# A Fast Symbolic Image Indexing and Retrieval Method Based On TSR and Linear Hashing

Mehri Sadooghi Yazdi, Kiana Najafzade, and Mohsen Ebrahimi Moghaddam
Electrical and Computer Engineering Department, Shahid Beheshti University of Tehran,G.C.
Complete Address: Velanjak Ave, Tehran, Iran Phone: (98) 21-22431728 Fax: (98)21-22431804
m_moghadam@sbu.ac.ir

***Abstract***: *In recent years, image databases have grown faster; hence there are a real need for fast indexing and retrieval methods in image databases. In this paper, we proposed an approach for fast image indexing and retrieval in symbolic image databases using Triangular Spatial Relations (TSR). The indexing data structure is based on a new introduced structure and hash function. To obtain the time complexity $O(1)$ ; the linear hashing was used that has constant load factor. The experimental results were great.*

**Key Words:** *symbolic image database, indexing and retrieval, TSR, hash function, linear hashing.*

## 1. INTRODUCTION

An image database keeps a large number of pictures that are required to index truly so that they would be retrieved faster perfectly. Researchers have tried to achieve this purpose for many years. Therefore, a lot of methods have been introduced in image indexing at image databases.

Chang et al. [1] proposed a structure that is named 2D string. This structure considers any pair of objects in a symbolic image with their spatial relation in order to construct iconic indexes for images. Thereafter many modifications on 2D string were presented; Like 2D C-string [2] and 2D Z-string [3]. But these methods had high time complexity and were not invariant to image transformation specially rotation. So, different data structures were used for the sake of image indexing like db-tree [4], Object Independent Tree (OIT), Object Dependent Tree (ODT) [5] and perfect hash table [6]. The retrieval time complexity of perfect hash table is very low but it should be reconstructed in any new insertion. To overcome to this problem Shahzad mughal et al. presented 3D hash function data structure in [7]. They used 2D string scheme to represent spatial relationships among objects of image. Their method obtained triples of the form of $(S_i, S_j, R_{ij})$ where $S_i$ and $S_j$ are object's labels and $R_{ij}$ is spatial relation between these objects. Any calculated triple is an index of a 3 dimensional matrix that formed a 3D hash table. Each element of this 3D hash table points to a link list that includes the addresses of images which have that triple. Although their method had low time €complexity (i.e. $O(1)$ ) in partial search, it doesn't support total search. In [8] the authors of this paper introduced a multi level link list after 3D hash table to reduce time complexity of total search in former method to $O(1)$ .

In another work, Puthina and Guru [9] proposed an approach based on B-tree for symbolic image indexing and retrieval by spatial similarity. Their model is based on modified triangular spatial relationships (TSR) and B-tree. This model preserves TSR among the components in a symbolic image. Retrieval time complexity of this method is logarithmic and it is invariant to image transformation.

In this paper, we proposed an approach using TSR and hash function to reduce retrieval time complexity in image indexing and retrieval. Also, the linear hashing was used to guarantee expected time complexity of proposed method as *O(1)*. The presented method is robust versus rotation, translation, etc.

The rest of paper is organized as follows:

In section 2 we describe preliminary discussions about TSR, 3D hash function with multi level link list and linear hashing. Section 3 includes our proposed method. Experimentations are explained in section 4 and finally section 5 concludes the paper.

## 2. PRELIMINARIES

### *2.1. TSR*

A TSR is formally defined by connecting three non-collinear components in a symbolic image as follows [10]:Let *A, B* and *C* is any three non-collinear components of a symbolic image. Let $L_a$ , $L_b$ and $L_c$ be the labels of *A, B* and *C*, respectively. Connecting the centroids of these components mutually forms a triangle with $M_1, M_2$ and $M_3$ as the midpoints of the sides of the triangle, $\theta_2$ and $\theta_3$ are the smaller angles subtended at $M_1, M_2$ and $M_3$ respectively. Fig. 1 shows an example. The TSR among the components *A, B* and *C* is represented by a set of quadruples as

$$\{(L_a, L_b, L_c, \theta_3), (L_a, L_c, L_b, \theta_2), (L_b, L_a, L_c, \theta_3), (L_b, L_c, L_a, \theta_1), (L_c, L_a, L_b, \theta_2), (L_c, L_b, L_a, \theta_1)\}$$

This representation is unwieldy, as there are six possible quadruples for every three non-collinear components.

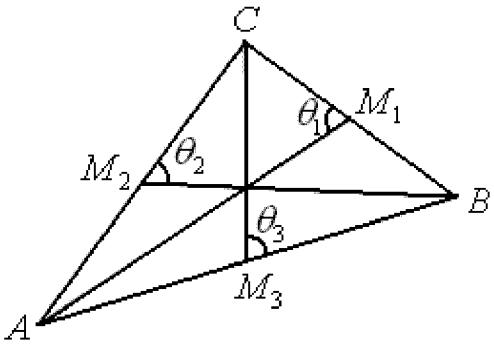


Fig. 1. Triangular Spatial Relationship

The concept of TSR is proved to be invariant to image transformations such as translation, rotation, scaling, and flipping. More details can be found in [10]. Also, for collinear components, basic TSR is modified and it is applied in the symbolic image databases [9].

### *2.2. 3D hash function with multi level link list*

The proposed method in [8] uses 2D string data structure for preserving spatial relations among iconic objects of images. This process tends to generate triples with the form of $(S_i, S_j, R_{ij})$ where $S_i$ and $S_j$ are object's labels and $R_{ij}$ is spatial relationship between them. Spatial relations are taken place into nine categories which are shown in Fig. 2.

| | | |
|---|---|---|
| 2 | 1 | 8 |
| 3 | ORp | 7 |
| 4 | 5 | 6 |

Fig. 2. Integer mapping of spatial relations

In the above figure, one object is considered as reference point and the spatial relation between it and the other objects are shown from integers 1 to 8. For example if a picture has 3 picture elements (or 3 objects), obtained triples from 2D string structure are shown in Fig. 3.

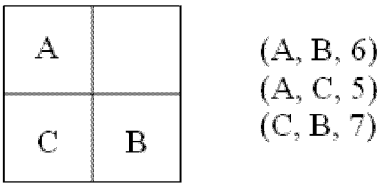


Fig. 3. A typical symbolic picture and corresponding triples

Since 2D string data structure is not invariant to image rotation, rotated forms of each triple are obtained too. Then each triple is considered as an index of a 3D matrix. Each hash table entry points to a multi level link list which includes the addresses of images that consist of this triple. The structure of each node in multi level link list is shown in Fig 4.

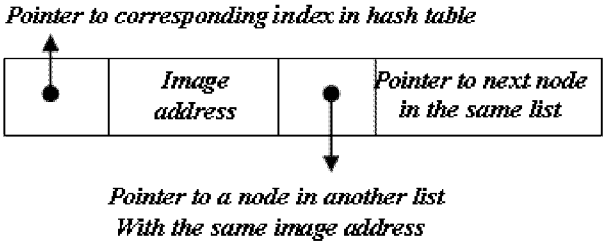


Fig. 4. Node structure in multi level link list

For a typical node p in a link list, we use p.index for showing pointer to corresponding index in hash table, p.Next_Node for showing pointer to a node in another list with the same image address and p.Link for showing pointer to next node in the same list. These abbreviations are used in the next section.

At new insertion of an image in database, its triples are extracted and its address is added to link lists that are associated to its triples. Then corresponding pointers are set, to form multi level link list. In retrieval process of total search, at first triples of query picture are extracted and after selecting smallest triple, it is used to find corresponding multi level link list in 3D hash table. Then this process searches the nodes by following pointers and finds the image address that its number of triples is equal to the number of query image's triples. In Partial Search, the procedure gets a portion of an image and finds images that are similar to it. The presented algorithm for Total Search is used for Partial Search, also.

It is showed that the retrieval time complexity of this method in partial and total search is $O(1)$ in an amortized analysis.

### *2.3. Linear hashing*

Linear Hashing is a dynamic hash table algorithm invented by Witold Litwin in [11]. It is a hashing in which the address space may grow or shrink dynamically. In general, a record in the file is found in one access, while the load factor (which is the ratio between the number of records and the number of spaces for records in the primary area that is an area to keep records) may stay practically constant.

In linear hashing, the last binary bits in the hash number are used for placing the record. That is, after calculating a large number from the key, the record is placed in a bucket according to the last *k* binary digits in the hash number. For example, if we begin with 8 buckets, we use the last three bits in the hash number to place the record. If the number ended in 000, it would be placed in the first bucket. If it ended in

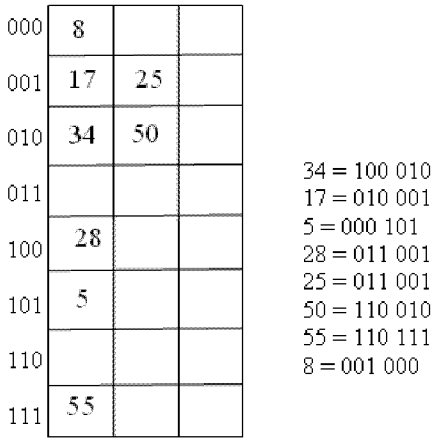


Fig. 5. Linear hashing using the last three digits of the hash number

001, it is placed in the second bucket and so on. Figure 5 shows an illustration. When table is expanded, we split an existing bucket which holds all the records that end in a particular *k* digits into two buckets using *k+1* last digits of the hash number.

The dominant note of linear hashing is that it keeps a nearly constant load factor. Thus it is space efficient and time efficient [12].

## 3. PROPOSED METHOD

In this section, the proposed method is described in detail. In this method, the modified version of TSR is used to represent the spatial relationships among objects in symbolic images that tends to generating quadruples of the form of $(L_a, L_b, L_c, \theta)$ where $L_a$, $L_b$ and $L_c$ are the labels of three objects *A, B* and *C* in a symbolic image and $\theta$ is the smaller angle subtended at the midpoint of the line that connects center of *A, B*.

These quadruples are used as indices of a table that is named hash table. $L_a$, $L_b$, $L_c$, and $\theta$ is used to find the index of table. Each hash table entry is a pointer to a multi level link list that any link list contains image addresses have the associated quadruples. Also linear hashing concept is used to keep the load factor of this hash table constant. Therefore, each quadruple becomes a pointer to a multi level link list of images contain that quadruple. The last dimension (θ) has 19 lines because we assume that θ is between 0° and 90° and it is measured with 5° accuracy.

The proposed structure for components A, B and C, shown in "fig 1" with the quadruples: $\{(L_a, L_b, L_c, \theta_3), (L_a, L_c, L_b, \theta_2), (L_b, L_a, L_c, \theta_3), (L_b, L_c, L_a, \theta_1), (L_c, L_a, L_b, \theta_2), (L_c, L_b, L_a, \theta_1)\}$ is shown in "fig 6".

When a new image is inserted in database, its quadruples are extracted and a new node is inserted at the first of corresponding link lists. After the insertion, each new node points to another node in other list that has the same image address.

Because the TSR method is invariant to rotation, scaling and transformation, only six quadruples that are defined above for each three non-linear components are sufficient.

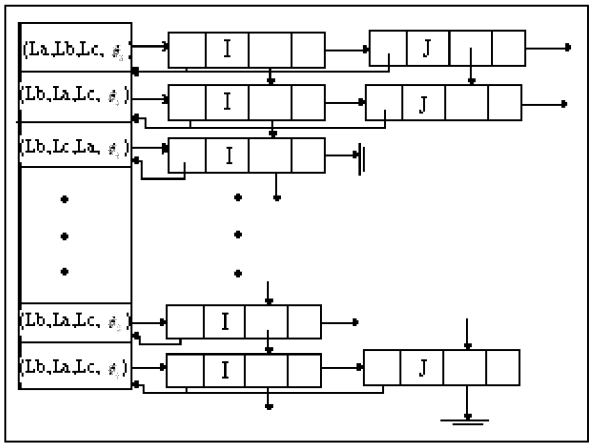


Fig. 6. Proposed structure.

There are two kind of retrieval process. The first kind is called Total Search and the second one is called Partial search. Total search gets an image and finds images that are exactly matched with input image. In Partial Search, the procedure gets a portion of an Image and finds images that are similar to this portion. The presented algorithm for Total Search is used for Partial Search also.

Also for implementation of our method we should use an auxiliary Link list N to save image addresses. This link list should have a node structure like Fig. 7.

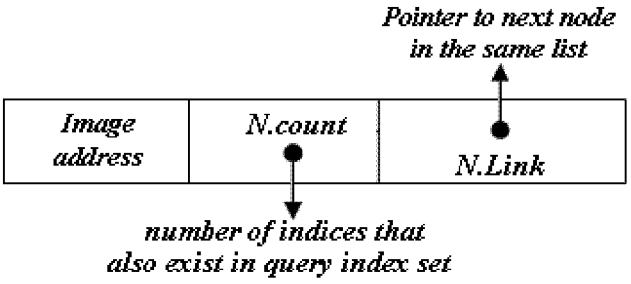


Fig. 7. Node structure of auxiliary link list

This search procedure contains following steps:

***Step 1***: Extract the quadruples of the each 3 objects with the following equations:

$$S_1 = Dist(Obj(L_{i1}), Obj(L_{i3}))$$
$$S_2 = Dist(Obj(L_{i1}), Obj(L_{i2}))/2$$
$$S_3 = Dist(Mid(Obj(L_{i1}), Obj(L_{i2})), Obj(L_{i3}))$$
$$\theta = \frac{Cos(S_1 - S_2 - S_3)}{2 * S_2 * S_3}$$

In above equations distance measure is considered Euclidean distance and *Mid* refers to the midpoint of given objects.

***Step2***. Finding the index of final hash table with Linear Hashing (its procedure described later) (for each quadruple of each 3 object one index should be found and set of query indices is created).

***Step3***: Select an index from set of query indices which is smaller than the others.

***Step4***: Use this index to find corresponding multi level link list in hash table, we refer it as L.

***Step5***: Let p as the first node of link list L then while p.Link is not null do the following steps:

5.1. N.Image_Address=p.Image_Address, q=p, counter_q=0;
5.2. if q=null then N.count = counter_q, N=N.Link, Goto *step5* with p=p.Link
5.3. if q.index exists in set of query indices then counter_q=counter_q+1;
5.4. q=q.Next_Node, Goto 5.2

***Step6***: Using auxiliary link list N we can find easily the image address that its counter is equal to query index set size for exact match.

Now we describe Linear Hashing Algorithm briefly [13]:

***Step 1:*** Transform the extracted quadruples of each three objects into a number using a simple function: $f(L_{i1}, L_{i2}, L_{i3}, \theta) = \theta * M^3 + L_{i3} * M^2 + L_{i2} * M + L_{i1}$ (M is a typical integer)

***Step2:*** This number, later on, is going to be the index of the final hash table which is searched in linear hashing.

***Step3:*** We form an *n* digit *boundary*. If the last *n* digits of the index are less than the *boundary* value, then the location is in the bucket with the last *n+ 1* digit. Otherwise, the location is in the bucket labeled with the last *n* digits.

***Step4:*** While inserting new records, if more than (*bucket factor* * *load factor*) records wanted to be inserted, then we have to do a *split*, starting from the first bucket. One bucket is added to the primary area, using the *n+1* digits to allocate these two buckets (adding 0 to the left end of one and 1 to another)

In partial search we can use the same procedure ; all the members of auxiliary link list (N) shows the matched images.

With respect to described steps above, retrieval time complexity of our method is depend on

1. Average number of nodes in link lists that is called $Lc$ .
2. The number of extracted triples of query image that called $k'$ .

Therefore, the retrieval time complexity of our method is:

$$O(Lc \cdot k')$$

## 4. EXPERIMENTAL RESULTS

In order to simulate proposed method, we used symbolic image database presented in [8]. Its authors proposed this image database which consists of 24 several images originally and grouped them into five different groups. We used this image database to evaluate the proposed method. Group1 and Group2 are images with five symbols, Group3 are images with 9 symbols and Groups 4 and 5 are images with 13 symbols. "fig.7" shows sample images from Group1 and Group2, Group3, and Group 4 and Group 5 images.

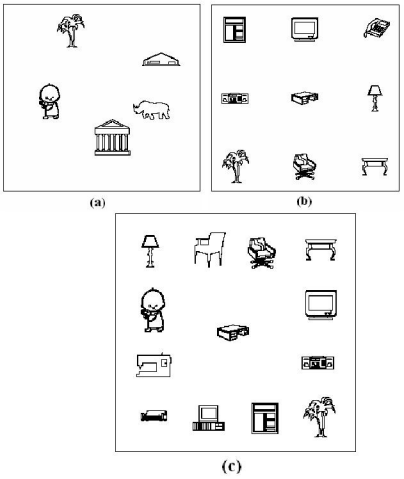


Fig. 8. (a) Sample picture of Group1 and Group2, (b) Sample picture of Group 3 and (c) Sample picture of Group4 and 5

To improve this image database, we added some transformed images, using the original images of the database and we raised the number of symbolic images in group 1, 2 to 100, in group 3, to 60 and in Group 4 and 5 to 100 symbolic images.

In *"Table 2"* we show the accuracy of our proposed method in Total and Partial search on the mentioned database.

In order to keep constant the number of images in each link list, a linear hash function that its bucket factor was 20 was used.

Table1. Experimental results of proposed method using specified

| .Image Group | Number of Extracted quadruples | Average Number of nodes in each link list (*Lc*) |
|---|---|---|
| **Group1 and Group2** | **10** | 1.02 |
| **Group3** | **84** | 1.04 |
| **Group 4 and Group5** | **286** | 1.08 |

Table2. Accuracy of proposed method in Total and Partial Search Using proposed Retrieval method on specified Image database.

| | |
|---|---|
| **Total Search** | **100%** |
| **Partial Search** | **100%** |

As a result of "*Table1*", we see that the average number of images in each multi level link list (*Lc*) almost is a fixed value. Also, the number of extracted triples in the query image ($k'$) has a constant value. Therefore, retrieval time complexity, tends to $O(1)$.

"*Table3.*" compares our methods with some other methods with regards to their time complexity and transform sensitivity.

Table3. Comparison of proposed method with several related methods

| **Applied Data structure** | **Retrieval time Complexity of Total search** | **Invariant to Image transformation** |
|---|---|---|
| **2D string** | NP | variant |
| **Db-tree** | O(log n) | variant |
| **3D hash function** | $O(n^a)$ | variant |
| **TSR with B-tree** | $O(\log^a r)$ | Invariant |
| **The proposed method** | O(1) | Invariant |

## 5. CONCLUSION

In this paper, we presented a new method for fast image indexing and retrieval in symbolic image databases. We used Triangular Spatial Relationships (TSR) for representation of spatial relations among iconic objects in images. Obtained quadruples from applying TSR, is used to form a 4D hash function.

Also we used linear hashing on this hash function to keep its load factor to a constant number. Each hash table entry points to a multi level link list that each link list includes image addresses that have the associated quadruple. Experimental results show the performance of the proposed method and we conclude that the retrieval time complexity of this method is $O(1)$.

## References


[1]. S. K. Chang, Q. Y. Shi, and C. W. Yan: Iconic Indexing of 2D Strings. IEEE Transactions on Pattern Analysis and Machine Intelligence, pami-9(3):413–428, 1987.

[2]. S.Y. Lee, F.J. Hsu: 2D-C string: a new spatial knowledge representation for image database system, Pattern recognition 23 (10) (1990) 1077–1087.

[3]. A.J.T. Lee, H.P. Chiu: 2D Z-string: a new spatial knowledge representation for image databases, Pattern Recognition Lett.doi:10.1016/S0167-8655(03)00162-4 (2003) 3015–3026.

[4]. X. Li, X. Qu: Matching Spatial Relations Using db-Tree for image Retrieval, Pattern Recognition, Fourteenth International Conference on Volume 2, Issue, 16-20 Aug 1998 Page(s):1230 - 1234 vol.2.

[5]. Y.Gu, B. Panda and K. A. Haque: Design and analysis of data structures for querying image databases, SAC 2001, Las Vegas, NV (ACM).

[6]. C.L. Sabharwal, S.K. Bhatia: Perfect hash table algorithm for image databases using negative associated values, Pattern Recognition 28 (7) (1995) 1091–1101.

[7]. M.S. Mughal, M. Nawaz, F. Ahmad, S. Shahzad, A.K. Bhatti, S. Mohsin: A 3D-Hash Function for Fast Image Indexing and Retrieval, Computer Graphics, Imaging and Visualization. CGIV apos; 07 Volume, Issue, 14-17 Aug. 2007 Page(s):341 – 348.

[8]. M. Sadooghi Yazdi and M. Ebrahimi Moghaddam: A Fast Indexing and Retrieval Method for Image Databases, International Conference on Image, System and Signal Processing (IWSSIP, Slovak, 2008.

[9]. P. Punitha, D.S. Guru: Symbolic image indexing and retrieval by spatial similarity: An approach based on Btree, Pattern Recognition (2007), doi: 10.1016/j.patcog.2007.09.012.

[10]. D.S. Guru, P. Nagabhushan, Triangular spatial relationship: a new approach for spatial knowledge representation, Pattern Recognition Lett. 22 (9) (2001) 999–1006.

[11]. Litwin, Witold , Linear hashing: A new tool for file and table addressing, Proc. 6th Conference on Very Large Databases: pp. 212-223, (1980).

[12]. CLRs: An Introduction to Algorithms, Mcgraw hill, (2001).

[13]. Betty Salzburg, File structures: an analytic approach, Prentice Hall, ISBN: 0-13-314691-X, (1988).