A Generalized Approach for Image Indexing and Retrieval Based on 2-D Strings[1]

Euripides G.M. Petrakis and Stelios C. Orphanoudakis

Department of Computer Science, University of Crete
and
Institute of Computer Science, Foundation for Research and Technology-Hellas
Heraklion, Crete, Greece

Abstract

2-D strings is one of a few representation structures originally designed for use in an IDB environment. In this chapter, a generalized approach for 2-D string based indexing, which avoids the exhaustive search through the entire database of previous 2-D string based techniques, is proposed. The classical framework of representation of 2-D strings is also specialized to the cases of scaled and unscaled images. Index structures for supporting retrieval by content, utilizing the 2-D string representation framework, are also discussed. The performance of the proposed method is evaluated using a database of simulated images and compared with the performance of existing techniques of 2-D string indexing and retrieval. The results demonstrate a very significant improvement in retrieval performance.

1 Introduction

Much attention has been given during the past few years to the design and development of Image DataBase (IDB) systems which support the archiving and retrieval of images by content [1, 2, 3]. In such systems, images are analyzed so that descriptions of their content can be extracted and stored in the IDB together with the original images. These descriptions are then used to search the IDB and to determine which images satisfy the query selection criteria. However, the problem of retrieving images by content is difficult due to reasons related to the complexity and uncertainty inherent to image analysis and interpretation tasks, the large amounts of data involved in derived image descriptions, as well as the dependence of such descriptions on the content of images relating to a specific application. Furthermore, the time efficiency of the processing and database search techniques which are applied may not satisfy the speed requirements of many IDB applications.

Prior to storage, derived image descriptions need to be appropriately represented (mapped) in database storage structures and models (e.g. relational tables). The amount of data stored, data processing and communication overhead can be reduced by using compact image representations. Furthermore, retrieval responses can be speeded up by incorporating into the IDB storage and search mechanisms efficient techniques which support the indexing of images by content.

[1]Intelligent Image Database Systems, World Scientific, S.-K. Chang, E. Jungert and G. Tortora Eds, pp. 197-218, 1996

Once image representations are stored in database storage structures, they can be indexed using the indexing mechanisms provided by the particular DBMS [4, 5]. Queries specifying constraints on object properties are then easy to be answered. Queries specifying object properties and relationships between objects (e.g. queries by example image), have to be translated into conditional statements involving constraints on object properties and relationships. However, such queries are difficult to be processed and image comparisons involve time intensive operations such as graph matching [6, 7]. The time complexity of matching increases with the number of objects in the images which are compared.

2-D strings by S.K. Chang et. al. [8] is one of a few representation structures originally designed for use in an IDB environment. The technique provides a simple and compact representation of spatial image properties in the form of two one-dimensional strings. 2-D strings can be used to resolve queries based on image content. A query is specified by providing an example image or icon. After computing its 2-D string representation, the problem of image retrieval is transformed into one of two-dimensional string matching. However, the search based on 2-D strings is exhaustive: in order to determine which images must be retrieved, 2-D string representations corresponding to all stored images are compared with a similar representation extracted from the query image.

A technique for the indexing of 2-D strings stored in an IDB has been proposed by C.C. Chang and S.Y. Lee in [9]. Specifically, 2-D strings are indexed based on representations corresponding to all pairs of objects. Each pair of objects is assigned an address (index) and entered into a hash table. Similarly, the objects contained in a given query image are taken in pairs. Each pair of query objects acts as a separate query and used to retrieve a set of images. The intersection of the retrieved sets contains images which are then compared to the original query.

In this chapter, a generalized scheme for 2-D string indexing and retrieval is proposed. Indexing is based not only on pairs of objects, but also on groups consisting of more than 2 objects. These groups are first represented by 2-D strings. An address is then computed to each one of the above 2-D strings. The representation of 2-D strings is specialized to the cases of scaled and unscaled images respectively.

The addressing scheme proposed in [9] requires that the images to be stored in the IDB are given in advance. A preprocessing step is needed to derive a hash function which is perfect (i.e. guarantees that no two pairs of objects are mapped to the same address unless they have the same properties) for this particular set of images. However, if new images are entered into the IDB, the hash function ceases to be perfect. The addressing scheme used in this work requires no preprocessing, the images need not be given in advance and it is not affected (i.e. remains perfect) by the number of images which are entered in the IDB.

The performance (given in terms of the average size of the space searched and of the average retrieval response time) of the proposed methodology is compared against the performance of the 2-D string matching algorithms of [8] and of the indexing mechanism of [9]. To our knowledge, the performance of the techniques under consideration has not been studied elsewhere in the literature. Indexing based on pairs of objects incurs a significant overhead due to excessive processing of intermediate query results. Retrieval responses are speeded up significantly when indexing is based on groups consisting of more than 2 objects. However, retrieval response times and storage space are traded off. An IDB of 1000 simulated images has been used as a testbed for all comparisons.

This work completes and extends previous work by the authors: a similar addressing scheme and a general approach to reprepresenting spatial image properties has been presented in [10]. String representations taking into account any number of object properties and the inclusion relationships between objects have been proposed. The effectiveness of such representations in retrieving images by content has been investigated. However, certain retrievals incur a significant processing overhead and are relatively slow. In this chapter, string image presentations and similar indexing structures are combined with known 2-D string matching algorithms to provide a more time efficient but less space demanding approach of image indexing and retrieval.

The rest of this chapter is organized as follows: an overview of 2-D strings is presented in Section 2. A specialization of 2-D strings in the cases of scaled and unscaled images is presented in Section 3. Our approach to indexing 2-D strings is presented in Section 4. The proposed search strategy and evaluations of the performance of retrievals based on exhaustive search, the indexing scheme of [9] and our approach are discussed in Section 5, followed by conclusions in Section 6.

2 Overview of 2-D Strings

To produce the 2-D string representation of a given image, the image is segmented and the positions of all objects (i.e. their center of mass) are projected along the x and y directions. By taking the objects from left to right and from below to above, two one-dimensional strings are obtained forming the 2-D string representation of the image. The objects themselves are represented by values corresponding to classes or names. For example, the objects contained in the image of Figure 1 can be of one of 4 classes, namely a, b, c and d, and are numbered from 0 to 6. Their indices are shown on the left of their classes. Object indices (numbers) and object classes will be used alternatively in this chapter to denote objects.

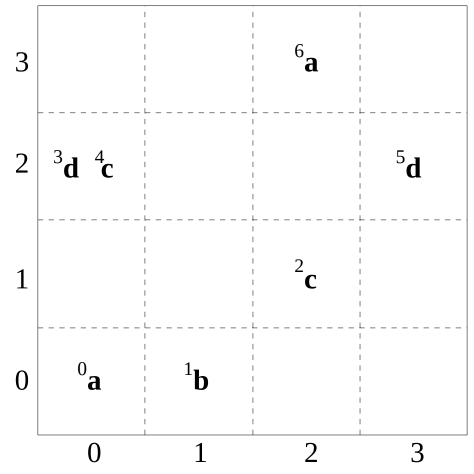
The 2-D string representation of a given image may take various forms, namely "reduced", "augmented" etc. Details about these forms and the transformations between them can be found in [8]. The reduced 2-D string representation of the image of Figure 1 is

$$(u,v) = (a\ d\ c < b < c\ a < d,\ a\ b < c < d\ c\ d < a) \qquad \text{(reduced 2-D string).}$$

The symbol "$<$" denotes the "left/right" relationships in string u and the "below/above" relationships in string v. Notice that objects having the same x or y projection can be written in any order. For example, objects 0, 3 and 4 have the same x projection and can be written as acd or cad or dca etc. To obtain the augmented 2-D string, we substitute each object in string v by its position in string u. For example, object a is first in u and it is substituted by 0 in v, object b is fourth in string u and it is substituted by 3 in v etc. The augmented 2-D string representation of the image of Figure 1 is

$$(u,v) = (a\ d\ c < b < c\ a < d,\ 0\ 3 < 4 < 1\ 2\ 6 < 5) \qquad \text{(augmented 2-D string).}$$

In this chapter, we make use of a form, which will henceforth be called "expanded 2-D string": the augmented 2-D string representation (u,v) of a given image is expanded into a form (z,r,s) of three one-dimensional strings, where z contains the names or the classes of all objects in the same order as they appear in u, while strings r and s contain the projections of all objects along the horizontal and the vertical direction respectively. The values in strings r



Augmented 2-D string:

$$(u, v) \;=\; (adc < b < ca < d,\; 03 < 4 < 126 < 5)$$

Figure 1: A symbolic image (left) and its corresponding augmented 2-D string (right).

and s will henceforth be called "ranks". In this example, r and s take values in the range $[0, 3]$. However, the range for r and s need not be always equal. For example, object b in the image of Figure 1 has rank 1 along the x axis and rank 0 along the y axis. Objects c and d in the cell with coordinates $(0, 2)$, share the same position and have the same ranks. The expanded 2-D string representation of the image of Figure 1 is

$$(z, r, s) = (a\ d\ c\ b\ c\ a\ d,\ 0\,0\,0\,1\,2\,2\,3,\ 0\,2\,2\,0\,1\,3\,2) \qquad \text{(expanded 2-D string).}$$

2-D strings are an adequate representation of spatial image properties in the case of images consisting of nonoverlapping objects (i.e. their minimum enclosing rectangles do not intersect) having simple shapes. Various extensions of 2-D strings, such as the 2-D G strings and the 2-D C strings have been proposed in [11] and [12] respectively, and deal with situations of overlapping objects with complex shapes. However, such representations are not as simple and compact as the original 2-D string representation, nor can they be produced very efficiently. Extensions of 2-D strings for the representation of three-dimensional scenes can be found in [13].

2.1 Image Similarity Based on 2-D Strings

The similarity between two images (e.g. a query and a retrieved image), whose content is represented by two-dimensional strings, can be determined based either on exact or approximate matching techniques. In the first case, every object in the query image has to be associated to at least one similar object (i.e. an object having the same name or class) in the retrieved image and the matched objects in these two images must have exactly the same relationships. In the case of a successful match, there is at least one 2-D subsequence within the second 2-D string matching the query. An algorithm must determine the existence of a match and, in the case of a successful match, it must find all the matching subsequences. Three algorithms for 2-D string matching and three types of similarity criteria, namely type-0,1,2, have been defined in [8]. The first algorithm, referred to as $2DmatchA$, is a more general one and may be used with all types of similarity criteria. The other two, referred to as $2Dquery1$ and $2Dquery2$, may be used with only the first two types of similarity criteria.

The effectiveness of 2-D string based representations in retrieving images by content has been investigated in [10]. It has been shown that, in certain cases, such representations are not tolerant to small variations of property values and become unstable (i.e. change drastically). Due to unstable representations, retrievals may be proven to be inaccurate yielding images that a user would not consider similar to the query. In addition, not all images which a user would consider similar to a query are retrieved.

In 2-D string matching, a query and its retrieved subsequence must have exactly the same representation (exact match retrievals). However, image retrieval is not an exact process. Images are rarely identical. To be effective, database search must seek for images having approximately similar representations with the query. Then, appropriate distance (or similarity) measures must be adopted. An approach to dealing with such issues by making use of alternative representations of image content (e.g. attributed graphs) is discussed in [14].

The definition of similarity is less strict in the case of approximate matching techniques, since the two-dimensional string of the query need not be an exact subsequence of the second string. The approximate similarity between two 2-D strings can be determined based either on a maximum likelihood or a minimum distance criterion. In the first case, the similarity is determined based on the longest 2-D string subsequence that the two 2-D strings under consideration have in common. Regarding image retrieval, the problem is then to retrieve the most similar image from a set of stored images. This is the one having the longest common subsequence with the query among the images stored in the IDB. The problem of finding the longest common subsequence between two 2-D strings is transformed into a problem of finding the maximal complete subgraphs (cliques) of a given graph. Therefore, the 2-D string longest common subsequence problem has nonpolynomial time complexity. The problem of finding the longest common subsequence between 2-D strings is treated in [15]. The problem of finding the longest common subsequence between 2-D C strings is treated in [16].

The similarity between two 2-D strings can also be determined based upon a minimum distance criterion. For example, the minimum distance between two 2-D or 2-D C strings can be defined as the cost of the minimum cost transformations required in order to transform the first string to the second. Algorithms for determining the minimum distance between 1-D strings do exist [17, 18] and have to be generalized to the case of two dimensional strings.

In this chapter, we focus our attention on 2-D strings and on exact match retrievals. We make use of the original algorithms for 2-D string matching of [8] and we propose a generalized approach for indexing and retrieval based on 2-D string subsequences extracted from all stored images. However, to increase the accuracy of retrievals, techniques of indexing 2-D G strings or 2-D C strings need to be developed. To our knowledge, such techniques have not been proposed so far. The proposed approach may be regarded as a first step in this direction.

2.2 Correctness of 2-D String Matching

The algorithm $2DmatchA$ allows "false drops" in certain cases [19] (i.e. there may exist instances in the answer set which do not actually match the query). The conditions of occurrence of false drops are discussed in [19] and an algorithm which avoids false drops has been proposed. False drops may occur with type-0 and type-1 matching while, false drops cannot occur with type-2 matching. Moreover, the original algorithms for 2-D string matching are, in certain cases, inexact (i.e. miss matched subsequences). These algorithms have been corrected in [20] to list all matched subsequences and have been extended to take into account any number of object properties and the inclusion relationships between objects.

3 Extensions to 2-D Strings

So far, we have assumed that objects which are close enough to each other may be assigned the same position and the same rank. This is possible in cases where images are at a fixed scale. However, when images are scaled with respect to each other, there is no reference distance for comparing the relative distances between objects. Therefore, objects which are close enough to each other may not be assigned the same position or have the same rank. Therefore, there is a need to specialize the representation of 2-D strings to the cases of scaled and unscaled images respectively.

Without loss of generality, we require that objects are ordered. Ordering is mainly required for indexing, but it also helps us to derive representations which are easier to understand and use. In the following, two criteria for ordering image objects are presented and discussed. The first ordering criterion can be used even in cases where images are scaled with respect to each other, while the second ordering criterion can be used only in cases where images are found at a fixed scale.

3.1 First Ordering Criterion

Let $(a[0], a[1], \ldots, a[n-1])$ be the set of image objects. For any two objects $a[i]$, $a[j]$, $i, j \in [0, n-1]$ with centers of mass $(cg_x[i], cg_y[i])$ and $(cg_x[j], cg_y[j])$ respectively, either $a[i]$ is a "predecessor" of $a[j]$, which is written as $a[i] \prec a[j]$, or $a[i]$ is a "successor" of $a[j]$, which is written as $a[i] \succ a[j]$. Specifically, the first ordering criterion is written as follows:

$$\forall\, i, j \in [0, n-1] \begin{cases} a[i] \prec a[j], & \text{if } cg_x[i] < cg_x[j] \text{ OR } cg_y[i] < cg_y[j] \text{ if } cg_x[i] = cg_x[j]; \\ a[i] \succ a[j], & \text{otherwise.} \end{cases} \quad (1)$$

We assume that no two objects have the same center of mass. However, cases where objects have the same center of mass are very rare and may occur only when objects contain other objects. By applying this ordering criterion to the objects contained in an image, a permutation string p is obtained which is the ordered sequence of indices corresponding to the above objects. In particular, string p corresponds to the sequence of objects produced by projecting their positions along the x axis and by taking them from left to right. Henceforth, p will be used to characterize the image itself. The ordered sequence of objects corresponding to the example image of Figure 1 is $p = (0\ 3\ 4\ 1\ 2\ 6\ 5)$. By substituting each object in p by its name or class, we obtain the string z of the expanded 2-D string representation. Specifically, $z = (a\ d\ c\ b\ c\ a\ d)$.

The expanded 2-D string representation makes use of strings r and s representing the ranks of objects along the x and y directions respectively. The first ordering criterion guarantees that objects are ordered from left to right, while no two objects may be assigned the same position and have the same rank. Therefore, string r takes the form $r = (0, 1, 2, \ldots n-1)$. A string s

corresponding to the ranks of objects along the y direction is obtained by computing, for each object in p, the number of objects which are below it. To compute s we proceed as follows: first, a second permutation string $p^{'}$ is produced by projecting the positions of objects along the y axis and by taking them from below to above. In particular, string $p^{'}$ is produced by ordering objects according to the following rule:

$$\forall\, i, j \;\in\; [0, n-1] \quad \begin{cases} a_i \prec a_j, & \text{if } cg_y[i] < cg_y[j] \text{ OR } cg_x[i] < cg_x[j] \text{ if } cg_y[i] = cg_y[j]; \\ a_i \succ a_j, & \text{otherwise.} \end{cases} \tag{2}$$

For example, the permutation string p' corresponding to the image of Figure 1 is $p' = (0\ 1\ 2\ 3\ 4\ 5\ 6)$. The rank $s[i]$ of the i-th object in p equals its position in string p'. Specifically, $s[i]$ is computed as follows:

$$s[i] = j \quad \Longleftrightarrow \quad p[i] = p'[j], \quad 0 \leq i, j < n. \tag{3}$$

The string s corresponding to the ordered sequence $p = (0\ \ 3\ \ 4\ \ 1\ \ 2\ \ 6\ \ 5)$ is $s = (0\ \ 3\ \ 4\ \ 1\ \ 2\ \ 6\ \ 5)$. Finally, the expanded 2-D string representation corresponding to the example image of Figure 1 and the first ordering criterion is

$$(z, r, s) = (a\ d\ c\ b\ c\ a\ d,\ 0\ 1\ 2\ 3\ 4\ 5\ 6,\ 0\ 3\ 4\ 1\ 2\ 6\ 5).$$

Representations derived by the application of the first ordering criterion are both translation and scale invariant (i.e. images translated or scaled with respect to each other result in the same representation). Translation invariance is assured, since only relative positions are taken into account in determining the order of objects. Similarly, scale invariance is assured, since no distance criterion is used.

3.2 Second Ordering Criterion

The image area is partitioned by an $M \times N$ rectangular grid. The position of an object with index i, where $i \in [0, n-1]$, is defined by the pair of ranks $(r[i], s[i])$, where $r[i] \in [0, M-1]$ and $s[i] \in [0, N-1]$, corresponding to the coordinates of the grid cell containing its center of mass. Equivalently, the position of an object is represented by a single value $R[i]$ which is computed as a function of $r[i]$ and $s[i]$ as $R[i] = r[i] \cdot N + s[i]$. For any two objects $a[i]$, $a[j]$, $i, j \in [0, n-1]$, with values $R[i]$, $R[j]$ respectively, the second ordering criterion is written as

$$\forall\, i, j \;\in\; [0, n-1] \begin{cases} a[i] \prec a[j], & \text{if } R[i] < R[j] \text{ OR } z[i] < z[j] \text{ if } R[i] = R[j]; \\ a[i] \succ a[j], & \text{otherwise.} \end{cases} \tag{4}$$

Objects sharing the same grid position are ordered based on their class values. We assume that an order can be defined over the set of names or classes (e.g. $a < b < c \ldots$). Objects sharing the same grid position and having the same name or class cannot be ordered. However, this situation is rather rare, especially when grids are dense (e.g. $M, N > 3$), when objects have various sizes (i.e. a large object is less likely to have the same position with other objects), they are outside one another, and when the number of different classes is large (e.g. greater than 3).

The ordered sequence of objects corresponding to the example image of Figure 1 is $p = (0\ \ 4\ \ 3\ \ 1\ \ 2\ \ 6\ \ 5)$. In this case $M = N = 4$. Object 4 is before object 3 in p, since $c < d$.

The string R corresponding to p is $(0\ 2\ 2\ 4\ 9\ 11\ 14)$. By substituting each object in p by its name or class, we obtain the string z of the expanded 2-D string representation. Specifically, $z = (a\ c\ d\ b\ c\ a\ d)$. Finally, the expanded 2-D string representation corresponding to the image of Figure 1 and the second ordering criterion is

$$(z, r, s) = (a\ c\ d\ b\ c\ a\ d,\ 0\,0\,0\,1\,2\,2\,3,\ 0\,2\,2\,0\,1\,3\,2).$$

Translation invariance of the derived representation is achieved by a simple coordinate transformation: let α and β be the minimum $r[i]$ and $s[i]$ values respectively. The ranks of each object i along x and y directions become $r[i] - \alpha$ and $s[i] - \beta$ respectively.

4 Indexing on 2-D Strings

An image is decomposed into groups of objects called "image subsets". All image subsets from 2 up to a prespecified size K_{max} are produced; these are $\sum_{k=2}^{\min(n, K_{max})} \binom{n}{k}$. For example, when $K_{max} = 6$, 57 subsets are produced from the image of Figure 1. The number of image subsets which are produced becomes very large especially when $K_{max} > 5$ and $n > 10$. Therefore, to reduce space demands, K_{max} must take low values. Methods for specifying K_{max} are discussed in Section 5.

An image is indexed based on the set of image subsets derived from it. First, the expanded 2-D string representation corresponding to all image subsets is computed. Once the expanded 2-D string representation of an image subset of size k has been derived, we compute two addresses namely $I_k^{r,s}$ and I_k^z corresponding to spatial relationships and object classes respectively.

When the first ordering criterion is used, $I_k^{r,s}$ is computed based on string s. A string s completely characterizes the spatial relationships between the objects contained in an image subset. String s defines a permutation over the set $\{0, 1, \ldots k-1\}$, since no two objects have equal ranks. Thus, string s may take $k!$ different values. An ordered image subset p is mapped to a unique address $I_k^{r,s}$ in an address space of size $D_k^{r,s} = k!$ by computing the rank (order) of r in a listing of permutations. The ranking algorithm we used is that by Johnson and Trotter [21]. For example, the ordered image subset $p = (0\ 1\ 2\ 5)$ derived from the image of Figure 1 has $s = (0\ 1\ 2\ 3)$ and $I_k^{r,s} = 0$, while $D_4^{r,s} = 24$.

When the second ordering criterion is used, the spatial relationships between the objects contained in an ordered image subset are completely characterized by string R. The number of different ways for placing k objects on an $M \times N$ rectangular grid, is equal to the number of the "l-part compositions of k elements", where $l = M \cdot N$. Therefore, the number of different values a string R may take is equal to $\binom{k+l-1}{k}$. An ordered image subset p is mapped to a unique address $I_k^{r,s}$ in an address space of size $D_k^{r,s} = \binom{k+l-1}{k}$ by computing the rank of string R in a listing of compositions [22]. For example, the ordered image subset $p = (0\ 1\ 2\ 5)$ derived from the image of Figure 1 has $R = (0\ 4\ 9\ 14)$ and $I_k^{r,s} = 2555$, while $D_4^{r,s} = 3060$.

Finally, an address I_k^z corresponding to object classes is computed based on string z. A string z consists of k elements and each one may take q different values, where q is the number of object classes. A string z may be regarded as one of the "q-base representations of k elements"; these are $D_k^z = q^k$. An ordered image subset p is mapped to a unique address I_k^z in an address space of size D_k^z by computing the rank of string z in a listing of the k-base representations of

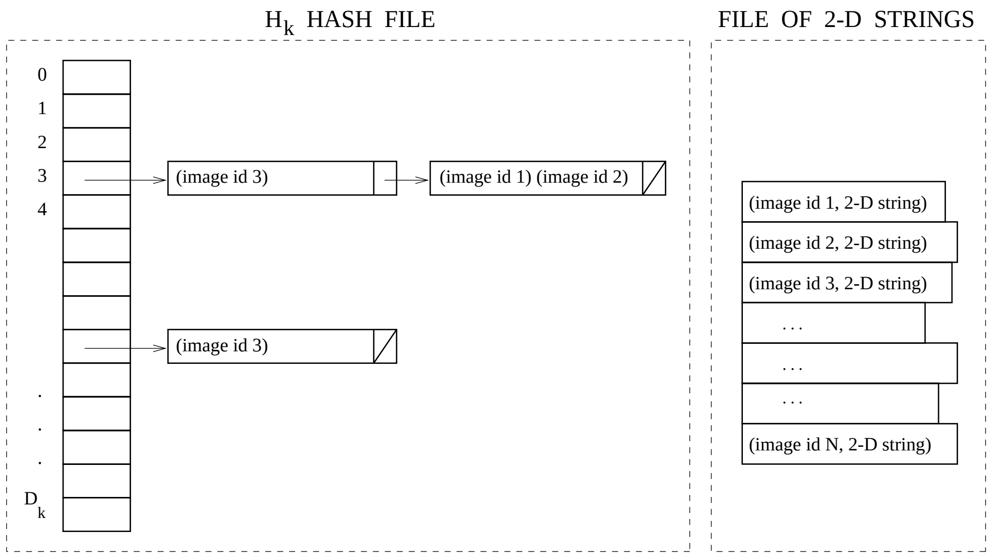


Figure 2: Proposed file structure. Only one of the $(H_2, H_3, \ldots H_{K_{max}})$ hash files is shown.

k elements. For example, assuming 4 classes ($q = 4$), namely a, b, c and d, the ordered image subset $p = (0\ 1\ 2\ 5)$ has $z = (a\ b\ c\ d)$ and $I_4^z = 228$, while $D_4^z = 256$.

Once the indices $I_k^{r,s}$ and I_k^z have been computed, a 2-D string corresponding to an image subset of size k is mapped to a unique address I_k in an address space of size $D_k = D_k^{r,s} \cdot D_k^z$ as follows:

$$I_k = I_k^{r,s} + D_k^{r,s} \cdot I_k^z, \quad 2 \leq k \leq K_{max} \tag{5}$$

The pair $(I_k^{r,s}, I_k^z)$ is considered to be the representation of I_k in the "mixed radix system" $(D_k^{r,s}, D_k^z)$. I_k is unique, in the sense that only image subsets having exactly the same 2-D string, and therefore the same indices $I_k^{r,s}$ and I_k^z, are mapped to the same address I_k.

The proposed addressing scheme guarantees that only image subsets having the same 2-D string representation are mapped to the same index. Therefore, hashing is perfect and remains perfect regardless of the number of images which are entered in the IDB. The addressing scheme discussed in this section can be extented to take into account any number of properties representing global object characteristics (i.e. area, roundness, orientation etc.), as well as the inclusion ("inside/outside") relationships between the objects contained in a given image or image subject. For further details, the reader is referred to [10].

An index (e.g. an image identifier) to the image from which the image subset has been derived is then entered into a hash table of size D_k. The storage space consists of data pages of fixed capacity. Overflows are handled by creating linked lists of data pages. The IDB consists of a set $(H_2, H_3, \ldots H_{K_{max}})$ of such index structures corresponding to subsets consisting of $2, 3, \ldots, K_{max}$ objects respectively.

The 2-D string representations of the original images are stored in a separate indexed file using the image identifiers as keys. Figure 2 shows the file structure of the data on the disk. Only one hash file ($H_k, 2 \leq K_{max}$) with address space D_k holding indices to three images is shown.

5 Image Retrieval

We distinguish between, “direct access” queries corresponding to $2 \leq m \leq K_{max}$ and “indirect access” queries corresponding to $m > K_{max}$, where m is the number of query objects and K_{max} is the maximum size of image subsets used for indexing.

Direct access queries: The 2-D string representation and the index I_m corresponding to the query are computed first. The query address the H_m hash file. The 2-D strings corresponding to all images having indices stored in the list of data pages pointed by I_m are retrieved from the file of 2-D strings. These are images matching the query (i.e. each image contains at least one 2-D subsequence matching the query). To produce all 2-D subsequences matching the query, all retrieved 2-D strings are compared with the 2-D string of the query using a 2-D string matching algorithm.

Indirect access queries: The query is decomposed into $\rho = \binom{m}{K_{max}}$ subsets, thus creating ρ new queries. Each of these queries performs as a direct access query on the $H_{K_{max}}$ file. Their answer sets, namely $S_0, S_1, \ldots S_{\rho-1}$ are derived and their intersection $S = S_0 \cap S_1 \ldots \cap S_{\rho-1}$ is obtained. S consists of indices corresponding to candidate images matching the original query (i.e. there may exist images in S not matching the query). To determine which images match the original query and to produce all 2-D matching subsequencies, the 2-D strings corresponding to all images in S are retrieved from the file of 2-D strings and compared with the 2-D string of the query using a 2-D string matching algorithm.

5.1 Performance Evaluation

Before producing the 2-D string representation of a given image, the image must first be segmented into disjoint regions corresponding to image objects. It is assumed that objects are not hidden or overlapped with others so that their positions can be computed correctly and their classes can be identified (by a human expert or by applying a computer algorithm). Producing such segmentations from images of a real application is a very hard task and is beyond the scope of this chapter.

To evaluate the performance of the alternative indexing and retrieval techniques under consideration we make use of an IDB consisting of 1000 simulated images: a random number generator has been used to produce 1000 2-D strings corresponding to images containing between 4 and 10 objects. The positions of objects in a 3×3 rectangular grid have been generated at random. Objects are distinguished in 3 categories ($q = 3$). Each object is assigned a class at random. All images are considered to be at a fixed scale and the second ordering criterion has been used. The IDB has been implemented on a magnetic disc connected to a SUN SPARCstation ELC.

The access methods which are compared are: (a) The original method proposed by S.K. Chang et. al. [8]. An exhaustive search through the entire database of 2-D strings is performed using one of the 2-D string matching algorithms mentioned in Section 2. All these algorithms, namely $match2DA$, $2Dquery1$ and $2Dquery2$, have been implemented as in [20] and support either type-0, type-1 or type-2 matching. (b) Indexed search based on the technique proposed by C.C. Chang and S.Y. Lee [9] in combination with the above algorithms. Indexing is based on

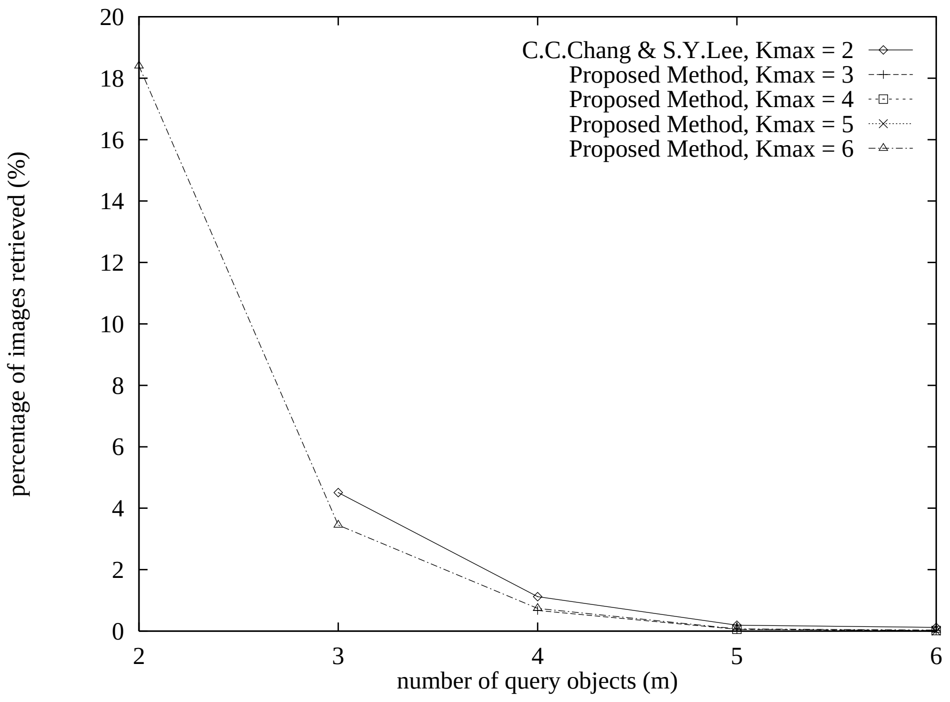


Figure 3: Average size of space searched as a percentage of images retrieved, plotted against the number of query objects for (a) $K_{max} = 2$ following C.C. Chang and Y.S. Lee and (b) for $K_{max} \in [3, 6]$ according to the proposed method.

pairs of objects ($K_{max} = 2$). (c) Indexed search using our method in combination with 2-D string matching. Indexing is based on groups of objects having size greater than 2 ($3 \leq K_{max} \leq 6$).

The performance of each of the above access methods is given in terms of (a) the retrieval response time and of (b) the size of the space searched. The size of the searched space is given as a percentage of images which are retrieved and compared with the query. To obtain average performance measures, for each value of m ranging from 2 to 6, 20 image queries have been applied and the average performance of queries specifying an equal number of objects has been computed.

The average size of the answer set as a function of the number of query objects and of K_{max} is shown in Figure 3. The shape of the curves can be justified as follows:

The size of the answer set returned decreases with m. Queries address hash tables holding subsets of equal size, or at most, of size K_{max}. However, the size of the hash table addressed increases with the size of stored image subsets. Similarly, the size of the hash table holding subsets of size K_{max} increases with K_{max}. The size of an address space increases also with q (the number of object classes).

As the size of the address space increases, the subsets are distributed over larger address spaces and less image subsets are stored per index. Therefore, queries specifying more objects result in smaller searched spaces. For example, the address space has size 405 for $m = 2$, 4455 for $m = 3$ and 40095 for $m = 4$ (we assumed $M = N = 3$ and $q = 3$). Queries specifying two objects ($m = 2$) retrieve approximately $18,5$% of the images stored in the IDB. The size of the searched space drops to less than 5% for queries specifying more than 2 objects ($m > 2$). For $m > 5$, most queries return empty answer sets.

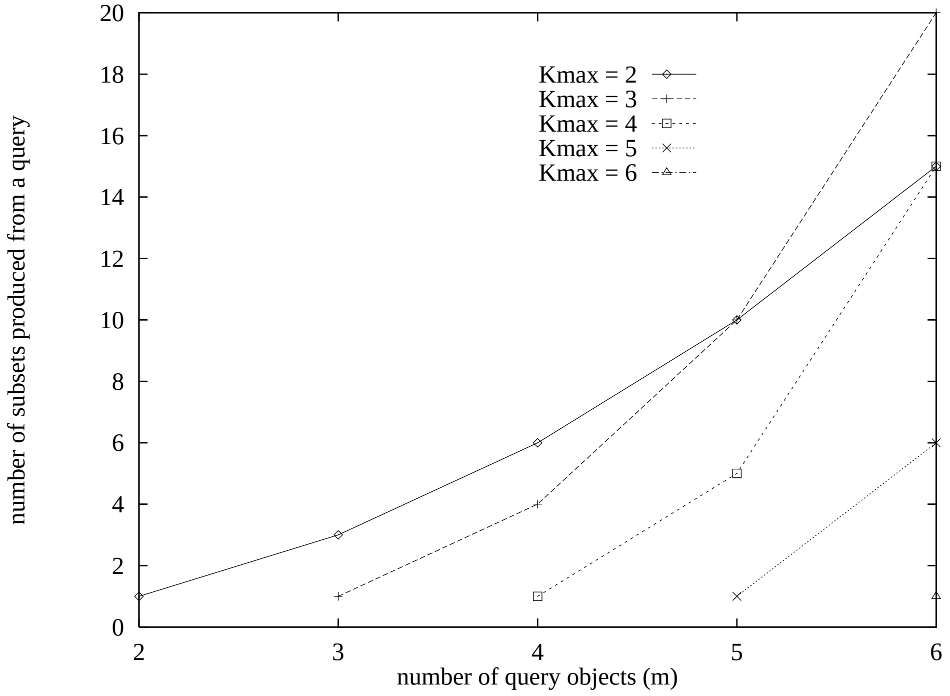


Figure 4: Number ρ of intermediate queries produced from a query specifying m objects, as a function of m and of the maximum size of image subsets K_{max}.

Indirect access queries ($m > K_{max}$) return answer sets which may contain images not matching the query (false drops). A 2-D string matching algorithm must then be applied to eliminate the false drops and to produce all sequences of objects matching the original query. For any m, the number of false drops equals the difference between the sizes of the answer sets which are obtained in response to direct and indirect access queries respectively. As observed in Figure 3, indirect access queries specifying three objects ($K_{max} = 2$, $m = 3$), yield approximately 1% false drops. The number of false drops is minimized for $m > 3$ and $K_{max} > 2$. For greater values of K_{max} the intermediate queries which are produced become more specific and return smaller answer sets; their intersection set has even smaller size and becomes approximately equal to the answer set which is obtained in response to direct access queries specifying the same number of objects.

Indirect access queries incur a significant processing overhead. The overhead increases both with the number ρ of intermediate queries and with the size of the answer sets which are obtained in response to these queries. The number of intermediate queries as a function of m and K_{max} is shown in Figure 4. For a given query specifying m objects, ρ takes its maximum value for $K_{max} = 2$ or for $K_{max} = 3$, while it is minimized for $K_{max} > 3$. In addition, as observed in Figure 3, the size of an answer set decreases with K_{max}. For example, if $K_{max} = 2$, queries specifying 5 objects ($m = 5$) have to retrieve and process $\rho = 10$ answer sets each consisting of 18.5% of the total number of images stored in the IDB. If $K_{max} = 4$, 5 answer sets are produced each holding less than 1% of the total number of images stored in the IDB. Therefore, to minimize the overhead and to speed-up retrievals, a value of K_{max} greater than 2 must be selected (e.g. $K_{max} = 4$).

The average retrieval response time corresponding to indexed and exhaustive search utilizing the $2DmatchA$ is shown in Figure 5. Indexed search results always in faster retrieval response

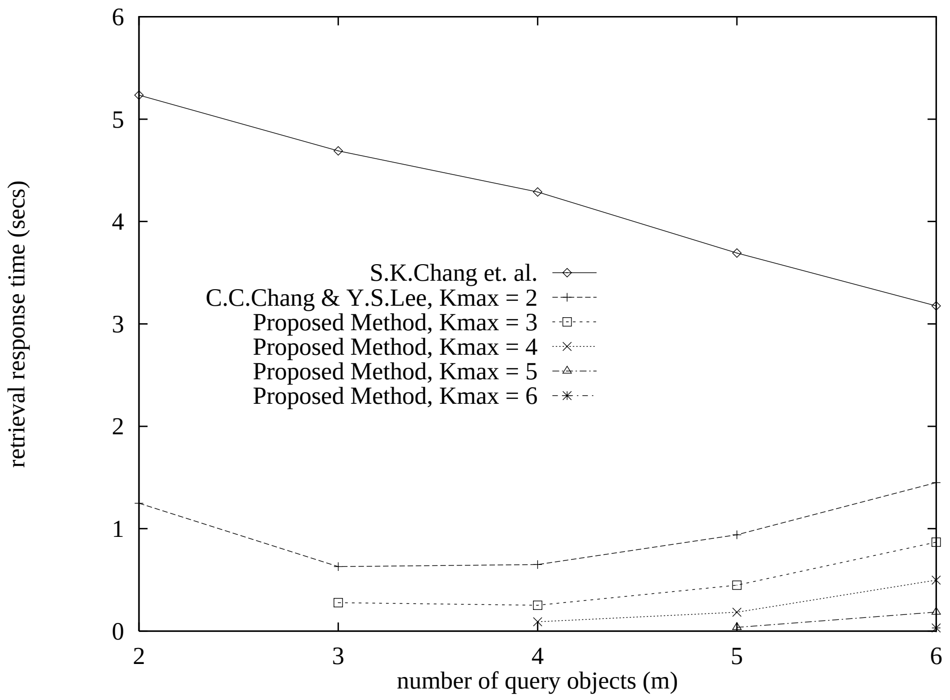


Figure 5: Average retrieval response times as a function of the number of query objects, utilizing $2DmatchA$ algorithm and corresponding to (a) exhaustive search following S.K. Chang et. al., (b) indexed search with $K_{max} = 2$ following C.C. Chang and S.Y. Lee and (c) indexed search with $K_{max} \in [3, 6]$ according to the proposed method.

times. Retrievals become faster with K_{max}. However, as observed in Figure 4, as m increases, the number ρ of intermediate queries specifying K_{max} objects increases too, the processing overhead increases and retrievals are slowed down.

The average retrieval response times corresponding to indexed and exhaustive search utilizing $2Dquery1$ or $2Dquery2$ algorithm are shown in Figure 6. The time responses of $2Dquery1$ are similar, on the average, to those obtained by applying $2Dquery2$. For queries specifying more than K_{max} objects, the processing overhead grows excessively. For $K_{max} < 5$, indexed search may result in slower retrieval response times than those obtained when the entire database of 2-D strings is searched. An exhaustive searching through the entire IDB is preferred in this case, unless we take $K_{max} > 4$.

5.2 Observations - Optimization Techniques

Retrieval response times and store space are traded off: any attempt at speeding up time responses by storing subsets of greater size result in more demanding space requirements. By optimizing the processing (i.e. retrievals and set intersections) of indirect access queries, retrieval responses can be further speeded up and K_{max} may take lower values. So far, retrievals and intersections have not been optimized.

We attempted to minimize the processing overhead by applying, instead of ρ, only one intermediate query. The time responses obtained utilizing the $2DmatchA$ algorithm are shown in Figure 7. Similar, results are obtained when $2Dmatch1$ or $2Dmatch2$ is applied. Although

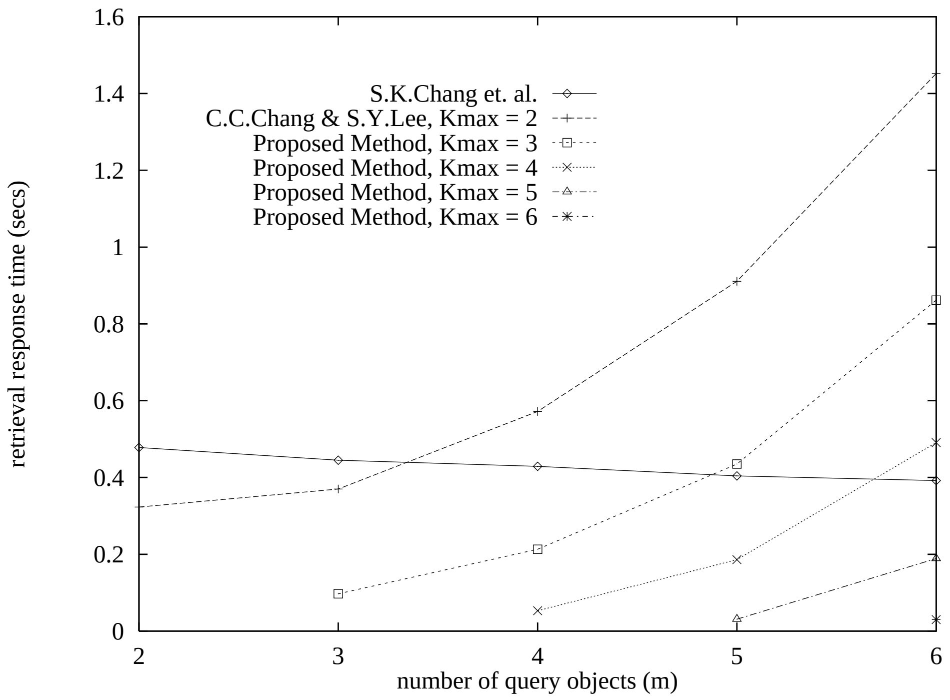


Figure 6: Average retrieval response times as a function of the number of query objects, utilizing $2Dquery1$ or $2Dquery2$ algorithm and corresponding to (a) exhaustive search following S.K. Chang et. al., (b) indexed search with $K_{max} = 2$ following C.C. Chang and S.Y. Lee and (c) indexed search with $K_{max} \in [3, 6]$ according to the proposed method.

the size of the answer sets returned in response to this query is, on the average, greater than that obtained when all ρ queries are applied, retrieval responses are speeded up significantly, since no intersections are necessary. The time responses do not decrease with m since the size of the answer set obtained for all $m > K_{max}$ is the same. A value of 3 or 4 for K_{max} can be used in this case.

Whether retrievals are always speeded up by applying all or just one intermediate query is an open question. There may exist cases, (e.g. specific kinds of images) where the maximum speedup is encountered for some particular number ρ' of intermediate queries between 1 and ρ.

Two are the important parameters that must be specified: the maximum size of images subsets K_{max} and the number p' of intermediate queries. To specify K_{max} and ρ' a prototype IDB consisting of images which are characteristic of the application under consideration has to be used and an analysis of performance has to be performed.

In the experiments discussed in this chapter, retrievals are faster compared to retrievals in [10]. Specifically, indirect access queries in [10] incur a significant processing overhead: all images in S have to be retrieved, their representations have to be computed and matched with the query. By maintaining the indexed file of 2-D strings, the above processing overhead is eliminated. As a consequence, K_{max} takes lower values and the space demands are lower.

The shape of the curves corresponding to database search using the original 2-D string matching algorithms can be justified as follows: (a) given a query specifying m objects, matching with images consisting of less than m objects is rejected immediately (early rejection). As m increases, the number of early rejections increases too. Notice, that all images contain

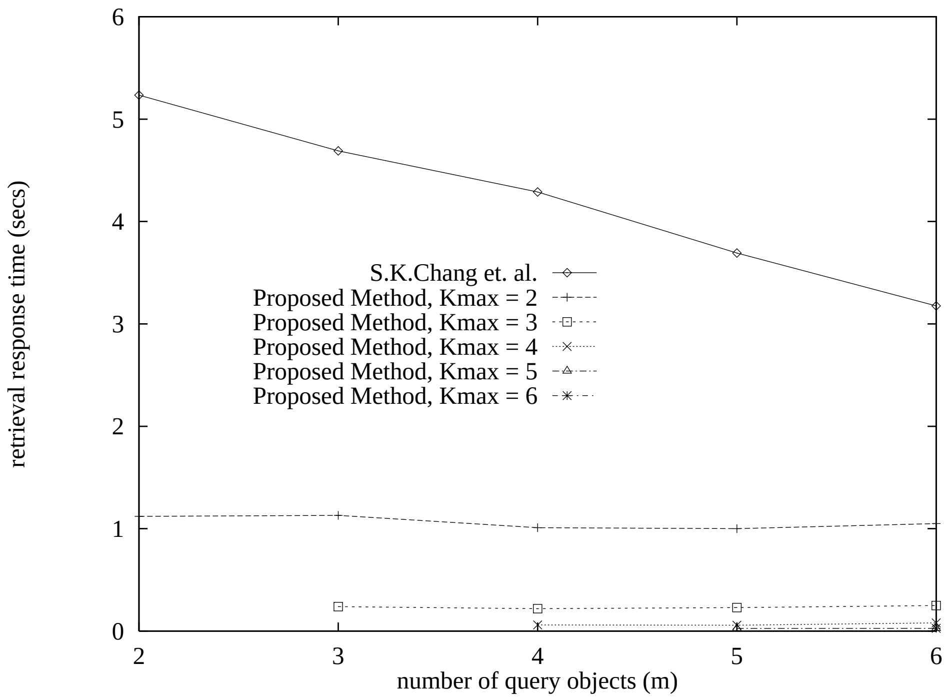


Figure 7: Average retrieval response times as a function of the number of query objects, utilizing $2DmatchA$ algorithm and corresponding to (a) exhaustive search following S.K. Chang et. al., (b) indexed search with $K_{max} \in [2, 6]$ utilizing one query subset.

between 4 and 10 objects, while all queries specify between 2 and 6 objects. Therefore, time responses become faster with the number of query objects. $2DmatchA$ algorithm, in particular, makes use of termination conditions (see [8]) which are satisfied faster as queries become more specific. Algorithms $2Dquery1$ and $2Dquery2$ result in faster time responses than $2DmatchA$ due to their lower time complexity.

6 Conclusions

In this chapter, a new method which supports the efficient indexing of 2-D strings stored in an IDB has been proposed and its performance has been compared with the performance of three 2-D string matching algorithms proposed in [8] and with the performance of the indexing method proposed in [9].

Our approach to indexing 2-D strings is based on the idea of computing addresses to equal size 2-D strings corresponding to image subsets derived from all stored images. All subsets up to a prespecified size K_{max} are considered. Searching through the entire IDB is avoided and retrievals are speeded up significantly when $K_{max} > 2$. The time responses become faster for greater values of K_{max}. Methods for the specification of K_{max} have also been discussed. However, any attempt at speeding up time responses by storing subsets of greater size would result in more demanding average space requirements. This is a trade off which must be carefully considered in the context of specific applications.

The addressing scheme used is perfect (i.e. no two subsets are mapped to the same address unless they have the same properties) and, in contrast to the addressing scheme of [9], remains perfect regardless of the number of images which are entered in the IDB. We have shown extensive experimental results analyzing the performance of the indexing technique presented in comparison to the performance of the previous approaches.

This work completes and extends the work of others. In particular, the technique of [9] may be regarded as a special case of the proposed indexing scheme when the maximum size of image subsets is 2 ($K_{max} = 2$). In addition, the work of [10] has been extended to take advantage of the effectiveness of known 2-D string matching algorithms and a more general, more time efficient but less space demanding approach to image indexing and retrieval has been proposed.

Indexing techniques with lower space demands have to be developed. Furthermore, attempts must be made to extend this method to include the indexing of 2-D C string representations which has been shown to be better suited for applications where objects are overlapping and have complex shapes. Retrievals based on 2-D C strings are less efficient compared to retrievals based on 2-D strings and indexing could speed up time responses significantly.

Acknowledgement

We would like to thank Prof. Stavros Christodoulakis for valuable contributions in this work.

References

[1] Hideyuki Tamura and Naokazu Yokoya. Image Database Systems: A Survey. *Pattern Recognition*, 17(1):29–49, 1984.

[2] Petros Kofakis and Stelios C. Orphanoudakis. Image Indexing by Content. In M. Osteaux et. al., editor, *A Second Generation PACS Concept*, chapter 7, pages 250–293. Springer-Verlag, 1992.

[3] Shi-Kuo Chang and Arding Hsu. Image Information Systems: Where Do We Go From Where? *IEEE Transactions on Knowledge and Data Engineering*, 4(5):431–442, 1992.

[4] Shi-Kuo Chang and King-Sun Fu. A Relational Database System for Images. In Shi-Kuo Chang and King-Sun Fu, editors, *Pictorial Information Systems*, pages 288–321. Springer-Verlag, 1980.

[5] Stelios C. Orphanoudakis, Euripides G. Petrakis, and Petros Kofakis. A Medical Image DataBase System for Tomographic Images. In *Proceedings of Computer Assisted Radiology, CAR89*, pages 618–622, Berlin, June 1989.

[6] Martin A. Fischler and Robert A. Elschlager. The Representation and Matching of Pictorial Structures. *IEEE Transactions on Computers*, c-22(1):67–92, 1973.

[7] Linda G. Shapiro and Robert M. Haralick. Structural Discriptions and Inexact Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(5):504–519, 1981.

[8] Shi-Kuo Chang, Qing-Yun Shi, and Cheng-Wen Yan. Iconic Indexing by 2-D Strings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(3):413–428, May 1987.

[9] Chin-Chen Chang and Suh-Yin Lee. Retrieval of Similar Pictures on Pictorial Databases. *Pattern Recognition*, 24(7):675–680, 1991.

[10] Euripides G.M. Petrakis and Stelios C. Orphanoudakis. Methodology for the Representation, Indexing and Retrieval of Images by Content. *Image and Vision Computing*, 11(8):504–521, October 1993.

[11] Shi-Kuo Chang, Erland Jungert, and Y. Li. Representation and Retrieval of Symbolic Pictures Using Generalized 2-D Strings. In *SPIE Proceedings, Visual Communications and Image Processing*, pages 1360–1372, Philadelphia, November 1989.

[12] Suh-Yin Lee and Fang-Jung Hsu. 2D C-String: A New Spatial Knowledge Representation for Image Database Systems. *Pattern Recognition*, 23(10):1077–1087, 1990.

[13] Gennaro Costagliola, Genoveffa Tortora, and Timothy Arndt. A Unifying Approach to Iconic Indexing for 2-D and 3-D Scenes. *IEEE Transactions on Knowledge and Data Engineering*, 4(3):205–222, June 1992.

[14] Euripides G.M. Petrakis and Christos Faloutsos. Similarity Searching in Medical Image Databases. *IEEE Transactions on Knowledge and Data Engineering*, 9(3):435–447, May/June 1997.

[15] Suh-Yin Lee, Man-Kwan Shan, and Wei-Pang Yang. Similarity Retrieval of Iconic Image Databases. *Pattern Recognition*, 22(6):675–682, 1989.

[16] Suh-Yin Lee and Fang-Jung Hsu. Spatial Reasoning and Similarity Retrieval of Images using 2D C-Sstring Knowledge Representation. *Pattern Recognition*, 25(3):305–318, 1992.

[17] Robert A. Wagner and Michael J. Fischer. The String-to-String Correction Problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, January 1974.

[18] Patrick A. V. Hall and Geoff R. Dowling. Approximate String Matching. *ACM Computing Surveys*, 12(4):381–402, December 1980.

[19] John Drakopoulos and Panos Constantopoulos. An Exact Algorithm for 2-D String Matching. Technical Report 021, Institute of Computer Science, Foundation for Research and Technology - Hellas, Heraklion, Greece, November 1989.

[20] Euripides G.M. Petrakis. *Image Representation, Indexing and Retrieval Based on Spatial Relationships and Properties of Objects*. PhD thesis, University of Crete, Department of Computer Science, March 1993. Available as Technical Report FORTH-ICS/TR-075.

[21] Dennis Stanton and Dennis White. *Constructive Combinatorics*, chapter 1. Springer-Verlag, 1986.

[22] Edward M. Reingold, Jurg Nievergelt, and Narsingh Deo. *Combinatorial Algorithms*, chapter 5. Prentice Hall, 1977.