ELSEVIER

Pattern Recognition Letters 19 (1998) 1241–1255

Pattern Recognition Letters

Multidimensional interval filter: A new indexing method for subpicture query of image retrieval

Man-Kwan Shan *, Suh-Yin Lee

Institute of Computer Science and Information Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, HsinChu, Taiwan, ROC

Received 23 June 1997; received in revised form 26 June 1998

Abstract

In this paper a new indexing method, called *multidimensional interval filter*, is proposed to speed up processing of subpicture query. The basic idea is the transformation of spatial information of each image into a multidimensional rectangle. Processing of subpicture query becomes that of rectangle containment query which can be further speeded up by some well-developed spatial access methods such as R-Trees. © 1998 Elsevier Science B.V. All rights reserved.

Keywords: 2D string; Content-base image retrieval; Image databases; Spatial access method

1. Introduction

Content-based image retrieval is one of the important issues in the design of multimedia information system, digital library and visual information system. Approaches of recent works on content-based image retrieval include retrieval by color, by shape, by texture, by sketch and by spatial constraints (Gudivada and Raghavan, 1995). Image retrieval by spatial constraints retrieves images based on the spatial relationships among the objects in images. This paper only focuses on retrieval by spatial constraints.

Approaches of image retrieval by spatial constraints were originated from the *2D String* approach (Chang et al., 1987). In this approach, objects and their spatial relationships in an image are represented as a spatial data structure, 2D String. Processing of subpicture query is achieved by 2D subsequence matching. However, usually there are a great number of images in an image database. It is essential to develop an indexing method to avoid exhaustive 2D subsequence matching.

In this paper, a new indexing method for subpicture query is proposed. The basic idea of this indexing method is to prune off a large amount of unqualified images by a fast filter mechanism. The spatial information of each image is transformed to a multidimensional rectangle. Those images whose corresponding rectangles do not contain the query rectangle are filtered out. The beauty of the proposed filter mechanism lies in that rectangle containment query can be speeded up by some well-developed spatial indexing structures, such as R-Trees, TV-Trees (Guttman, 1984; Lin et al., 1994).

* Corresponding author. Fax: +886-35-724176; e-mail: mkshan@info4.csie.nctu.edu.tw.

The remainder of this paper is organized as follows. In Section 2, a brief review of 2D String approaches is given. Section 3 describes the proposed indexing method. In Section 4, experimental results are presented. The conclusions are presented in Section 5.

2. Review of 2D String approaches

The 2D String describes objects and their spatial relationships according to projection of the image along x- and y-axis direction, respectively. The first part of the 2D String describes the left–right relationship while the second part of that describes the bottom–top relationship. A permutation function may be added to the 2D String to prevent ambiguity when there are multiple identical objects in the image. For example, the symbolic picture I_0 in Fig. 1 can be represented as the 2D String ($A < B = D < C$, $B < A < D < C$, 2134).

Because a symbolic picture is represented as a 2D String, a picture query can also be specified as a 2D String. The problem of subpicture query then becomes the problem of 2D subsequence matching. Chang et al. (1987) have defined type-0, type-1 and type-2 2D subsequence as follows.

Definition 1. A String U' is a type-i 1D subsequence of string U, if (1) U' is contained in U and (2) if $a_1w_1b_1$ is a substring of U', a_2 and b_2 are symbols of U, a_1 matches a_2 in U and b_1 matches b_2 in U, then

(type-0) $r(b_2) - r(a_2) \geqslant r(b_1) - r(a_1)$ or $r(b_1) - r(a_1) = 0$,

(type-1) $r(b_2) - r(a_2) \geqslant r(b_1) - r(a_1) > 0$ or $r(b_2) - r(a_2) = r(b_1) - r(a_1) = 0$,

(type-2) $r(b_2) - r(a_2) = r(b_1) - r(a_1)$,

where $r(x)$, the rank of symbol x, is defined as one plus the number of "<" preceding this symbol x.

Definition 2. The 2D String (U', V') is a type-i 2D subsequence of the 2D String (U, V) if U' is a type-i 1D subsequence of U and V' is a type-i 1D subsequence of V.

For example, in Fig. 1, I_1, I_2 and I_3 are all type-0 subpictures of I_0, I_1 and I_2 are type-1 subpictures of I_0. Only I_1 is a type-2 subpicture of I_0. The problem of 2D subsequence matching has been proven to be NP-complete (Tucci et al., 1991).

Since the invention of 2D String, some variants have been proposed to represent more complex spatial relationships among nonzero sized objects of symbolic pictures. There are generalized 2D G-String (Chang et al., 1988), 2D C-String (Lee and Hsu, 1991), 2D B-String (Lee et al., 1992), 2D C^+-String (Huang and Jean, 1994), 2D H-String (Chang and Li, 1988), Adaptive 2D H-String (Chang and Lin, 1996) and 2D N-String (Chou et al., 1997), et al. The basic concept of our proposed indexing method can be applied to any of these variants. Therefore, for the sake of clarity, we only consider the original 2D String representation.

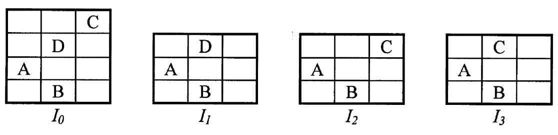

Fig. 1. Example of type-0, type-1, type-2 subpicture query.

3. The proposed indexing method

The basic idea of the proposed indexing method is a fast filter mechanism. The spatial information of each image is transformed to a multidimensional rectangle. The spatial information of the query image is also transformed to a multidimensional rectangle. Those images whose corresponding rectangles do not contain the query rectangle are filtered out. Only those images whose rectangles contain the query rectangle are processed further by 2D subsequence matching. Rectangle containment query can be speeded up by some well-developed spatial indexing structures such as R-Trees.

Of course, the filter mechanism may incur *false drops*. False drops denote those images, which are qualified by the rectangle containment matching, but actually are not qualified by 2D subsequence matching. False drops cause unnecessary 2D subsequence matching and should be minimized as small as possible.

There are two steps for the transformation of spatial information into a rectangle. The first step transforms the spatial information of each image to a set of spatial strings. The second step transforms each set of spatial strings into a rectangle. We first introduce the definition of the spatial string.

Definition 3. *A* type-*i* 1D *spatial code V_{AB}* is a code describing the spatial relationship between symbols A and B in the 1D String, whereas

$$\begin{array}{lll}
\text{(type 0)} & V_{AB} = \text{“0”} & \text{if } r(A) = r(B), \\
 & V_{AB} = \text{“1” and } V_{AB} = \text{“0”} & \text{if } r(A) < r(B), \\
 & V_{AB} = \text{“2” and } V_{AB} = \text{“0”} & \text{if } r(A) > r(B), \\
\text{(type 1)} & V_{AB} = \text{“0”} & \text{if } r(A) = r(B), \\
 & V_{AB} = \text{“1”} & \text{if } r(A) < r(B), \\
 & V_{AB} = \text{“2”} & \text{if } r(A) > r(B), \\
\text{(type 2)} & V_{AB} = \text{“0”} + \mathrm{str}(r(A)–r(B)) & \text{if } r(A) = r(B), \\
 & V_{AB} = \text{“1”} + \mathrm{str}(r(A)–r(B)) & \text{if } r(A) < r(B), \\
 & V_{AB} = \text{“2”} + \mathrm{str}(r(A)–r(B)) & \text{if } r(A) > r(B),
\end{array}$$

where $r(X)$ is the rank of symbol X, symbol “+” denotes the string concatenating operator, $\mathrm{str}(Y)$ is a transformation function which returns the string form of integer Y.

Definition 4. A type-*i* 2D *spatial string T_{AB}* of objects A and B is a string by concatenating symbol A, B and type-*i* spatial character V^x_{AB} and V^y_{AB}, where the alphanumeric order of the object A is smaller than or equal to that of the object B, V^x_{AB}, V^y_{AB} is the spatial code along the x-dimension and the y-dimension, respectively.

Definition 5. A type-*i spatial string collection $P(I)$* for an image I is defined as the collection $\{T_{AB} \mid \forall A, B \in I\}$.

Note that it is permitted that there may exist duplicate copies of the spatial string in the spatial string collection.

Example 1. Given an image represented as 2D String ($A < B = C$, $B < A < C$), the type-0, type-1, type-2 spatial string collections are (type-0): {‘*AB*02’, ‘*AB*01’, ‘*AB*12’, ‘*AC*01’, ‘*AC*10’, ‘*AC*11’, ‘*BC*01’}, (type-1): {‘*AB*12’, ‘*AC*11’, ‘*BC*01’}, (type-2): {‘*AB*1121’, ‘*AC*1111’, ‘*BC*0012’}.

The methodologies for the fast indexing method of the three types of subpicture query are all the same. Therefore, for the clarity of explanation, we only deal with the type-1 subpicture query in the following. Other types of subpicture query can be processed in the same way as that of type-1 subpicture query. In the following description, the term "subpicture query" denotes the type-1 subpicture query.

It is obvious that if a query image Q is a subpicture of an image I in the image database, then the spatial string collection of Q, $P(Q)$, is a subset of $P(I)$. The reverse is not always true. If $P(Q)$ is a subset of $P(I)$, it is possible that the query image Q is not a subpicture of database image I. This occurs when there are multiple same objects in the image. However, since the proposed method is a filter mechanism which prunes most, not all, of the unqualified images, we can apply subset testing for the indexing of subpicture query.

Having transformed the process of subpicture query into that of subset matching, the next step tries to avoid exhaustive subset testing of all the spatial string collection. Some strategies may be employed. The first strategy, *interval filter*, assigns each spatial string in the image database a unique number. Each spatial string collection $P(I)$ is represented as an interval $R(I)$ which denotes the range of the assigned numbers of spatial strings contained in $P(I)$. Given a query image Q, if $R(I)$ does not contain $R(Q)$, then $P(Q)$ is not a subset of $P(I)$ and I must be an unqualified image.

The assigned numbers of the spatial strings can be determined arbitrarily, as long as the numbers are assigned uniquely.

Example 2. Assume that in the image database, there are eight images shown in Fig. 2. The 2D Strings corresponding to these pictures are I_0: ($C < A < B$, $B < C < A$, 312), I_1: ($C < A < B$, $A < C < B$, 213), I_2: ($A = B < C$, $B < C < A$, 231), I_3: ($C < B < A < B$, $B < B < C < A$, 4213), I_4: ($C < A < B < C$, $A < C < C < B$, 2413), I_5: ($A = B < C = C$, $B = C < C < A$, 2341), I_6: ($C < B < A$, $B < C < A$, 213), I_7: ($C < B < A$, $A < C < B$, 312). The spatial string collection corresponding to each image is shown in the second column of Table 1. Assume that each spatial string in the image database is assigned a unique number shown in the third column of Table 2, the third column of Table 1 lists the interval for each image.

It is not necessary to assign numbers to those spatial strings which do not exist in the example image database. These absent spatial strings are not shown in Table 2. Given the query image Q represented as 2D String ($C < B < A$, $B < C < A$), then $P(Q) = \{$'AB22', 'AC22', 'BC21'$\}$ and $R(Q) = (5–13)$. I_2 is filtered out by *interval filter*. Actually, I_3 and I_6 are qualified images. I_0, I_1, I_4, I_5 and I_7 are false drops.

The filtering effect of *interval filter* may be improved by the second strategy, *multidimensional interval filter*. Instead of assigning a unique number, we can assign each spatial string a unique multidimensional vector in the Euclidean vector space. Therefore, each spatial string collection $P(I)$, the spatial information of image I, is represented as a *multidimensional interval* $R(I)$. In terms of geometry, each spatial string is represented as a multidimensional point, each spatial string collection is represented as a multidimensional rectangle. With multidimensional interval filter, the false drop probability is lowered down.

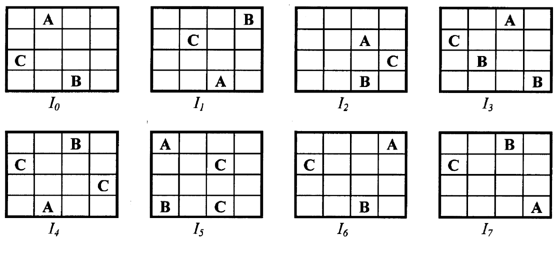

Fig. 2. Symbolic pictures of eight images in an example image database.

Table 1
Spatial string collections and intervals of images for the example image database

Image	Spatial string collection	Interval filter	2-dimensional interval filter	2-dimensional interval filter with clustering	2-dimensional interval filter with clustering after ordering
I_0	AC22, BC21, AB12	(3∼13)	(3∼13, 9∼14)	$(-1.01 \times 10^{-15} \sim -8.72 \times 10^{-16}, -8.60 \times 10^{-17} \sim -7.45 \times 10^{-17})$	(1∼6, 1∼5)
I_1	AC21, BC22, AB11	(2∼14)	(2∼14, 5∼15)	$(3.44 \times 10^{-1} \sim 3.61 \times 10^{-1}, 1.25 \times 10^{-17} \sim 2.25 \times 10^{-17})$	(14∼16, 8∼10)
I_2	AB02, AC12, BC11	(1∼11)	(1∼11, 1∼6)	$(1.25 \times 10^{-16} \sim 1.43 \times 10^{-16}, 3.91 \times 10^{-1} \sim 4.31 \times 10^{-1})$	(10∼12, 15∼20)
I_3	AB12, AB22, AC22, BC21, BC21, BB12, BB21	(3∼16)	(3∼16, 4∼19)	$(-1.01 \times 10^{-15} \sim -8.72 \times 10^{-16}, -8.60 \times 10^{-17} \sim -7.45 \times 10^{-17})$	(1∼6, 1∼6)
I_4	AB11, AC11, AC21, BC22, BC12, CC12, CC21	(2∼20)	(2∼20, 2∼20)	$(3.44 \times 10^{-1} \sim 3.95 \times 10^{-1}, 1.25 \times 10^{-17} \sim 3.14 \times 10^{-17})$	(14∼20, 8∼14)
I_5	AB02, AC12, AC12, BC10, BC11, CC01, CC02	(1∼18)	(1∼18, 1∼18)	$(7.69 \times 10^{-17} \sim 1.43 \times 10^{-16}, 3.91 \times 10^{-1} \sim 4.31 \times 10^{-1})$	(7∼12, 15∼20)
I_6	AB22, AC22, BC21	(5∼13)	(5∼13, 11∼17)	$(-1.01 \times 10^{-15} \sim -8.72 \times 10^{-16}, -8.60 \times 10^{-17} \sim -7.45 \times 10^{-17})$	(1∼6, 1∼6)
I_7	AB21, AC21, BC22	(4∼14)	(4∼14, 10∼15)	$(8.54 \times 10^{-2} \sim 3.44 \times 10^{-1}, -1.01 \times 10^{-17} \sim 1.25 \times 10^{-17})$	(13∼16, 7∼9)

Table 2
Vector assignment of spatial strings in the example image database

Spatial string	Occurrence vector								Assigned vector			
	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	Interval filter	Two-dimensional interval filter	Two-dimensional interval filter with clustering	Two dimensional interval filter with clustering after ordering
AB02	0	0	1	0	0	1	0	0	1	(1, 1)	$(1.43 \times 10^{-16}, 3.91 \times 10^{-1})$	(11, 15)
AB11	0	1	0	0	1	0	0	0	2	(2, 5)	$(3.61 \times 10^{-1}, 2.25 \times 10^{-17})$	(14, 10)
AB12	1	0	0	1	0	0	0	0	3	(3, 9)	$(-8.76 \times 10^{-16}, -7.45 \times 10^{-17})$	(4, 5)
AB21	0	0	0	0	0	0	0	1	4	(4, 13)	$(8.45 \times 10^{-2}, -1.01 \times 10^{-17})$	(13, 7)
AB22	0	0	0	1	0	0	1	0	5	(5, 17)	$(-8.76 \times 10^{-16}, -7.45 \times 10^{-17})$	(5, 6)
AC11	0	0	0	0	1	0	0	0	6	(6, 2)	$(3.95 \times 10^{-1}, 3.14 \times 10^{-17})$	(17, 11)
AC12	0	0	1	0	0	2	0	0	7	(7, 6)	$(1.25 \times 10^{-16}, 4.31 \times 10^{-1})$	(10, 20)
AC21	0	1	0	0	1	0	0	1	8	(8, 10)	$(3.44 \times 10^{-1}, 1.25 \times 10^{-17})$	(15, 8)
AC22	1	0	0	1	0	0	1	0	9	(9, 14)	$(-8.72 \times 10^{-16}, -7.59 \times 10^{-17})$	(6, 4)
BC10	0	0	0	0	0	1	0	0	10	(10, 18)	$(7.69 \times 10^{-17}, 4.12 \times 10^{-1})$	(7, 17)
BC11	0	0	1	0	0	1	0	0	11	(11, 3)	$(1.43 \times 10^{-16}, 3.91 \times 10^{-1})$	(12, 16)
BC12	0	0	0	0	1	0	0	0	12	(12, 7)	$(3.95 \times 10^{-1}, 3.14 \times 10^{-17})$	(18, 12)
BC21	1	0	0	2	0	0	1	0	13	(13, 11)	$(-1.01 \times 10^{-15}, -8.60 \times 10^{-17})$	(1, 1)
BC22	0	1	0	0	1	0	0	1	14	(14, 15)	$(3.44 \times 10^{-1}, 1.25 \times 10^{-17})$	(16, 9)
BB12	0	0	0	1	0	0	0	0	15	(15, 19)	$(-9.68 \times 10^{-16}, -7.93 \times 10^{-17})$	(2, 2)
BB21	0	0	0	1	0	0	0	0	16	(16, 4)	$(-9.68 \times 10^{-16}, -7.93 \times 10^{-17})$	(3, 3)
CC01	0	0	0	0	0	1	0	0	17	(17, 8)	$(7.69 \times 10^{-17}, 4.12 \times 10^{-1})$	(8, 18)
CC02	0	0	0	0	0	1	0	0	18	(18, 12)	$(7.69 \times 10^{-17}, 4.12 \times 10^{-1})$	(9, 19)
CC12	0	0	0	0	1	0	0	0	19	(19, 16)	$(3.95 \times 10^{-1}, 3.14 \times 10^{-17})$	(19, 13)
CC21	0	0	0	0	1	0	0	0	20	(20, 20)	$(3.95 \times 10^{-1}, 3.14 \times 10^{-17})$	(20, 14)

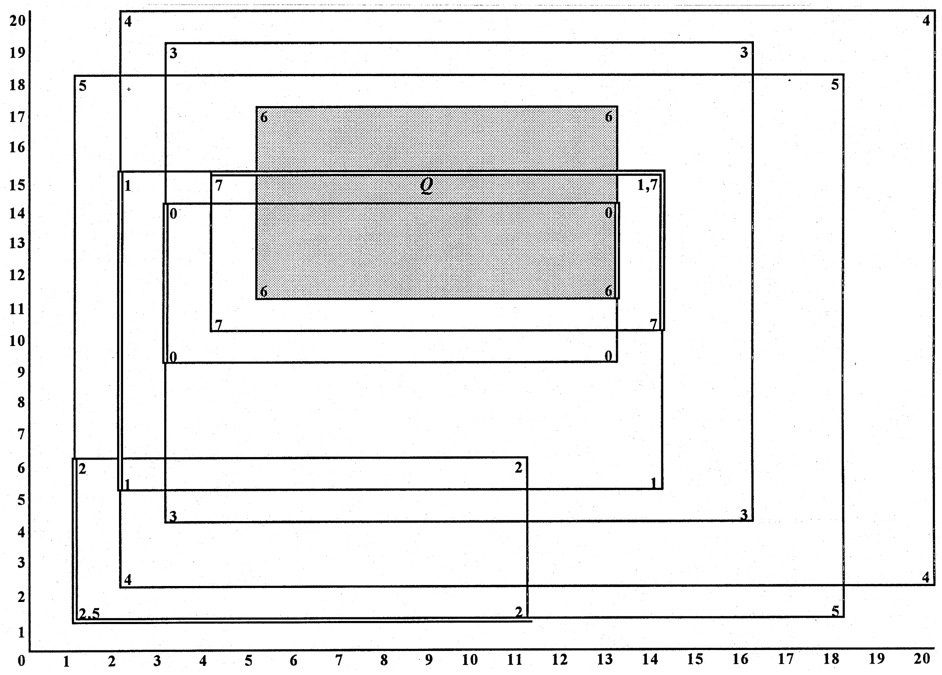

Fig. 3. The two-dimensional rectangles of the eight images in Example 4.

Example 3. For the same example image database in Example 2, each spatial string is assigned a unique 2-dimensional vector as shown in the fourth column of Table 2. Therefore, the interval corresponding to each image is shown in the fourth column of Table 1. Fig. 3 plots the corresponding two-dimensional rectangles. The numbers in the four corners of each rectangle denote the corresponding image numbers. The query rectangle is the shadowed block. Given the query image Q represented as 2D String $(C < B < A,\ B < C < A)$, then $R(Q) = (5{\sim}13,\ 11{\sim}17)$, which is enclosed by $R(I_3)$, $R(I_4)$, $R(I_5)$, $R(I_6)$. I_0, I_1, I_2 and I_7 are filtered out. I_3 and I_6 are actually qualified, I_4 and I_5 are false drops. For the same query image, two false drops are generated by *multidimensional interval filter* while four false drops are generated by *interval filter*.

In Example 3, it can be found that most of the rectangles overlap each other. Certainly, two rectangles overlap if the corresponding images have some common spatial strings. For example, in Example 3, $R(I_1)$ and $R(I_7)$ overlap because I_1 and I_7 have the common spatial strings '*BC*22' and '*AC*21'. However, it is possible that two overlapping rectangles have no common spatial strings. For example, $R(I_2)$ and $R(I_4)$ overlap but there is no common spatial string between I_2 and I_4. This phenomenon comes from the assignment of multidimensional vector for the spatial strings. If the query rectangle falls into these overlapping region, it is expected to incur false drops.

The third strategy, *multidimensional interval filter with clustering* improves the filtering effect by efficient assignment of multidimensional vector. The goal of efficient assignment tries to minimize the areas of multidimensional rectangles of the images in the database, because larger rectangles are more likely to cover the query rectangle and produce more false drops. This goal can be achieved heuristically by clustering the coordinates of spatial strings in multidimensional space. The criterion of clustering depends on the correlation among the spatial strings in the image database. Those spatial strings which tend to appear concurrently are clustered. On the contrary, those spatial strings which seldom appear concurrently are dispersed. This will reduce the false drop probability. We first give some definitions concerning the correlation between spatial strings.

Definition 6. The *occurrence vector* of a spatial string T_{AB} is defined as a vector of images. The ith element of the vector represents the occurrences of T_{AB} in the spatial string collection of image i.

Definition 7. Consider two vectors v_i and v_j which are the occurrence vectors of two spatial strings, the *correlation* c_{ij} between these two spatial strings is measured as $v_i \circ v_j / |v_i| \times |v_j|$, where '$\circ$' is the inner product of two vectors and $|v_i|$ stands for the Euclidean norm of the vector.

Therefore, the problem to be solved is stated as follows. Given that in the image database, totally there are b objects, m images and n spatial strings. The goal is to find the vectors of the n spatial strings in multidimensional space such that the Euclidean distances between vectors of spatial strings satisfy the following *distance criterion*. For any three spatial strings with occurrence vectors v_i, v_j, v_k, if $c_{ij} > c_{ik}$, then $d_{ij} < d_{ik}$, where c_{ij}, d_{ij} is the correlation, distance between spatial strings corresponding to v_i, v_j, respectively.

To solve this problem for multidimensional case, the multidimensional vector of the spatial string with occurrence vector v_i is assigned the *normalized occurrence vector*, $v_i/|v_i|$. That is, the assigned multidimensional vectors of the spatial strings are unit vectors. Given three unit vectors u_r, u_s, u_t, if $u_r \circ u_s > u_r \circ u_t$, we have $c_{rs} > c_{rt}$. Since, u_r, u_s, u_t are normalized, so $c_{rs} \leqslant 1$ and $c_{rt} \leqslant 1$. Therefore, $1 - c_{rs} < 1 - c_{rt}$ which implies $d_{rs} < d_{rt}$. It is obvious that the assignment satisfies the above criterion.

The assignment stated above satisfies the distance criterion. However, usually the number of images, m, is large. It is impractical to assign each spatial string an m-dimensional vector. The reason comes not only from the storage space consideration but also from the efficiency consideration of existing spatial indexing structure. Most of the developed spatial indexing structures, such as R-Trees, work well when the number of dimensions is less than 20 (Faloutsos, 1994). It is necessary to find the solution for a specific number of dimension, say k. Of course, it is possible that the solution doesn't exist. In this situation, the goal is not so strict but to retain the distance criterion as well as possible.

We use the singular value decomposition (SVD) (Leon, 1990) to achieve this goal. Before giving an introduction to SVD, we first define a spatial information matrix.

Definition 8. Given an image database with m images and b objects, the *spatial information matrix* X_0 for this image database is defined as a matrix of n spatial strings by m images. Each row represents the m-dimensional normalized occurrence vector of the corresponding spatial string. That is, for each entry of X_0, x_{ij} equals $y_{ij}/\sqrt{\Sigma_{j=1}^{m} y_{ij}^2}$, where y_{ij} is the number of occurences of ith spatial string in the jth image.

The spatial information matrix represents the spatial information in an image database. The dot product between two rows represents the correlation between corresponding spatial strings. The spatial information matrix is taken as the input to the process of SVD. We then give a brief description of SVD.

Given an $n \times m$ matrix X_0 with rank f, SVD involves factoring X_0 into a product $U_0 S_0 V_0^{\mathrm{T}}$, where U_0 is an $n \times f$ orthonormal matrix, V_0 is an $f \times m$ orthonormal matrix, and S_0 is an $f \times f$ matrix whose off diagonal entries are all 0's and whose diagonal elements $\sigma_1, \sigma_2, \ldots \sigma_f$ satisfy $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_f \geqslant 0$.

The orthonormal columns of U_0 and V_0 are called *left* and *right singular vectors*, respectively. The diagonal elements of S_0 is called singular values. The beauty of SVD lies in the decreasing order of singular values which is useful for optimal approximation. If the first k largest singular values and corresponding left and right orthonormal vectors are kept, then the original matrix X_0 is approximated by $X = USV^{\mathrm{T}}$, where U is an $n \times k$ orthonormal matrix, V is a $k \times m$ orthonormal matrix, and S is a $k \times k$ diagonal matrix (Leon, 1990).

The dot product between two rows of X_0 reflects the correlation between two spatial strings. Therefore, the symmetric matrix $X_0X_0^{\mathrm{T}}$ represents the correlation between each pair of spatial strings, where

$$X_0X_0^{\mathrm{T}} \approx XX^{\mathrm{T}} = (USV^{\mathrm{T}})(USV^{\mathrm{T}})^{\mathrm{T}} = (USV^{\mathrm{T}})(VS^{\mathrm{T}}U^{\mathrm{T}}).$$

Because V is an orthonormal matrix, S is a diagonal matrix

$$XX^{\mathrm{T}} = (USV^{\mathrm{T}})(VS^{\mathrm{T}}U^{\mathrm{T}})^{\mathrm{T}} = USS^{\mathrm{T}}U^{\mathrm{T}} = US(US)^{\mathrm{T}}.$$

Thus, we can assign the row of US as the k-dimensional vector for the spatial string. Besides, instead of taking US, we can take U as the k-dimensional vectors. This is because S is diagonal; the positions of the points are the same except that each of the axes has been stretched or shrunk in proportion to the corresponding diagonal elements of S.

Example 4. The fifth column of Table 2 shows the two-dimensional vector of each spatial strings by using the SVD for the example image database in Example 2. The interval corresponding to each image is shown in the fifth column of Table 1. Given the query image Q represented as 2D String $(C < B < A,\ B < C < A)$, then $R(Q) = (-1.01 \times 10^{-15} \sim -8.72 \times 10^{-16}, -8.60 \times 10^{-17} \sim -7.45 \times 10^{-17})$, which is enclosed by $R(I_0)$, $R(I_3)$ and $R(I_6)$. I_1, I_2, I_4, I_5 and I_7 are filtered out. I_3 and I_6 are actually qualified while I_0 is a false drop. For the same query image, one false drop is incurred while two false drops are generated by *multidimensional interval filter*.

Observe that in Example 4 the elements of assigned vectors are all real numbers. It takes the storage space and the query processing time. Therefore, for each dimension of the assigned vectors, we impose a partial ordering on the corresponding coordinate of the assigned vectors and take the order numbers as the coordinate of assigned vectors. Two spatial strings with the same singular value are assigned with distinct order numbers.

Example 5. The sixth column of Table 2 lists the two-dimensional vectors of the spatial strings after ordering. The interval corresponding to each image is shown in the sixth column of Table 1. Given the query image Q represented as 2D String $(C < B < A,\ B < C < A)$, then $R(Q) = (1 \sim 6,\ 1 \sim 6)$, which is enclosed by $R(I_3)$ and $R(I_6)$. I_0, I_1, I_2, I_4, I_5 and I_7 are filtered out. I_3 and I_6 are actually qualified. No false drops are incurred. Fig. 4 plots the two-dimensional rectangles of the eight images in Example 5. Comparing Fig. 4 with Fig. 3, it is observed that less overlapping rectangles exist in the former. In Fig. 4, two rectangles overlap only if the corresponding images have some common spatial strings. It is expected that the false drop probability is much lower.

4. Performance analysis

We measure the performance of the proposed indexing method by the false drop probability. The false drop probability F_{d} denotes the probability that an image rectangle qualifies, given that the image does not actually qualify. High false drop probability incurs unnecessary 2D subsequence matching. Therefore false drop probability of an effective filter should be as small as possible.

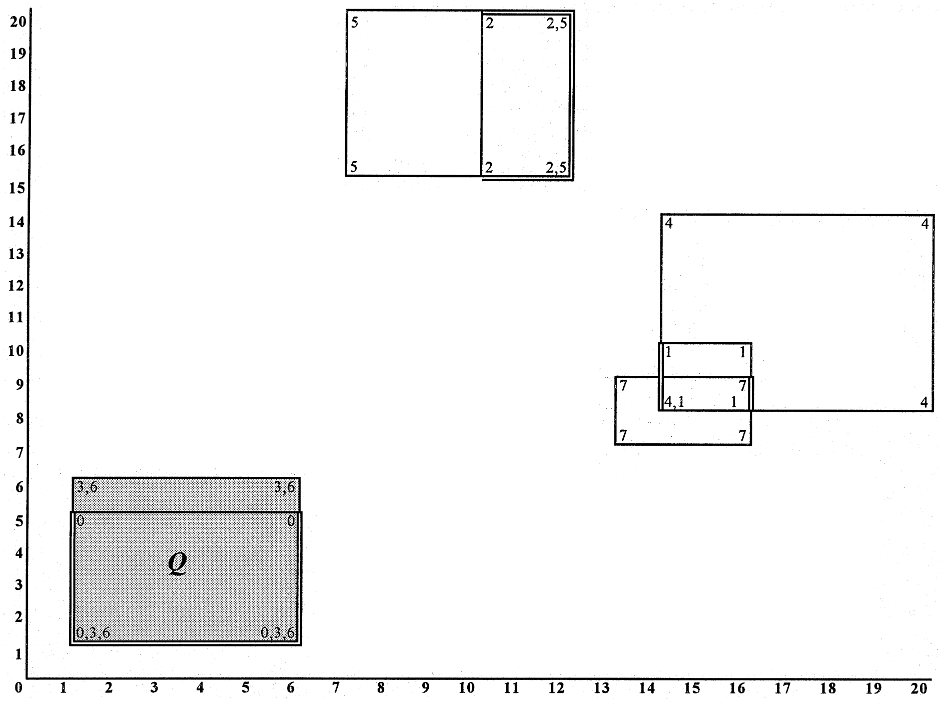

Fig. 4. The two-dimensional rectangles of the eight images in Example 5.

Table 3
Design parameters of simulation

Symbol	Definition
m	Total number of images in the image database
b	Total number of objects in the image database
h	The number of objects in each image
k	The number of dimensions
g	The number of objects in each query image

We implement our *multidimensional filter with clustering* algorithm and carry out experiments in order to demonstrate the effectiveness of our proposed indexing method. The design parameters which affect the false drop probability are listed in Table 3. The software package SVDPACKC developed in University of Tennessee (Berry et al., 1993) was used for computing the SVD of large sparse matrices using ANSI C.

Given m, b and h, database images are generated randomly. In other words, h objects are randomly selected from the b objects for each image and randomly placed in a symbolic picture of 10 by 10 cells. The query images are generated from the database images. For a set of m database images, m query images are generated. Each query image is generated by randomly selecting g objects from a database image. This way of generating query images guarantees that at least one image is qualified for the query image.

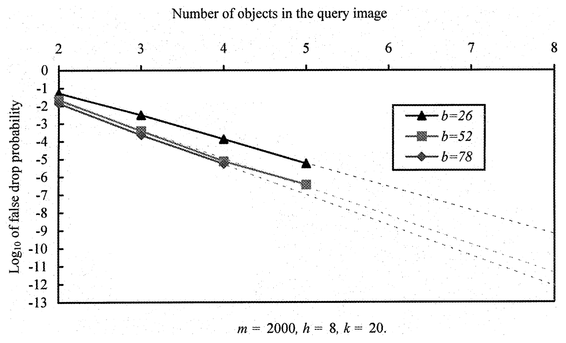

Fig. 5. Average false drop probability versus the number of objects in the query image for total number of objects in the image database.

Figs. 5–8 demonstrate the average false drop probability as the function of the design parameters. Note that in these figures, the value of y-axis denotes the base 10 logarithm of false drop probability. Besides, there exists the case with no false drops in our simulation, and hence the false drop probability is zero. It is meaningless for logarithm of zero, therefore we approximate the curve by regression analysis.

Fig. 5 shows the average false drop probability as a function of the number of objects in the query image for different number of objects in the image database (b equals 26, 52 and 78). From the analysis, it can be seen that the average false drop probability decreases with increasing number of query objects. Increasing the number of query objects would produce larger size of query rectangle and reduce the false drop probability. Besides, given a specific number of query objects, the false drop probability decreases with increasing number of objects in the image database.

Fig. 6 gives the average false drop probability as a function of the number of objects in the database image for total number of objects in the image database b being equal to 26, 52, 78, respectively. It is obvious that the average false drop probability increases with increasing number of objects in the database image. Increasing number of objects in the database image would produce larger database image rectangle, and hence increase the false drop probability. Fig. 7 demonstrates the effect of the number of dimensions for the number of query objects g being equal to 2, 3, 4, 5, 6, 7, 8, 9, 10, respectively. Of course, the average false drop probability decreases with increasing number of dimensions.

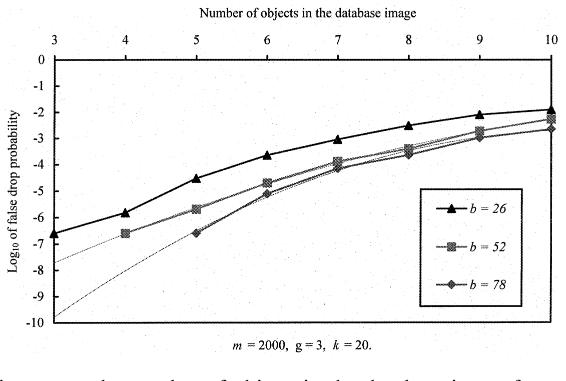

Fig. 6. Average false drop probability versus the number of objects in the database image for total number of objects in the image database.

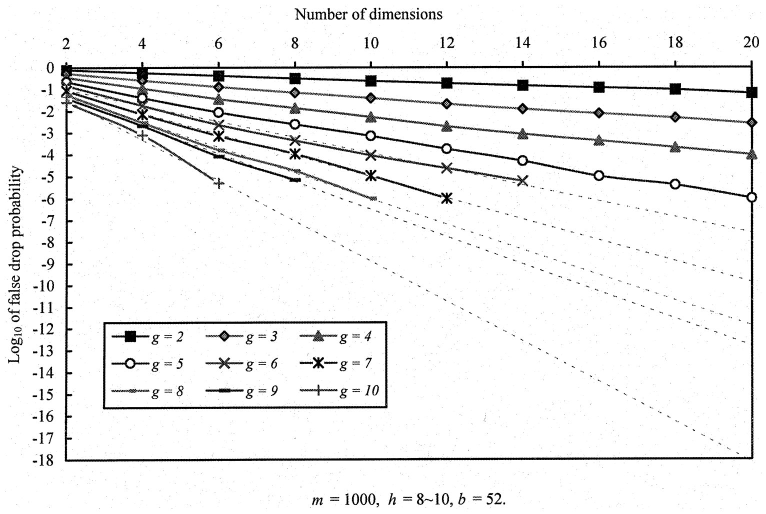

Fig. 7. Average false drop probability versus the number of dimensions for the number of query objects.

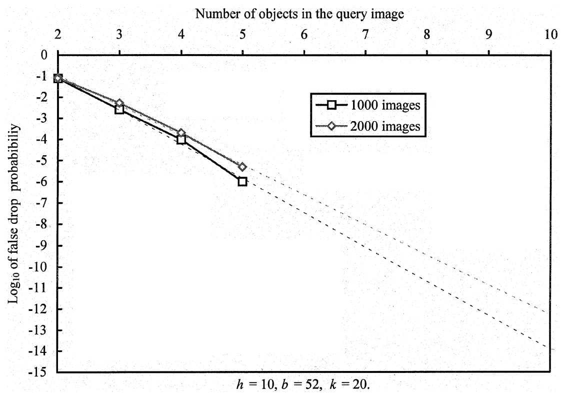

Fig. 8. Average false drop probability versus the number of objects in the query image for the number of images in the image database.

Fig. 8 compares the average false drop probability between the image database with 1000 images and that with 2000 images. The number of dimension is set to 20. Obviously, the false drop probability of the case with 2000 images is higher than that with 1000 images. The relationship among the number of images, the number of dimensions and average false drop probability are critical to the performance of our system. In practice, the number of dimensions must be chosen to achieve a good retrieval performance.

5. Conclusions

In this paper, we propose a new indexing method for subpicture query of iconic image databases. The basic mechanism of this indexing method is a multidimensional filter which is able to prune out most of the unqualified images. The spatial information of each image is transformed into a multidimensional rectangle. Those images whose corresponding rectangles do not contain the query rectangle are filtered out. We also present the method using SVD to improve the effectiveness of multidimensional filter.

The advantages of the proposed indexing method lies in the effectiveness and efficiency of filter process. For the efficiency, the proposed indexing method can adopt a spatial searching structure such as R-Trees to speed up rectangle containment test in a query process. The results of simulations have shown that the average false drop probability depends on the number of objects in the image database, the number of objects in each image, the number of objects in the query image, the number of dimensions and the number of images in the image database. In a typical case with eight objects in each image, 2000 images in the image database, three query objects in the query image, the average false drop probability for 20 dimensions is only 0.003, which is very effective.

Future research includes the derivation of the formula of false drop probability as a function of the design parameters and the performance analysis for the other variants of 2D string representation. The other variants capture more complex spatial relationships. It is worth to investigate the effect of more complex spatial relationships on false drop probability.

Acknowledgements

We thank Prof. P.W. Huang for his valuable suggestion and corrections.

Appendix A

This appendix presents the detailed numerical results for the multidimensional filter with clustering using the example image database in Example 4. The spatial information matrix X_0 is presented below. Note that the corresponding rows for those spatial strings that are absent in the image database are not shown in this matrix. It does not matter, because the assigned multidimensional vectors (left singular vectors) for these absent spatial strings, generated from the SVD of the spatial information matrix, are all zero vectors.

$$X_0 = \begin{array}{c|ccccccccc|}
 & I_0 & I_1 & I_2 & I_3 & I_4 & I_5 & I_6 & I_7 & \\
AB02 & 0 & 0 & 1/\sqrt{2} & 0 & 0 & 1/\sqrt{2} & 0 & 0 & \\
AB11 & 0 & 1/\sqrt{2} & 0 & 0 & 1/\sqrt{2} & 0 & 0 & 0 & \\
AB12 & 1/\sqrt{2} & 0 & 0 & 1/\sqrt{2} & 0 & 0 & 0 & 0 & \\
AB21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & \\
AB22 & 0 & 0 & 0 & 1/\sqrt{2} & 0 & 0 & 1/\sqrt{2} & 0 & \\
AC11 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \\
AC12 & 0 & 0 & 1/\sqrt{5} & 0 & 0 & 2/\sqrt{5} & 0 & 0 & \\
AC21 & 0 & 1/\sqrt{3} & 0 & 0 & 1/\sqrt{3} & 0 & 0 & 1/\sqrt{3} & \\
AC22 & 1/\sqrt{3} & 0 & 0 & 1/\sqrt{3} & 0 & 0 & 1/\sqrt{3} & 0 & \\
BB12 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
BB21 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \\
BC10 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \\
BC11 & 0 & 0 & 1/\sqrt{2} & 0 & 0 & 1/\sqrt{2} & 0 & 0 & \\
BC12 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \\
BC21 & 1/\sqrt{6} & 0 & 0 & 2/\sqrt{6} & 0 & 0 & 1/\sqrt{6} & 0 & \\
BC22 & 0 & 1/\sqrt{3} & 0 & 0 & 1/\sqrt{3} & 0 & 0 & 1/\sqrt{3} & \\
CC01 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \\
CC02 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \\
CC12 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \\
CC21 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 &
\end{array}.$$

Recall that SVD involves factoring X_0 into a product $U_0S_0V_0^{\mathrm{T}}$. These matrices U_0, S_0 and V_0 are shown in the next page. We can verify that $X_0 = U_0S_0V_0^{\mathrm{T}}$ except for small rounding errors. To generate the two-dimensional vectors of the spatial strings, only the first two columns of the left singular matrix are kept. In other words, we approximate X_0 by keeping only the first two singular values and the corresponding columns from the left singular and right singular matrices U and V.

$$\begin{array}{l|cccccccc}
AB02 & 1.43\times10^{-16} & 3.91\times10^{-1} & -1.18\times10^{-17} & -2.98\times10^{-17} & 5.18\times10^{-1} & 4.35\times10^{-15} & -1.43\times10^{-20} & -1.89\times10^{-15} \\
AB11 & 3.61\times10^{-1} & 2.25\times10^{-17} & -9.32\times10^{-16} & -4.93\times10^{-2} & -1.09\times10^{-16} & 6.58\times10^{-17} & -6.5\times10^{-1} & 9.88\times10^{-16} \\
AB12 & -8.76\times10^{-16} & -7.45\times10^{-17} & -3.9\times10^{-1} & -1.03\times10^{-16} & -1.26\times10^{-15} & 1.62\times10^{-1} & 7.21\times10^{-22} & -7.07\times10^{-1} \\
AB21 & 8.54\times10^{-2} & -1.01\times10^{-17} & -3.99\times10^{-16} & -6.58\times10^{-1} & -1.52\times10^{-16} & 1.65\times10^{-17} & 5.81\times10^{-1} & -8.13\times10^{-15} \\
AB22 & -8.76\times10^{-16} & -7.45\times10^{-17} & -3.9\times10^{-1} & -1.03\times10^{-16} & -1.26\times10^{-15} & 1.62\times10^{-1} & 7.21\times10^{-22} & -7.07\times10^{-1} \\
AC11 & 3.95\times10^{-1} & 3.14\times10^{-17} & -9.42\times10^{-16} & 2.30\times10^{-1} & -6.39\times10^{-17} & 7.16\times10^{-17} & 2.02\times10^{-1} & -1.65\times10^{-15} \\
AC12 & 1.25\times10^{-16} & 4.31\times10^{-1} & -4.51\times10^{-17} & -7.43\times10^{-18} & 1.56\times10^{-1} & 1.36\times10^{-15} & 1.57\times10^{-20} & -4.53\times10^{-16} \\
AC21 & 3.44\times10^{-1} & 1.25\times10^{-17} & -9.91\times10^{-16} & -4.2\times10^{-1} & -1.77\times10^{-16} & 6.32\times10^{-17} & -1.96\times10^{-1} & -3.89\times10^{-15} \\
AC22 & -8.72\times10^{-16} & -7.59\times10^{-17} & -4.01\times10^{-1} & -3.62\times10^{-17} & -4.59\times10^{-15} & 5.75\times10^{-1} & 7.41\times10^{-22} & -8.48\times10^{-15} \\
BB12 & -9.68\times10^{-16} & -7.93\times10^{-17} & -4.08\times10^{-1} & -2.29\times10^{-16} & 4.41\times10^{-15} & -5.4\times10^{-1} & 7.55\times10^{-22} & 3.51\times10^{-15} \\
BB21 & -9.68\times10^{-16} & -7.93\times10^{-17} & -4.08\times10^{-1} & -2.29\times10^{-16} & 4.41\times10^{-15} & -5.4\times10^{-1} & 7.55\times10^{-22} & 3.51\times10^{-15} \\
BC10 & 7.69\times10^{-17} & 4.12\times10^{-1} & -8.41\times10^{-17} & 2.56\times10^{-17} & 3.83\times10^{-1} & -3.1\times10^{-15} & 1.5\times10^{-20} & 1.67\times10^{-15} \\
BC11 & 1.43\times10^{-16} & 3.91\times10^{-1} & -1.18\times10^{-17} & -2.98\times10^{-17} & 5.18\times10^{-1} & 4.35\times10^{-15} & 1.43\times10^{-20} & -1.89\times10^{-15} \\
BC12 & 3.95\times10^{-1} & 3.14\times10^{-17} & -9.42\times10^{-16} & 2.30\times10^{-1} & -6.39\times10^{-17} & 7.16\times10^{-17} & 2.02\times10^{-1} & -1.65\times10^{-15} \\
BC21 & -1.01\times10^{-15} & -8.6\times10^{-17} & -4.5\times10^{-1} & -1.19\times10^{-16} & -1.45\times10^{-15} & 1.86\times10^{-1} & 8.32\times10^{-22} & -4.63\times10^{-15} \\
BC22 & 3.44\times10^{-1} & 1.25\times10^{-17} & -9.91\times10^{-16} & -4.2\times10^{-1} & -1.77\times10^{-16} & 6.32\times10^{-17} & -1.96\times10^{-1} & -3.89\times10^{-15} \\
CC01 & 7.69\times10^{-17} & 4.12\times10^{-1} & -8.41\times10^{-17} & 2.56\times10^{-17} & 3.83\times10^{-1} & -3.1\times10^{-15} & 1.5\times10^{-20} & 1.67\times10^{-15} \\
CC02 & 7.69\times10^{-17} & 4.12\times10^{-1} & -8.41\times10^{-17} & 2.56\times10^{-17} & 3.83\times10^{-1} & -3.1\times10^{-15} & 1.5\times10^{-20} & 1.67\times10^{-15} \\
CC12 & 3.95\times10^{-1} & 3.14\times10^{-17} & -9.42\times10^{-16} & 2.30\times10^{-1} & -6.39\times10^{-17} & 7.16\times10^{-17} & 2.02\times10^{-1} & -1.65\times10^{-15} \\
CC21 & 3.95\times10^{-1} & 3.14\times10^{-17} & -9.42\times10^{-16} & 2.30\times10^{-1} & -6.39\times10^{-17} & 7.16\times10^{-17} & 2.02\times10^{-1} & -1.65\times10^{-15}
\end{array},$$

$$S_0 = \begin{bmatrix}
2.38 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2.30 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 2.19 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1.32 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.85 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.82 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.78 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.71
\end{bmatrix},$$

$$V_0^{\mathrm{T}} = \begin{bmatrix}
-6.45\times10^{-16} & 2.74\times10^{-1} & 2.99\times10^{-16} & -2.3\times10^{-15} & 9.4\times10^{-1} & 1.83\times10^{-16} & -6.45\times10^{-16} & 2.03\times10^{-1} \\
-6.01\times10^{-17} & 9.61\times10^{-19} & 3.24\times10^{-1} & -1.82\times10^{-16} & 7.2\times10^{-17} & 9.46\times10^{-1} & -6.01\times10^{-17} & -2.32\times10^{-17} \\
-3.15\times10^{-1} & -8.25\times10^{-16} & 1.48\times10^{-16} & -8.96\times10^{-1} & -2.07\times10^{-15} & -1.85\times10^{-16} & -3.15\times10^{-1} & -8.75\times10^{-16} \\
1.09\times10^{-16} & -3.94\times10^{-1} & -8.94\times10^{-17} & -3.01\times10^{-16} & 3.03\times10^{-1} & 3.37\times10^{-17} & 1.09\times10^{-16} & -8.68\times10^{-1} \\
-5.24\times10^{-15} & -7.66\times10^{-17} & 9.46\times10^{-1} & 3.74\times10^{-15} & -5.42\times10^{-17} & -3.24\times10^{-1} & -5.24\times10^{-15} & -1.29\times10^{-16} \\
6.33\times10^{-1} & 1.77\times10^{-17} & 7.62\times10^{-15} & -4.45\times10^{-1} & 5.9\times10^{-17} & -2.55\times10^{-15} & 6.33\times10^{-1} & 1.36\times10^{-17} \\
2.07\times10^{-22} & -8.77\times10^{-1} & 4.03\times10^{-21} & 5.9\times10^{-22} & 1.58\times10^{-1} & 1.17\times10^{-20} & 2.07\times10^{-22} & 4.54\times10^{-1} \\
-7.07\times10^{-1} & 2.16\times10^{-15} & -3.07\times10^{-15} & 2.48\times10^{-15} & -1.17\times10^{-15} & 1.18\times10^{-15} & 7.07\times10^{-1} & -5.75\times10^{-15}
\end{bmatrix},$$

$$U = \begin{array}{c} AB02 \\ AB11 \\ AB12 \\ AB21 \\ AB22 \\ AC11 \\ AC12 \\ AC21 \\ AC22 \\ BB12 \\ BB21 \\ BC10 \\ BC11 \\ BC12 \\ BC21 \\ BC22 \\ CC01 \\ CC02 \\ CC12 \\ CC21 \end{array} \left[\begin{array}{cc} 1.43 \times 10^{-16} & 3.91 \times 10^{-1} \\ 3.61 \times 10^{-1} & 2.25 \times 10^{-17} \\ -8.76 \times 10^{-16} & -7.75 \times 10^{-17} \\ 8.54 \times 10^{-2} & -1.01 \times 10^{-17} \\ -8.76 \times 10^{-16} & -7.75 \times 10^{-17} \\ 3.95 \times 10^{-1} & 3.14 \times 10^{-17} \\ 1.25 \times 10^{-16} & 4.31 \times 10^{-1} \\ 3.44 \times 10^{-1} & 1.25 \times 10^{-17} \\ -8.72 \times 10^{-16} & -7.59 \times 10^{-17} \\ -9.68 \times 10^{-16} & -7.93 \times 10^{-17} \\ -9.68 \times 10^{-16} & -7.93 \times 10^{-17} \\ 7.69 \times 10^{-17} & 4.12 \times 10^{-1} \\ 1.43 \times 10^{-16} & 3.91 \times 10^{-1} \\ 3.95 \times 10^{-1} & 3.14 \times 10^{-17} \\ -1.01 \times 10^{-15} & -8.6 \times 10^{-17} \\ 3.44 \times 10^{-1} & 1.25 \times 10^{-17} \\ 7.69 \times 10^{-17} & 4.12 \times 10^{-1} \\ 7.69 \times 10^{-17} & 4.12 \times 10^{-1} \\ 3.95 \times 10^{-1} & 3.14 \times 10^{-17} \\ 3.95 \times 10^{-1} & 3.14 \times 10^{-17} \end{array}\right],$$

$$S = \begin{bmatrix} 2.38 & 0 \\ 0 & 2.30 \end{bmatrix},$$

$$D^{\mathrm{T}} = \begin{bmatrix} -6.45 \times 10^{-16} & 2.74 \times 10^{-1} & 2.99 \times 10^{-16} & -2.30 \times 10^{-15} & 9.4 \times 10^{-1} & 1.83 \times 10^{-16} & -6.45 \times 10^{-16} & 2.03 \times 10^{-1} \\ -6.01 \times 10^{-17} & 9.61 \times 10^{-19} & 3.24 \times 10^{-1} & -1.82 \times 10^{-16} & 7.2 \times 10^{-17} & 9.46 \times 10^{-1} & -6.01 \times 10^{-17} & -2.32 \times 10^{-17} \end{bmatrix}.$$

References

Berry M., et al., 1993. SVDPACKC (Version 1.0) user's guide. Technical Report CS-93-194, Department of Computer Science, University of Tennessee.

Chang, C.C., Lin, D.C., 1996. A spatial data representation: an adaptive 2D-H string. Pattern Recognition Letters 17 (2), 175–185.

Chang, S.K., Li, Y., 1988. Representation of multi-resolution symbolic and binary pictures using 2D-H string. In: Proceedings of IEEE Workshop on Languages for Automata, Maryland, pp. 190–195.

Chang, S.K., Shi, Q.Y., Yang, C.W., 1987. Iconic indexing by 2-D strings. IEEE Transactions Pattern Analysis and Machine Intelligence PAMI-9 (3), 413–428.

Chang, S.K., Jungert, E., Li, Y., 1988. Representation and retrieval of symbolic pictures using generalized 2D strings. Technical Report, PA15260, University of Pittsburgh.

Chou, A.Y.H., Chang, C.C., Yang, W.P., 1997. Symbolic indexing by N-strings. International Journal of Pattern Recognition and Artificial Intelligence (submitted).

Gudivada, V.N., Raghavan, V.V., 1995. Content-based image retrieval systems. Guest editor's introduction. IEEE Computer 28 (9), 3–7.

Guttman, A., 1984. R-trees: A dynamic index structure for spatial searching. In: Proceedings of ACM-SIGMOD 1984, International Conference on Management of Data, pp. 47–57.

Huang, P.W., Jean, Y.R., 1994. Using 2D C^{+}-string as spatial knowledge representation for image database systems. Pattern Recognition 27 (9), 1249–1257.

Lee, S.Y., Hsu, F.J., 1991. 2D C-string: A new spatial knowledge representation for image database systems. Pattern Recognition 12 (7), 425–435.

Lee, S.Y., Yang, M.C., Chen, J.W., 1992. 2D B-string spatial knowledge representation and picture retrieval for image database. In: Proceedings of Second International Computer Science Conference, Data and Knowledge Engineering: Theory and Applications.

Leon, S.J., 1990. Linear Algebra with Applications, 3rd edition, Macmillan, New York.

Lin, K.I., Jagadish, H.V., Faloutsos, C., 1994. The TV-tree: An index structure for high-dimensional data. VLDB Journal 3 (4), 517–542.

Tucci, M., Costagliola, G., Chang, S.K., 1991. A remark on NP-completeness of picture matching information. Information Processing Letters 39 (5), 241–243.