value to the second object of some group, we should backup to modify the value until one-to-one mapping principle is kept. Fig. 10 illustrates the value assignment using data in Fig. 8.

From Fig. 10, we perceive that Algorithm Perfect Hashing indeed enables to produce a perfect hashing function, four extra spaces are needed to accommodate fifteen ordered triples.

7. Conclusions

In this paper, an entirely new similarity retrieval algorithm is proposed. This algorithm first utilizes perfect hashing scheme to assign prime number to ordered triples. Second, calculate the lowest common multiples to determine the corresponding standing values. Third, execute the module operation to determine the similar matching between query picture $q$ and image picture $f_i$. The remainder of $V_f/V_q$ equals to 0, i.e., $V_f \mod V_q = 0$, algorithm filters out that query picture $q$ is type-1 similar subpicture of image picture $f_i$. $O(n^2)$ time complexity is binding, where $n$ is the number of query objects. Since the module operation is a common machine instruction, the implementation of our scheme is very simple and efficient.

However, in this paper, the standing value of each picture may not be minimum. Because the optimum algorithm has not been proposed but only heuristic algorithm is offered in this paper. Thus to find an optimum methodology to minimize the standing values still retains open.

References


