國立中山大學資訊工程學系

博士論文

一個以獨特位元模式為基礎來支援圖像資料庫系統中圖像旋轉與翻轉之索引技術

A Unique-Bit-Pattern-Based Indexing Strategy for Image Rotation and Reflection in Image Databases

研究生：葉韋宏　撰

指導教授：張玉盈 教授

中華民國 九十七 年 六 月

中文摘要

一個影像資料庫儲存了大量的影像資料與相關的資訊，這些影像資料與資訊是由真實的影像圖片與相對應的符號圖片所組成。影像資料庫系統的相似程度擷取的應用當中，空間關係是一個相當重要的參考因素。為了能夠從影像資料庫中尋找出有興趣的資料，必須有能力推論組成圖片的物件彼此之間的空間關係。隨著數位相機和影像處理軟體的普及，許多的影像資料可以輕易地旋轉或翻轉。也就是說，這些影像可以被旋轉到特定的角度、水平翻轉或垂直翻轉。一個穩固的影像相似擷取架構，可以辨識出各種影像轉換，例如：變形、放大縮小、旋轉、或任意的轉換組合。目前已發表的空間相似擷取演算法可以歸納成三種類型：符號圖像投射類型、幾何空間類型、與圖學技術比對類型。符號圖像投射可以保留物件中與空間有關的重要資訊，例如：長度、寬度、以及座落位置。然而，許多以符號圖像投射為主的物件索引技術對於圖像的翻轉與旋轉相當敏感。因此，以空間關係搜尋圖像時，如果指定的空間關係方位和資料庫中儲存不一致，搜尋的結果會遺漏符合條件的影像。為了解決這個問題，學者們整理出影像轉換後，空間關係變化的規則。並提供一組條件式，將空間關係對應到轉變後的結果。然而，這組條件式由一系列的條件判斷所組成，導致運算沒有效率。在這本博士論文中，首先，我們將上述的條件式分成三種類型。根據這樣的分類，細心地為每一個空間關係指定一個 16 位元長度的位元字串。這樣的對應，可以將空間關係的轉換根據我們提出的位元運算—*intra-exchange*—來完成。此位元運算的時間複雜度為 O(1)的 CPU 運算時間。除此之外，我們設計了一個物件索引技術來儲存物件之間的空間關係。此索引技術稱為 *Unique Bit Pattern Matrix*。處理影像相似擷取時，我們不需要從索引中推導出原來的影像，再透過旋轉或翻轉此影像來獲得相對應的索引。然後根據推導出來的索引做相似度比對。相反地，透過位元運算和矩陣運算，我們的索引技術可以直接推導出影像旋轉或翻轉之後相對應的索引。透過推導出的索引來做相似度比對，可以保證我們設計的索引機制不會遺漏符合使用者條件的影像。在效能評估中，首先分析我們設計的索引機制在進行相似度擷取的時間複雜度。接著，我們呈現透過模擬測試效能的結果。結果顯示，我們的索引機制的效能表現，遠比以條件式為主的索引機制還要優異。依據影像中所包含的物件個數的不同，效能的提升介於 13.64% 和 53.23% 之間。

A UNIQUE-BIT-PATTERN-BASED INDEXING STRATEGY FOR IMAGE ROTATION AND REFLECTION IN IMAGE DATABASES

A Dissertation
Submitted to the Faculty

of

National Sun Yat-sen University

by

Wei-Horng Yeh

In Partial Fulfillment of the
Requirements for the Degree

of

Doctor of Philosophy

June 2008

ABSTRACT

A symbolic image database system is a system in which a large amount of image data and their related information are represented by both symbolic images and physical images. Spatial relationships are important issues for similarity-based retrieval in many image database applications. How to perceive spatial relationships among the components in a symbolic image is an important criterion to find a match between the symbolic image of the scene object and the one being store as a modal in the symbolic image database. With the popularity of digital cameras and the related image processing software, a sequence of images are often rotated or flipped. That is, those images are transformed in the rotation orientation or the reflection direction. A robust spatial similarity framework should be able to recognize image variants such as translation, scaling, rotation, and arbitrary variants. Current retrieval by spatial similarity algorithms can be classified into symbolic projection methods, geometric methods, and graph-matching methods. Symbolic projection could preserve the useful spatial information of objects, such as width, height, and location. However, many iconic indexing strategies based on symbolic projection are sensitive to rotation or reflection. Therefore, these strategies may miss the qualified images, when the query is issued in the orientation different from the orientation of the database images. To solve this problem, researchers derived the rule of the change of spatial relationships in image transformation, and proposed a function to map the spatial relationship to its related transformed one. However, this mapping consists of several conditional statements, which is time-consuming. Thus, in this dissertation, first, we classify the mapping into three cases and carefully assign a 16-bit unique bit pattern to each spatial relationship. Based on the assignment, we can easily do the mapping through our proposed bit operation, *intra-exchange*, which is a CPU operation and needs only the complexity of O(1). Moreover, we propose an efficient iconic index strategy, called *Unique Bit Pattern* matrix strategy (UBP matrix strategy) to record the spatial information. In this way, when doing similarity retrieval, we do not need to reconstruct the original image from the UBP matrix in order to obtain the indexes of the rotated and flipped image. Conversely, we can directly derive the index of the rotated or flipped image from the index of the original one through bit operations and the matrix manipulation. Thus, our proposed strategy can do similarity retrieval without missing the qualified database images. In our performance study, first, we analyze the time complexity of the similarity retrieval process of our proposed strategy. Then, the efficiency of our proposed strategy according to the simulation results is presented. We show that our strategy outperforms those mapping strategies based on different number of objects in an image. According to the different number of objects in an image, the percentage of improvement is between 13.64% and 53.23%.

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

Figure Page

Figure Page

LIST OF TABLES

Table Page

CHAPTER 1

Introduction

Content-based image retrieval (CBIR), a technique which uses visual contents to search images from large scale image databases according to users' interests, has been an active and fast advancing research area since the 1990s. During the past decade, remarkable progress has been made in both theoretical research and system development. However, there remain many challenging research problems that continue to attract researchers from multiple disciplines.

Early work on image retrieval can be traced back to the late 1970s. In 1979, a conference on Database Techniques for Pictorial Applications [3] was held in Florence. Since then, the application potential of image database management techniques has attracted the attention of researchers [11, 12, 18, 21]. Early techniques were not generally based on visual features but on the textual annotation of images. In other words, images were first annotated with text and then searched using text-based approach from traditional database management systems. Comprehensive surveys of early *text-based image retrieval* methods can be found in [13, 72]. Text-based image retrieval uses traditional database techniques to manage images. Through text descriptions, images can be organized by topical or semantic hierarchies to facilitate easy navigation and browsing based on standard Boolean queries. However, since automatically generating descriptive texts for a wide spectrum of images is not feasible, most text-based retrieval systems require manual annotation of images. Obviously, annotating images manually is a cumbersome and expensive task for large image databases, and

is often subjective, context-sensitive and incomplete. As a result, it is difficult for the traditional text-based methods to support a variety of task-dependent queries.

In the early 1990s, as a result of advances in the Internet and new digital image sensor technologies, the volume of digital images produced by scientific, educational, medical, industrial, and other applications available to users increased dramatically. The difficulties faced by text-based retrieval became more and more severe. The efficient management of the rapidly expanding visual information became an urgent problem. This need formed the driving force behind the emergence of content-based image retrieval techniques. In 1992, the National Science Foundation of the United States organized a workshop on visual information management systems [50] to identify new directions in image database management systems. It was widely recognized that a more efficient and intuitive way to represent and index visual information would be based on properties that are inherent in the images themselves. Researchers from the communities of computer vision, database management, human-computer interface, and information retrieval were attracted to this field. Since then, research on content-based image retrieval has developed rapidly [5, 28, 32, 35, 79]. Since 1997, the number of research publications on the techniques of visual information extraction, organization, indexing, user query and interaction, and database management has increased enormously. Similarly, a large number of academic and commercial retrieval systems have been developed by universities, government organizations, companies, and hospitals. In the past decade, a few commercial products and experimental prototype systems have been developed, such as QBIC [32], Photobook [62], Virage [40], VisualSEEK [71], Netra [60], SIMPLIcity [74]. Comprehensive surveys of these techniques and systems can be found in [34, 67, 70, 78].

Content-based image retrieval, uses the visual contents of an image such as *color, shape, texture* and *spatial layout* to represent and index the image. In typical content-based image retrieval systems, as shown in Figure 1.1, the visual contents of the images in the database are extracted and described by multi-dimensional feature vectors. The feature vectors of the images in the database form a feature database. To retrieve images, users provide the retrieval system with example images or sketched

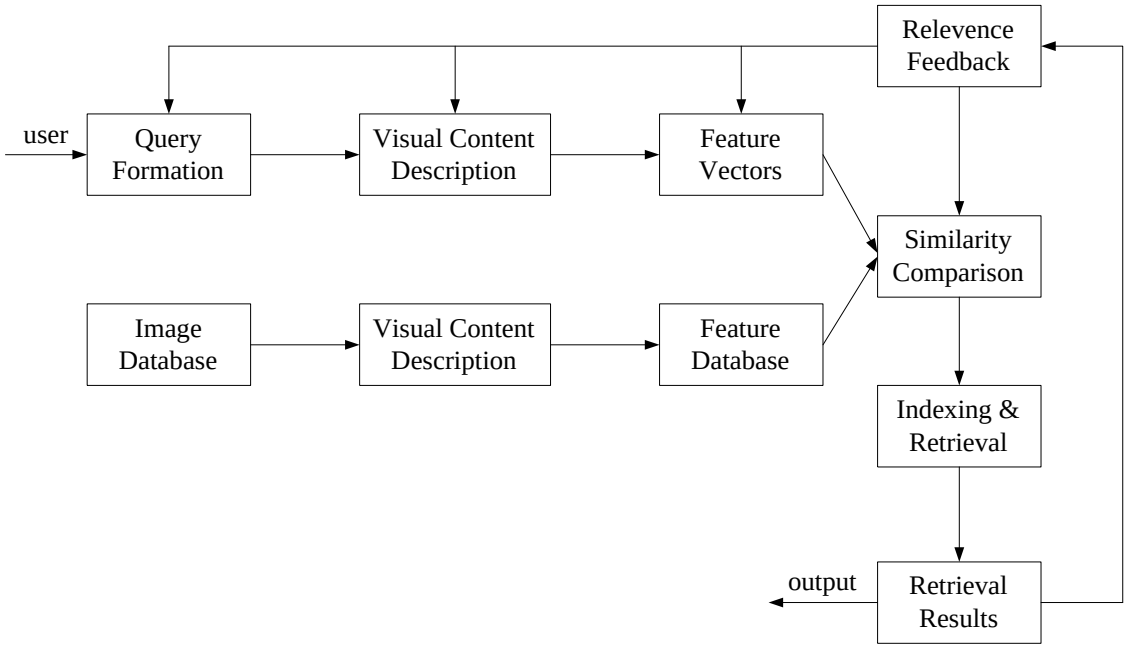


Figure 1.1 Process of the content-based image retrieval system

figures. The system then changes these examples into its internal representation of feature vectors. The similarities or distances between the feature vectors of the query example of sketch and those of the images in the database are then calculated and retrieval is efficient way to search for the image database. Recent retrieval systems have incorporated users' relevance feedback to modify the retrieval process in order to generate perceptually and semantically more meaningful retrieval results.

Eakins *et al.* [29] mentioned three levels of queries in CBIR.

Level 1: Retrieval by primitive features such as color, texture, shape or the spatial location of image elements. Typical query is query by example, "find pictures like this".

Level 2: Retrieval of objects of given type identified by derived features, with some degree of logical inference. For example, "find a picture of a flower".

Level 3: Retrieval by abstract attributes, involving a significant amount of high-level reasoning about the purpose of the objects or scenes depicted. This includes

retrieval of named events, of pictures with emotional or religious significance, etc. Query example, "find pictures of a joyful crowd".

Levels 2 and 3 together are referred to as semantic image retrieval, and the gap between Levels 1 and 2 as the semantic gap. More specifically, the discrepancy between the limited descriptive power of low-level image features and the richness of user semantics, is referred to as the "semantic gap" [27]. Users in Level 1 retrieval are usually required to submit an example image or sketch as query. But what if the user does not have an example image at hand? Semantic image retrieval is more convenient for users as it supports query by keywords or by texture. Therefore, to support query by high-level concepts, a CBIR systems should provide full support in bridging the "semantic gap" between numerical image features and the richness of human semantics [82]. A comprehensive survey of the high-level semantic-based image retrieval can be found in [59].

1.1 Image Content Descriptor

Generally speaking, image content may include both visual and semantic content. Visual content can be very general or domain specific. *General visual content* includes color, texture, shape, spatial relationship, *etc.* *Domain specific visual content*, like human faces, is application dependent and may involve domain knowledge. *Semantic content* is obtained either by textual annotation or by complex inference procedures based on visual content. This dissertation concentrates on general visual contents descriptors, especially spatial relationships.

A good visual content descriptor should be invariant to the accidental variance introduced by the image process. That is the variation of the illuminate of the scene. However, there is a tradeoff between the invariance and the discriminative power of visual features, since a very wide class of invariance loses the ability to discriminate between essential differences. Invariant description has been largely investigated in computer vision (like object recognition), but is relatively new in image retrieval [4].

A visual content descriptor can be either global or local. A global descriptor uses the visual features of the whole image, whereas a local descriptor uses the visual features of *regions* or *objects* to describe the image content. To obtain the local visual descriptors, an image is often divided into parts first. The simplest way of dividing an image is to use a *partition*, which cuts the image into tiles of equal size and shape. A simple partition does not generate perceptually meaningful regions but is a way of representing the global features of the image at a finer resolution. A better method is *region segmentation* algorithms that have been extensively investigated in computer vision. A more complex way of dividing an image is to undertake a complete *object segmentation* to obtain semantically meaningful objects, such as tree, lake, and bird. Currently, automatic object segmentation for broad domains of general image is unlikely to succeed.

1.2 Spatial Relationship

Regions or objects with similar color and texture properties can be easily distinguished by imposing spatial constraints. For instance, regions of blue sky and ocean may have similar color histograms, but their spatial locations in images are different. Therefore, the spatial location of regions (or objects) or the spatial relationship between multiple regions (or objects) in an image is very useful for searching images.

Spatial relationships may be classified into directional and topological relationships. The frequently used directional relationships are the strict directional relationships: north, south, east, and west. Some researchers add the mixed directional relationships: northeast, northwest, southeast, and southwest. Others use the positional directional relationships: left, right, above, and below. On the other hand, some researchers specify the directional relationship between two objects as the slope of the line joining their centroids.

Egenhofer *et al.* [30, 31] pointed out that there are eight fundamental topological relations that can hold between two planar regions. These relations are disjoint, contains, inside, meet, equal, covers, covered-by, and overlap. This model is

called the four-intersection model. A refinement to this model was proposed in [31] to distinguish between topologically distinct configurations whose empty/nonempty-intersection values are the same.

Directional relationships are not sufficient for characterizing spatial similarity because they only consider the spatial orientation of an object while ignoring its spatial extent. In some cases, directional relationships do not exist, while in other cases directional relationships may be identical in two images in spite of the fact that the images are not spatially identical. In addition, directional relations are not rotation invariant.

Topological relationships, on the other hand, always exist between any two objects [56]. Also, topological relationships are mutually exclusive, *i.e.*, there is one and only one topological relationship between any two objects in an image. Another interesting feature of topological relationships is that they are preserved under perfect translation, scaling, or rotation transformations.

1.3 Retrieval by Spatial Similarity

Retrieval by spatial similarity (RSS) deals with a class of queries that is based on spatial relationships among the domain objects. RSS queries may be utilized in applications such as

- Interior design: retrieve the floor layout designs in an archive that are spatially similar to a given floor design.
- Real estate marketing: retrieve the houses which have a swimming pool and a tennis court in the middle of a garden.
- Diagnostic medical imaging: retrieve all brain MRI images, treatment, and outcome for all studies that have a tumor in the same location as the one in the query image.
- Geographic information systems (GIS): retrieve all areas where a gas station and a bank are on the opposite sides of a road.

Spatial relationship is a fuzzy concept and is thus often dependent on human interpretation. A spatial similarity function assesses the degree to which the spatial relationships in a database image conform to those specified in the query image. A spatial similarity algorithm provides a ranked ordering of database images with respect to a query image by applying the spatial similarity function between the query image and each database image [36].

There are three major query types: (1) *spatial reasoning*, (2) *pictorial query*, and (3) *similarity retrieval*. Spatial reasoning means the inference of a consistent set of spatial relationships among the objects in an image. A pictorial query allows the users to query images with a specified spatial relationship. For example, "display all images with a lake east of a mountain." The target of similarity retrieval is to retrieve the images that are similar to the query image.

A robust spatial similarity framework should be able to recognize image variants such as translation, scaling, rotation, and arbitrary variants. Image variants may be perfect or multiple. In a perfect image variant, the same transformation is applied to all image objects by the same magnitude. Otherwise, the variant is a multiple variant. Most current spatial similarity algorithms [9, 20, 21, 48, 57, 55, 56] recognize translation and scaling variants but not rotation variants. A few algorithms [45, 38, 37] has certain constraints. Even fewer algorithms attempt to recognize multiple rotation variants [39].

Current RSS algorithms may be classified into symbolic projection methods, geometric methods, and graph-matching methods. Symbolic projection methods are based on a 2D image representation called the 2D Strings introduced by Chang *et al.* [20]. This representation preserves the object's spatial knowledge embedded in the image. A picture query can also be specified as a 2D string. The problem of pictorial information retrieval then becomes a problem of 2D subsequence matching. Various extensions of 2D strings such as the 2D G-String [14], the 2D C-string [55, 56], and the 2D C^+string [48] have been proposed to deal with situations of overlapping objects with complex shapes. An algorithm for similarity retrieval, based on the 2D string longest common subsequence (LCS) was proposed by Lee *et al.* [57]. The 2D string

LCS problem was transformed to the maximal complete subgraph (clique) problem that requires non polynomial time complexity. In general, all methods based on 2D Strings can only recognize translation and scaling image variants but not rotation variants.

In Chang and Lee [9], each symbolic image is represented as a set of ordered triplets (O_i, O_j, r_{ij}), where O_i and O_j are two symbolic objects and r_{ij} is the spatial relationship between O_i and O_j. All spatial relationships are exhaustively enumerated and stored in a hashing table. By searching the pre-constructed hashing table for all (O_i, O_j, r_{ij})s associated with a query, the pictures satisfying that query can be determined. Similarity is based on exact match which is a severe limitation and it only recognizes translation and scale variants but not rotation variants.

Geometric methods use the inherent geometric features in the image for image representation such as those based on the Weighted Center of Mass (WCOM) [45] and θR-Strings [37]. In Hou *et al.* [45], the center-of-mass of each individual object is used to calculate the center-of-mass of the overall image. Each object is then assigned a feature vector based on its area, the length and the angle from its center to the image center of mass and neighboring object. Object indexing was incorporated using existing indexing structure techniques. To avoid the high dimensionality, only the first four significant objects were considered for indexing. This might lead to true dismissals if one of the significant objects is not present in the query image. Another geometry-based image representation, the θR-String was proposed [37]. Each image was represented by a set of objects where each object has an id, centroid, left and right neighbors and left and right distances. Objects are ordered by the angle between the edge joining their centroid to the image center-of-mass and the x-axis (Θ) and by their distances from the image centroid (R) in case of ties.

Graph-matching methods represent domain objects included in an image and their spatial relationships by a graph called the spatial orientation graph (SOG). The nodes of that graph represent the domain objects while the edges carry the spatial relation between object pairs. Examples of these methods are SIM_R [39] in which the edges carry the slope of the edge connecting each object pair and the SIM_{vs} algorithm [38]

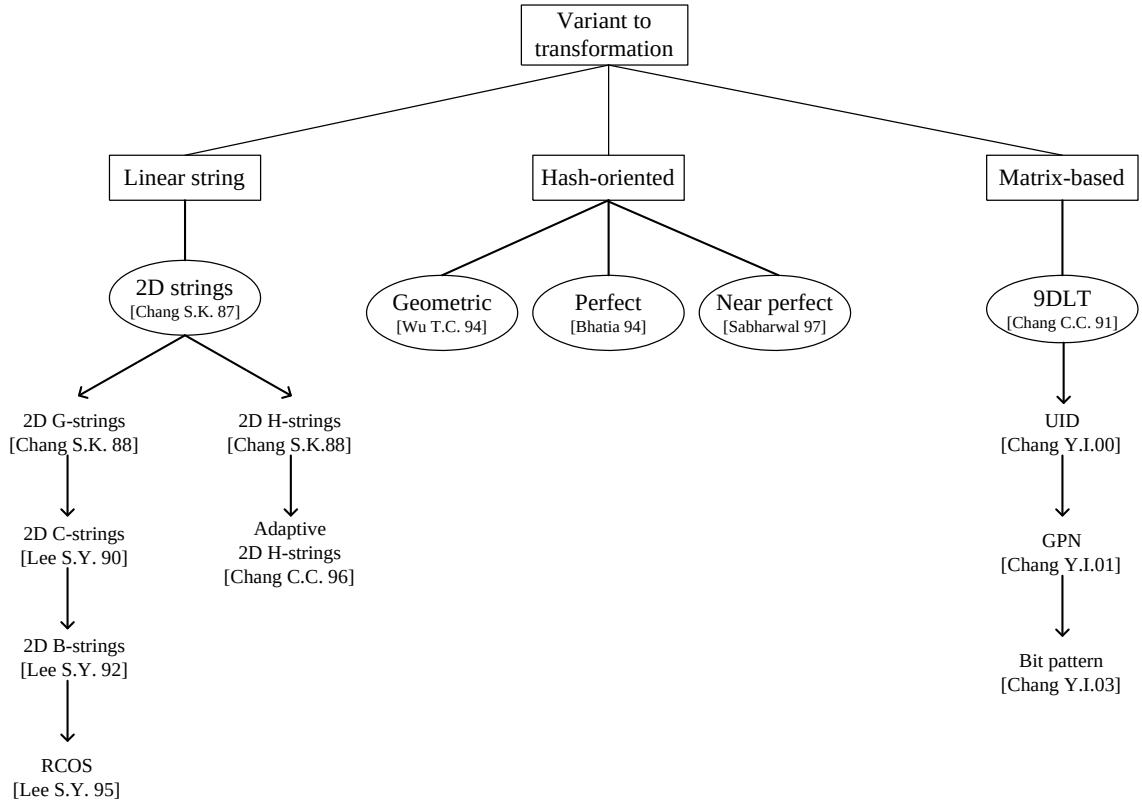


Figure 1.2 Comparison of transformation-variant strategies

which uses Hamiltonian Cycles to represent the order of the objects in the image. Spatial similarity is quantified in terms of the number and the extent to which the edges of the SOG of the database image conform to the corresponding edges of the SOG of the query image. SIM_R can deal with translation, scaling, and perfect rotation image variants. To recognize multiple rotation variants, SIM_R determines the largest subimage that has rotated as a unit using a clustering approach. Rotating all objects in the database image in a direction opposite to that of rotation of the largest group and by equal magnitude would align the database image spatially closer to the query image to yield maximal similarity.

According to the above discussion, the strategies could be classified into two groups. One is *image transformation variant*, the other is *image transformation invariant*. Some representatives in the first and second groups are illustrated in Figures 1.2 and 1.3, respectively.

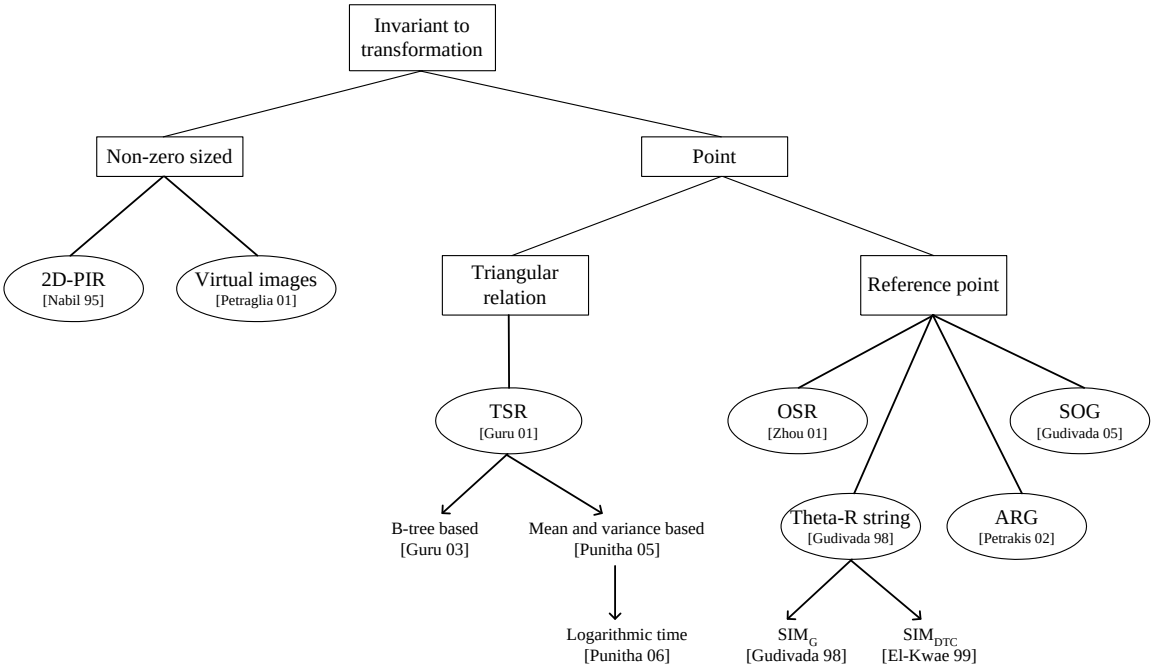
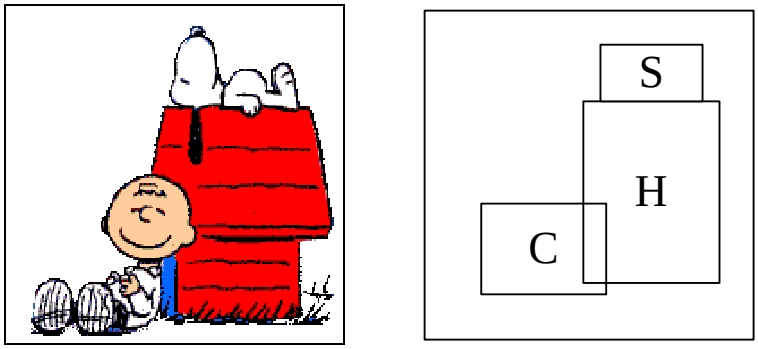


Figure 1.3 Comparison of transformation-invariant strategies

There are three different categories shown in Figure 1.2 based on the data structure used in the strategy. The categories are "linear string", "hash-oriented", and "matrix-based". 2D strings [20] is the first strategy employing the string-like data structure to index images. Following are the some variances [10, 21, 54, 56, 7] of the 2D strings. There are three different kinds of Hash-oriented strategy, they are geometric based strategy [76], perfect hash based strategy [2], and near perfect hash based strategy [68]. 9DLT strategy [6] is the first strategy applying matrix structure to record the spatial relationships. Following is the some extensions [23, 25, 26] of the 9DLT strategy.

Those strategies which are invariant to image transformation could be separated into two group according to the object size. One views objects as nonzero sized ones, the other denotes objects by points based on the centroids of the objects. The second group could be further divided into two groups. Strategies [41, 43, 65, 66] belonging to the first group, *i.e.*, triangular relation, using three centroids to define the spatial relationships. In the second group, *i.e.*, reference point, those strategies



(a) (b)

Figure 1.4 Iconic index: (a) an image picture; (b) the corresponding symbolic representation.

[39, 37, 64, 81] applying two centroids to define the spatial relationships. One of the centroids is called reference point.

1.4 Iconic Indexing Based on Symbolic Projection

Image objects can be represented in various forms. One form is to store significant points from each object such as the centroid or the corners of the minimum bounding rectangle (MBR). MBR is the minimum size rectangle that completely encloses a given object. The MBR is storage efficient and is useful in dealing with image objects that are arbitrarily complex in terms of their boundary shapes. MBR representation is efficient in terms of ascertaining certain topological spatial relationships such as whether or not two objects overlap and whether or not an object is completely contained in another object.

Tanimoto [73] introduced the concept of the iconic index and the use of picture icons as picture indices. For example, Figure 1.4-(a) shows an image picture and the corresponding symbolic representation is shown in Figure 1.4-(b).

The 2D string proposed by S. K. Chang *et al.* [20] is a compact and well-known representation of symbolic pictures. C. C. Chang *et al.* utilized image processing and pattern recognition techniques to transform images into symbolic pictures. Symbolic

pictures are represented in terms of their major components, known as icons. In the icon-based high level spatial reasoning approach, these icons or pictorial objects become the keywords for the description of the image and can be used for indexing the database. The pictorial objects can be extracted by projecting their boundaries in the image onto the x- and y-axes. Consequently, the 2D image is divided into a set of rectangles formed by the intersection of the projections. Each rectangle encloses a pictorial objects within itself and is denoted as a minimum boundary rectangle (MBR). The MBR introduces the concept of encapsulating the positional information of the pictorial objects. The position of an object can be seen as the centroid of the enclosing MBR.

However, the 2D string representation is deficient in describing the spatial knowledge of the nonzero sized objects with overlapping. Hence, Jungert [51] and S. K. Chang *et al.* [16] proposed the 2D G-string representation which introduced more spatial operators and a cutting mechanism to handle more types of spatial relationships among objects in image database. But a 2D G-string representation scheme is not ideally economic for complex images in terms of storage space efficiency and navigation complexity in spatial reasoning, since each overlapping object is partitioned at the begin-bound or end-bound of the other objects. In order to improve the disadvantage caused by 2D G-string, Lee and Hsu [54] proposed 2D C-string with more efficient cutting mechanism. Since the number of subparts generated by this new cutting mechanism is reduced significantly, the lengths of the strings representing pictures are much shorter while still preserving the spatial relationships among objects. The problems of how to infer the spatial relationships between pictorial objects from a given 2D C-string in spatial reasoning and similarity retrieval are solved by using the ranking mechanism. But, the procedure for pictorial query is complicated (which takes $O(N^2)$ time complexity for N nodes). Therefore, Hsu and Lee proposed a 2D C-tree representation [46, 47], which can provide more efficient procedure for pictorial query with time complexity $O(N)$ for N nodes.

In the above various 2D string representations, objects may be partitioned into subparts in order to obtain the spatial relations among objects, especially for a complex image with overlapping objects. If an object is partitioned into subparts and stored in the data structure, the several subparts must be treated together as a whole to infer the knowledge of the object integrally. If there are a large number of subparts, the storage space requirement is high and processing time is long [56]. Therefore, in [58], Lee *et al.* proposed 2D B-string representation to overcome this problem. 2D B-string preserves all the essential spatial information while at the same time provides indexes for the images without the need of partitioning of any objects. By using the ranks of symbols in a 2D B-string, the spatial relationships can be derived easily. However, it is time-consuming to answer the 13 spatial relationship types question based on the 2D-B string representation. Therefore, in [7], C. C. Chang and Lee proposed a new data structure called the Relative Coordinates Oriented Symbolic ($RCOS$) string representation to maintain the advantages of 2D B-strings, and to efficiently check whether a query precisely matches the desired image with one of the 169 varieties of spatial relations between each two objects based on a decision tree. From this data structure, the boundaries of each object can be extracted in linear time. Additionally, one exact type of the 169 possible spatial relationships can be obtained after tracing through two decision trees, each corresponding to a one-dimensional space (x or y-axis). For the number of operations for similarity retrieval by using the 2D B-string and the $RCOS$ string, the $RCOS$ string is observed to greatly reduce the number of operations since at most six comparison operations are needed for $RCOS$ strings while sixteen subtractions and eight multiplication are required in the 2D B-string algorithm proposed in [58].

As described before, based on 2D string representation, the problem of picture query turns out to be the matching of 2D subsequence, which takes non-polynomial time complexity. This makes the picture retrieval method inappropriate for implementation, especially when the number of objects in an image is large. Therefore, in [6], C. C. Chang *et al.* proposed a new approach of iconic indexing by a *nine direction lower-triangular (9DLT) matrix*. In this strategy, a pictorial query can be processed

by using the matrix minus operations; however, only 9 spatial relationships can be handled between any two objects.

In the previous approaches to represent pictorial data, as the complexity of representation strategy is increased, the more spatial relationships can be represented, which also results in a more complex strategy for query processing and a limited types of queries which can be answered. In [24], Y. I. Chang and Yang has proposed a *prime-number-based* matrix strategy, which combines the advantages of the 2D C-string and the 9DLT matrix. Next, Y. I. Chang *et al.* [23] proposed a *unique-number-based* strategy in which each spatial operator is represented as a unique number and a range checking operation is applied to answer a pictorial query.

Based upon the variations of 2D strings or the 9DLT matrix, another data structure (for indexing 2D-strings/9DLT matrix), a set of *triples*, to represent the spatial relationship between each pair of objects in a picture, was proposed [9]. For each triple, a hashing value is found and stored. Hence, the problem of image matching becomes a problem of matching hashing value sequences [2, 9, 8, 80]. Moreover, in order to solve the ambiguity of MBRs, Zhou and Ang [80] combined the topological and directional relationships to introduce another spatial relationships which are hashed in a hashing table to answer the spatial queries. Hash oriented algorithms for the similar match retrieval of symbolic images with $O(n)$ search time were also proposed, where n is the number of symbolic images in the database. However, the database grows unwieldy as the space requirement for the index structure is $O(n^2)$ where n is the number of objects in the database.

On the other way, in [19], S. K. Chang and Li proposed 2D-H strings, which can be viewed as a combination of quadtrees [69] and 2D strings [20]. Using the 2D-H string, the hierarchical symbolic pictures can be represented efficiently in terms of space complexity. Although the 2D-H string data structure has been proven to be an efficient approach to represent and to manipulate symbolic pictures, in [10], C. C. Chang and Lin has discovered some redundancies existing in those data representations. Therefore, they proposed another alternative, called adaptive 2D-H strings, for representing the relationships among the objects in an image [10]. In [10], C. C. Chang and Lin

has presented an algorithm for converting symbolic pictures of any size into adaptive 2D-H strings. They showed that their adaptive 2D-H string can work well for many unbalanced non-square small pictures, which frequently exist in our real environment. In [22], Y. I. Chang and Ann corrected some errors occurring in [10].

1.5 Motivations and Contributions

In the 3D animation field [52, 75], the sequence of images performed by computers are often operated by a number of basic transformations, *e.g.*, rotation and reflection. Figure 1.5-(a) is the original image. Figure 1.5-(b) to Figure 1.5-(f) show the results of five different basic transformations applied to the original one.

As in robotic scenes and virtual reality applications, they require to solve queries like this: "find those images that are satisfied with a given pattern even if it shows in a rotation orientation." However, indexing methods based on symbolic projection are sensitive to rotation or reflection. If we want to retrieve an image which is rotated and stored in the database and we have only the index that represent the image before being rotated, we must have the new index of the rotated image such that we can retrieve the image. There are two approaches to solve this problem. The first approach is that we reconstruct the image from the original index, rotate the image and construct the index of the rotated image as shown in Figure 1.6-(a). In order not to miss the qualified database images, we need to do steps 2 and 3 five times to derive the five indexes, *i.e.*, the indexes of the image rotated by 90, 180, and 270 degrees, and the indexes of the image flipped horizontally and vertically. Figure 1.6-(b) shows the second approach, in which we find a corresponding strategy such that the new index of the rotated image can be constructed by the original index. The first approach is time-consuming as compared to the second approach. Therefore, finding a good strategy such that we can efficiently get the new index of the adjusted image from the original index is important.

Petrakis [64] described that one of classes of the spatial image content representation and matching is *symbolic projection*. However, strategies based on symbolic

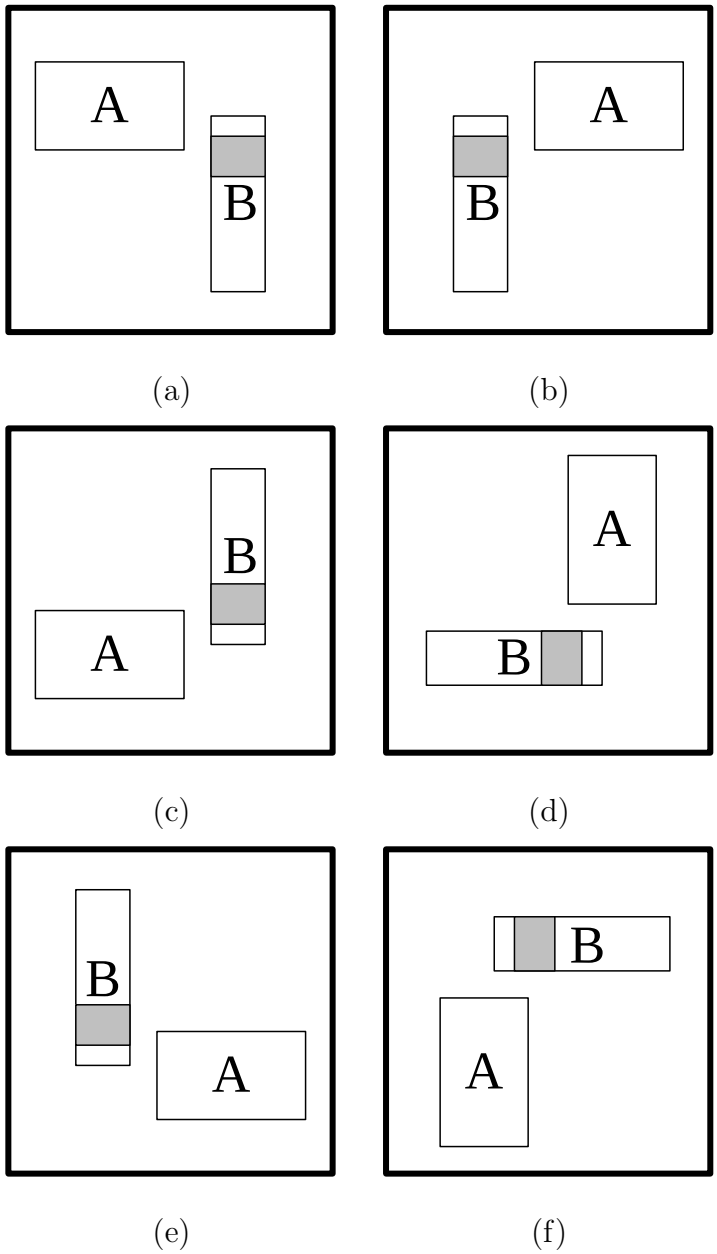
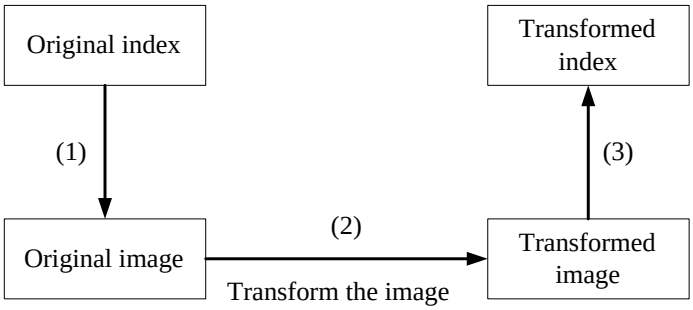
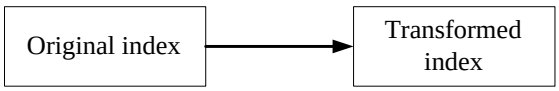


Figure 1.5 Image transformation: (a) the original image; (b) flipped horizontally; (c) flipped vertically; (d) rotated by 90°; (e) rotated by 180°; (f) rotated by 270°.



(a)



(b)

Figure 1.6 The process for obtaining the rotated or flipped index: (a) the original three steps; (b) the second approach.

projection, *e.g.*, [17, 49], cannot recognize similarity between two indexes corresponding to an image and one of its possible transformations, *e.g.*, rotation and reflection. To solve this problem, Nabil *et al.* [61] and Petraglia *et al.* [63] proposed a similar mapping for the rotation and reflection of the spatial relationships. However, the process of the index mapping is time-consuming. Thus, in this dissertation, we classify the mapping into three cases and carefully assign a 16-bit unique bit pattern to each spatial relationship. Based on the assignment, we can easily do the mapping with our proposed bit operation, *intra-exchange*. Moreover, we propose an efficient iconic index strategy, called *Unique Bit Pattern* matrix strategy (UBP matrix strategy) to record the spatial information. In this way, when doing similarity retrieval, we do not need to reconstruct the original image from the UBP matrix in order to obtain the indexes of the rotated and flipped image. Conversely, we can directly derive the

index of the rotated or flipped image from the index of the original one through bit operations and the matrix manipulation. Thus, our proposed strategy can do similarity retrieval without missing the qualified database images. From our performance study, we show that our strategy outperforms those mapping strategies [61, 63] based on different number of objects in an image. The percentage of improvement is between 13.64% and 53.23%.

1.6 Organization of the Dissertation

The rest of the dissertation is organized as follows. Chapter 2 gives a survey of the previous proposed representations for symbolic images. Chapter 3 presents our contributions in detail. In Chapter 4, a simulation study will be made to show that our proposed strategy is efficient. Finally, concluding remarks are made in Chapter 5.

CHAPTER 2

A Survey of Iconic Indexing Strategies

In this Chapter, we give a survey of some iconic indexing strategies. These strategies can be roughly classified into two groups. *2D C-strings* [54], *2D B-strings* [58], *2D-PIR* [61], Zhou and Ang's strategy [80], *UID matrix* [23], *Virtual images* [63], *2D Z-string* [53], and *9D-SPA* [49] belong to the group dealing with nonzero-sized objects. *2D strings* [20], *9DLT matrix* [6], *Triangular spatial relationship* (TSR) [41] and it's subsequent extensions [43, 65, 66] belong to the group viewing objects as points based on their centroids. Specifically, we will describe *UID matrix*, *Virtual images*, and *2D-PIR* strategies in more details, since these strategies are related to our proposed strategy.

Table 2.1 summarizes some characteristics of the strategies in the survey. The first column is the abbreviation name of those strategies which are listed in the order of the published year. The second column shows which type of the objects in an image is. "centroid" means the strategy applies the centroids of the objects to define the spatial relationships. On the other hand, "sized" means the strategy defines the spatial relationships based on the object's length and width. In the third column, it describes whether the spatial relationships among objects are invariant to image transformation, such as rotation and reflection. There is a special notion "specific degree" which means the spatial relationships defined in the strategy is invariant to only the multiple of 90° rotation and reflection. There are two types of image retrieval. One is exact match, the other is similarity retrieval. The fourth column shows the strategy is suited for which type of image retrieval. The last column shows

Table 2.1 Comparison of existing strategies

Strategy	Object	Invariant	Match	Time
2D strings [20]	centroid	Not invariant	Similarity	Non-polynomial
9DLT matrix [6]	centroid	Not invariant	Similarity	$O(n \times m^2)$
2D C-strings [54]	sized	Not invariant	Similarity	Non-polynomial
2D B-strings [58]	sized	Not invariant	Similarity	$O(n \times m^2)$
2D-PIR [61]	sized	specific degrees	Similarity	$O(n \times m^2)$
Zhou's strategy [80]	sized	Not invariant	Similarity	$O(m^2)$
UID matrix [23]	sized	Not invariant	Similarity	$O(n \times m^2)$
Virtual images [63]	sized	specific degrees	Similarity	$O(n \times m^2)$
2D Z-string [53]	sized	Not invariant	Similarity	Non-polynomial
9D-SPA [49]	sized	Not invariant	Similarity	$O(m^2)$
TSR [41]	centroid	Invariant	Exact	$O(\lg n)$
TSR B-Tree [43]	centroid	Invariant	Similarity	$O(m^3)$
TSR Statistics [65]	centroid	Invariant	Exact	$O(\lg n)$
Logarithmic strategy [66]	centroid	Invariant	Exact	$O(\lg n)$

the complexity of the search time, where m is the number of objects in the query and n is the total number of images in the database.

2.1 2D Strings

In pictorial information retrieval, we often want to retrieve pictures satisfying a certain picture query, for example, "find all pictures having a tree to the left of the house." S. K. Chang *et al.* [20] present a new way of representing a picture by a *2D string*. A picture query can also be specified as a 2D string. The problem of pictorial information retrieval then becomes a problem of 2D subsequence matching.

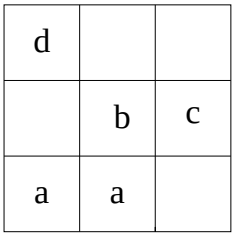


Figure 2.1 A picture f

This approach thus allows an efficient and natural way to construct iconic indexes for pictures.

Let V be a set of symbols, or the vocabulary. Each symbol could represent a pictorial object (a named object such as "house", "tree", etc.) or a pixel. Let A be the set $\{" =", " <", " :"\}$, where "=", "<" and ":" are three special symbols not in V. These symbols will be used to specify spatial relationships between pictorial objects.

For example, consider the picture shown in Figure 2.1. The vocabulary is $V = \{a, b, c, d\}$. The 2D (x, y) string representing the above picture f is as follows:

$$(\{a\} = \{\} = \{d\} < \{a\} = \{b\} = \{\} < \{\} = \{c\} = \{\},$$
$$\{a\} = \{a\} = \{\} < \{\} = \{b\} = \{c\} < \{d\} = \{\} = \{\}).$$

Where the symbol "<" denotes the left-right or below-above spatial relationship, and the symbol "=" denotes the spatial relationship "at the same spatial location as."

The corresponding *absolute 2D string* is as follows:

$$(a == d < a = b =<= c =, a = a =<= b = c < d ==).$$

The corresponding *normal 2D string* is as follows:

$$(ad < ab < c, aa < bc < d).$$

The same procedure can be applied to pictures whose "slots" may contain multiple objects (i.e., object sets). For example, if in Figure 2.1, another object e is added to the same slot with d, then the corresponding *normal 2D string with sets* is as follows:

$$(ad : e < ab < c, aa < bc < d : e).$$

Where the symbol ":" denotes the relationship "in the same set as."

The corresponding *reduced 2D string* is as follows:

$(ade < ab < c, aa < bc < de)$.

A picture is defined to be *ambiguous* if there exists a different reconstructed picture g from its 2D string representation (x, y). To reduce the ambiguity, the permutation function p is included in the *augmented 2D string* representation. The corresponding augmented 2D string of the picture f is as follows:

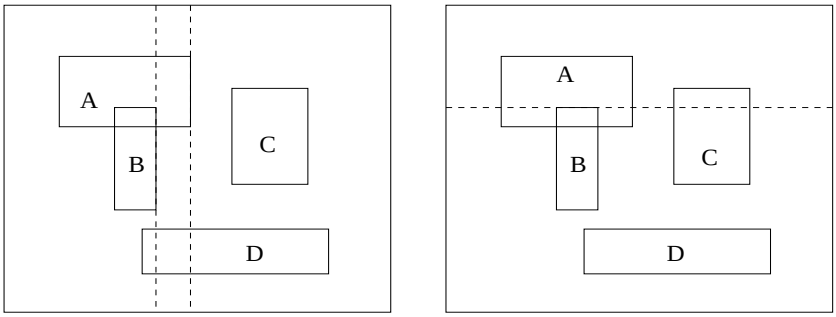
$(ade < ab < c, aa < bc < de, 145623)$,

$(ade < ab < c, 14 < 56 < 23)$.

2.2 2D C-Strings

Although 2D G-strings can represent the spatial relationships among objects in pictures and spatial reasoning can be carried out effectively on generalized 2D strings using a set of reasoning rules, there still exist some problems. One important observation is that the number of segmented subparts of an object is dependent of the number of bounding lines of other objects which are completely or partly overlapping with this targeted object. For the cases of objects with overlapping, the storage space overhead is high and it is time consuming in spatial reasoning if pictures are represented in 2D G-string. Therefore, to overcome this problem, *2D C-string* is proposed by Lee and Hsu [54]. The same set of spatial operators are employed to perform a more efficient and economic cutting mechanism, and have a sound and characteristic description for a picture.

Table 2.2 shows the formal definition of the set of spatial operators, where the notation "begin(A)" denotes the value of begin-bound of object A and "end(A)" denotes the value of end-bound of object A. According to the begin-bound and end-bound of the picture objects, spatial relationships between two enclosing rectangles along the x-axis (or y-axis) can be categorized into 13 types ignoring their length. Therefore, There are 169 types of spatial relationships between two rectangles in 2D space, as shown in Figure 2.3. Basically, a cutting of the 2D C-string is performed at the point of partly overlapping, and it keeps the former object intact and partitions the latter object. The cutting mechanism is also suitable for pictures with



(a) (b)

Figure 2.2 The cutting mechanism of 2D C-string: (a) cut along the x-axis; (b) cut along the y-axis.

many objects. Furthermore, the end-bound point of the dominating object does not partition other objects which contain the dominating object. Less cuttings and no unnecessary cuttings in 2D-C string will make the representation more efficient in the case of overlapping as shown in Figure 2.2. The corresponding 2D C-string is as follows:

2D C_x-string(f): $A](B|D|D)|D\%C$.

2D C_y-string(f): $D < B]C]A|A$.

2.3 2D B-Strings

In *2D G-strings* [51] and *2D C-strings*, the cutting mechanism is used for solving the problems of overlapping by segmenting an object to many subparts. But the partition process is quite time-consuming and it also creates a large storage overhead for the spatial reasoning on spatial queries. The *2D B-strings* strategy [58], on the other hand, does not require that the objects be partitioned. The spatial relationships are derived by using the *ranks* of symbols in the *2D B-strings*.

Let S be a set of symbols. Each symbol could represent the *Begin boundary* or *End boundary* of a pictorial object. The projection of each object is described by its two boundaries, *begin* and *end* boundaries. The only special symbol "=" not in S is

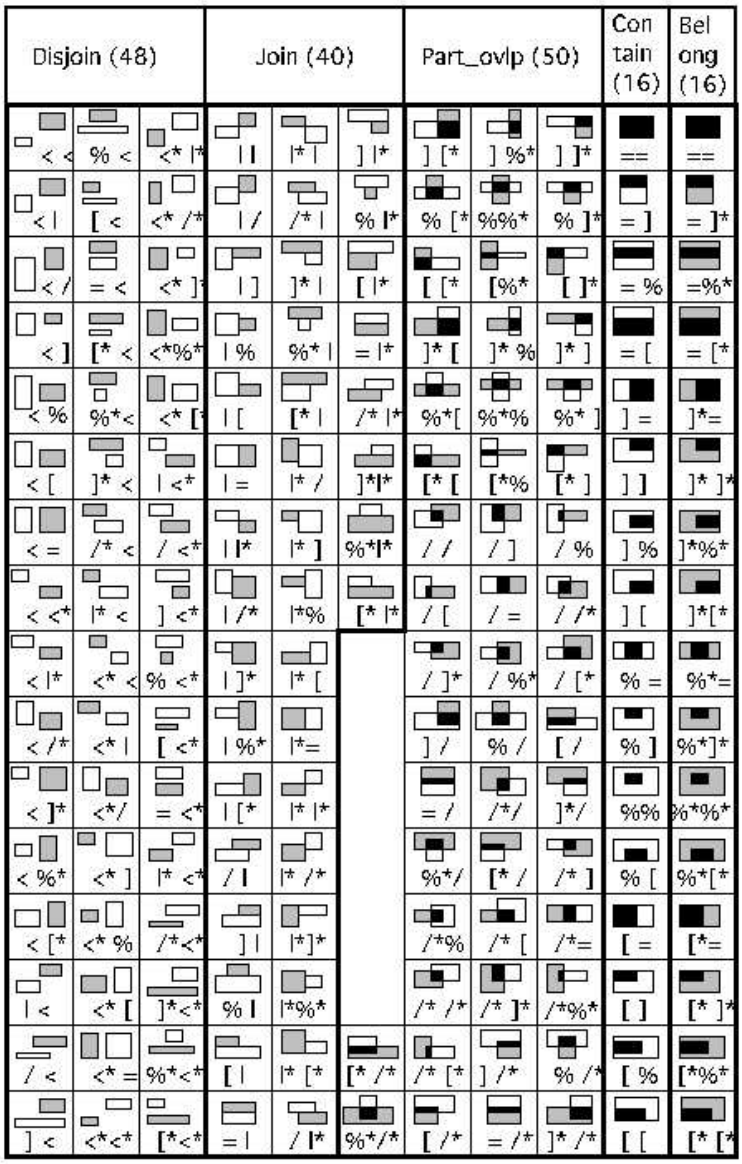


Figure 2.3 The 169 spatial relationship types of two objects

Table 2.2 Definitions of Lee *et al.*'s spatial operators

Notation	Condition	Meaning
$A < B$	$\text{end}(A) < \text{begin}(B)$	A disjoins B
$A\|B$	$\text{end}(A) = \text{begin}(B)$	A is edge to edge with B
A/B	$\text{begin}(A) < \text{begin}(B)$ $< \text{end}(A) < \text{end}(B)$	A is partly overlapping with B
$A]B$	$\text{begin}(A) < \text{begin}(B)$ $\text{end}(A) = \text{end}(B)$	A contains B and they have the same end bound
$A[B$	$\text{begin}(A) = \text{begin}(B)$ $\text{end}(A) > \text{end}(B)$	A contains B and they have the same begin bound
$A\%B$	$\text{begin}(A) < \text{begin}(B)$ $\text{end}(A) > \text{end}(B)$	A contains B and they do not have the same bound
$A = B$	$\text{begin}(A) = \text{begin}(B)$ $\text{end}(A) = \text{end}(B)$	A is at the same position as B

used to specify spatial relationship, when the projections of objects have the same boundary.

A *2D B-string* over S is written as (x, y), where x and y are 1D string over $S \cup \{=\}$ representing the boundary of the projection of objects along x- and y-axis, respectively. For example, the *2D B-string* to represent the picture f in Figure 2.4 is:

x: ABCABC

y: CC=AA=BB

The *rank* of each symbol in a string x, which is defined as the position of this symbol minus the number of "=" preceding this symbol in x string. The position of the first object symbol in 2D B-string is set to 1 and the symbol "=" is not counted in the calculation of the positions of object symbols. The projection of each object

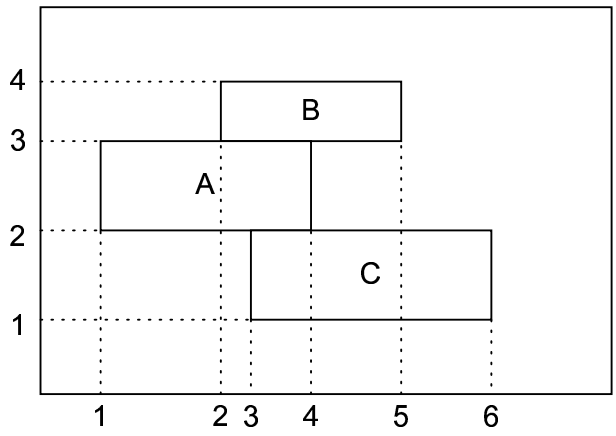


Figure 2.4 An example f

is described by two symbols, representing begin and end boundary. Therefore, each object has two ranks, begin rank and end rank, along x- and y-axis directions, respectively. For example, the ranks of each object in Figure 2.4 are:

x:

rank-begin(A)=1, rank-end(A)=4,

rank-begin(B)=2, rank-end(B)=5,

rank-begin(C)=3, rank-end(C)=6.

y:

rank-begin(A)=2, rank-end(A)=3,

rank-begin(B)=3, rank-end(B)=4,

rank-begin(C)=1, rank-end(C)=2.

Hence, the rank of each object can be represented as (i, j, k, l), where (i, j), (k, l) represent the ranks of begin boundary and end boundary along x- and y-axis , respectively. Thus, the above example can be written as:

rank(A)=(1, 4, 2, 3),

rank(B)=(2, 5, 3, 4),

rank(C)=(3, 6, 1, 2).

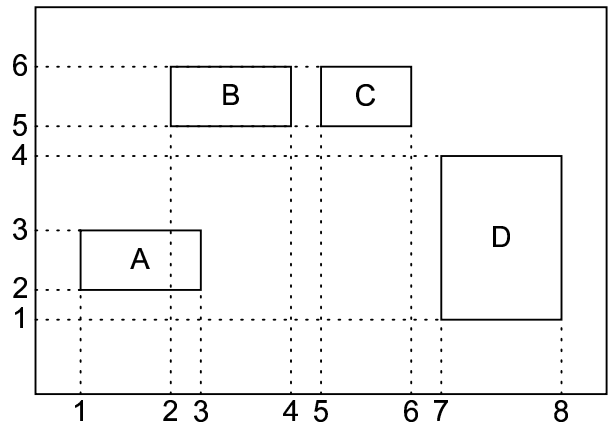


Figure 2.5 An example p

It can be observed that some symbols in *2D B-string* might be omitted and the spatial relationships between objects still keeps the same. The condition is satisfied when the projections of objects are disjoin or the projections of objects are at the same location. In this case, an object can be viewed as a point, *i.e.*, begin boundary and end boundary meet as one point. For example, the x and y strings

x: ABABCCDD
y: DAADB=CB=C

in Figure 2.5 can be reduced as

x: ABABCD
y: DADB=C

The representation is called *reduced 2D B-string*. It is worth noting that the symbol in *reduced 2D B-string* might represent the boundary of an object or the object itself depending on the symbol appearing once or twice in the x- or y- string.

There are total 13 kinds of spatial relationships between two objects in one dimension as illustrated in Figure 2.6. If we consider x- and y- axis directions independently,

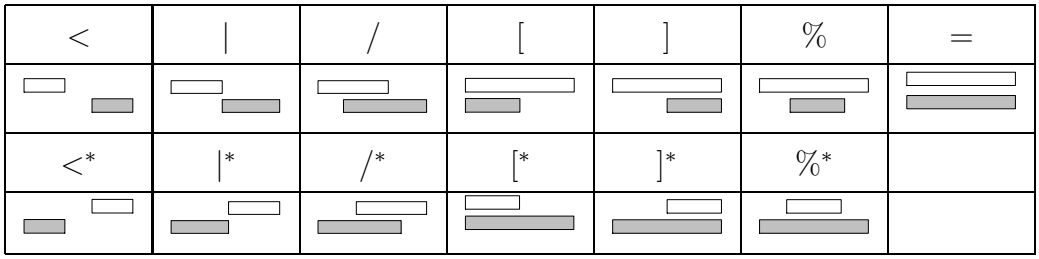


Figure 2.6 13 types of spatial operators in one dimension (horizontal projection)

there are total 169 kinds of spatial relationships between two objects in two dimension. They can be classified into five spatial categories, *disjoin, join, contain, belong,* and *partial overlapping.* The criterion for the spatial categories is the intersection area between each two objects. Figure 2.7 shows some examples of those five spatial categories.

2.4 2D Projection Interval Relationships

A 2D Projection Interval Relationship (2D-PIR) [61] is a symbolic representation of directional and topological relationships among objects in an image. Allen's temporal intervals [1] and 2D-strings [20] are adapted and combined in a novel way to produce the unified representation.

An interval-based temporal representation and a method to derive relationships between intervals are proposed by Allen [1]. Intervals can be represented by their endpoints. Based on this representation, 13 temporal relationships (7 relationships have inverses, one relationship has no inverse) are derived as shown in Table 2.3.

The project concept from the 2D-String representation is applied to 2D-PIR. The difference from the 2D-string representation is the Allen's interval relationships is employed over the projection s of the entire objects along x and y axes. Thus, instead of a *string*, a *graph* is used to represent the relationships among objects in a picture. A 2D-PIR graph is a connected labeled digraph $G(V, R)$ where V is a finite non-empty set of symbols representing objects in a picture and R is a set of edges labeled by 2D-PIR relationships.

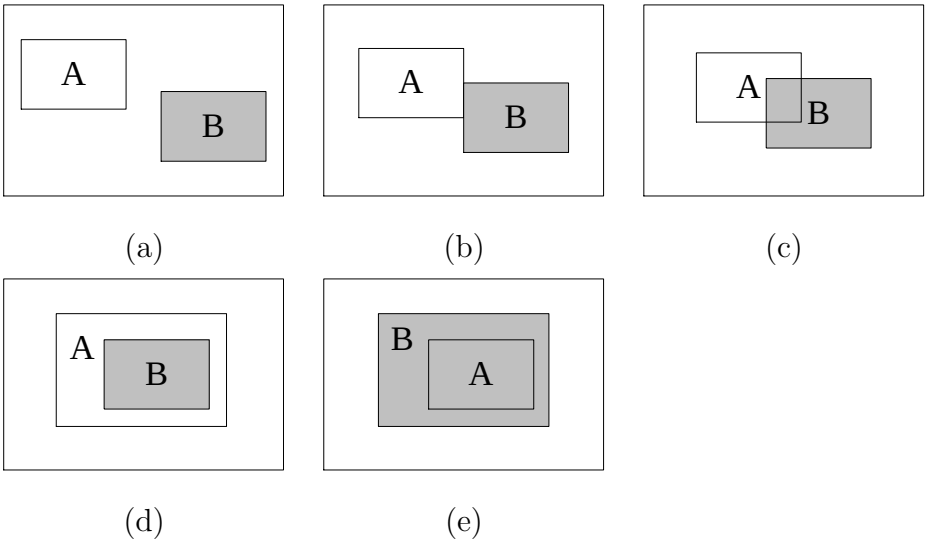


Figure 2.7 Examples of spatial categories: (a) disjoin; (b) join; (c) partial overlapping; (d) contain; (e) belong.

Relationship	Symbol	Inverse
X *before* Y	$<$	$>$
X *equal* Y	$=$	$=$
X *meet* Y	m	mi
X *overlaps* Y	o	oi
X *during* Y	d	di
X *starts* Y	s	si
X *finishes* Y	f	fi

Table 2.3 Allen's interval relationships

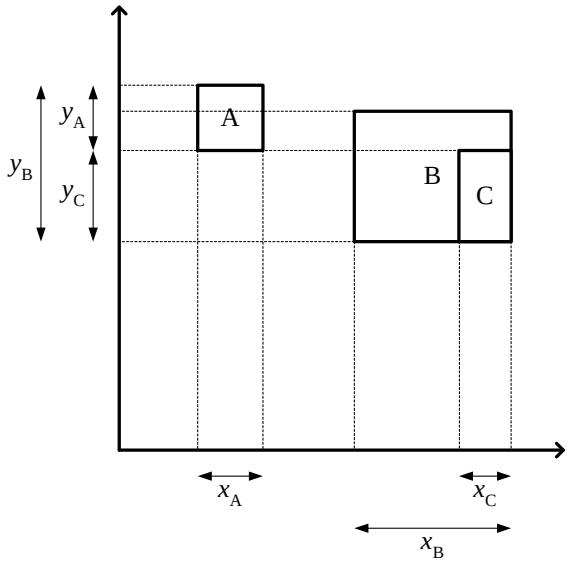


Figure 2.8 2D-projection image

A 2D-PIR between two spatial objects A and B is a triple (δ, χ, ψ). δ is a topological relationship from the set $\{dt, to, ct, in, ov, co, eq, cb\}$. These represent the topological relationships *disjoint*, *meets*, *contain*, *inside*, *overlaps*, *covers*, *equal*, *covered_by*. The domain of χ and ψ are $\{<, =, m, o, d, s, f, >, mi, oi, di, si, fi\}$ which are symbols for interval relationships. χ represents the interval relationship between objects A and B projected along the x-axis, and ψ represents the interval relationship between objects A and B projected along the y-axis.

For example, in Figure 2.8, 2D-PIR between A and B is $(dt, <, oi)$, between A and C is $(dt, <, m)$, and between B and C is (co, fi, si). Let us consider the relationships between B and C. The 2D-PIR between B and C among x and y axes are (fi, si). This means B covers C and C is located on the bottom-right of B. Thus, we can infer that C is covered by B on the bottom-right. The corresponding 2D-PIR graph of Figure 2.8 is shown in Figure 2.9.

The degree of similarity between two images represented using 2D-PIR image is dependent on the degree of similarity between their corresponding 2D-PIR relationships. For example, the degree of similarity between $(\delta_1, \chi_1, \psi_1)$ and $(\delta_2, \chi_2, \psi_2)$ is dependent on the degree of similarity between δ_1 and δ_2, χ_1 and χ_2, and ψ_1 and ψ_2.

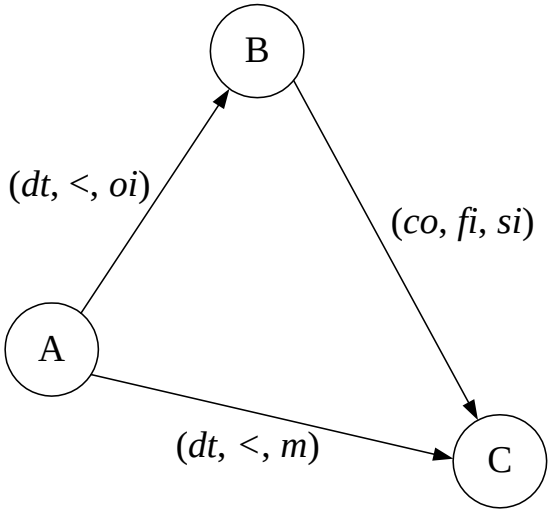


Figure 2.9 2D-PIR graph

Figure 2.10 shows the topological neighborhood graph, and the interval neighborhood graph is shown in Figure 2.11. Based on distance between any two relationships shown in Figures 2.10 and 2.11, we can use the following formula to determine the similarity metric between the 2D-PIR relationships to do the similarity retrieval.

$$D(r_1, r_2) = \sqrt{D(\delta_1, \delta_2)^2 + D(\chi_1, \chi_2)^2 + D(\psi_1, \psi_2)^2}$$

A similarity algorithm which is invariant for matching a image with another image where the latter image is a rotated version or reflected version of the first image this strategy is proposed in the 2D-PIR strategy. Suppose A and B are two objects in an image (I). Let (δ, χ, ψ) be a 2D-PIR relationship between A and B in I. Let ξ be a function such that

$$\xi(g) = \begin{cases} gi & \text{if } g \in \{m,\, o\} \\ g & \text{if } g \in \{d,\, di,\, =\} \\ h & \text{where } g = hi \text{ and } g \in \{mi,\, oi\} \\ k & \text{where } g \in \{s,\, si\} \text{ and } k \in \{f,\, fi\} \\ l & \text{where } g \in \{f,\, fi\} \text{ and } l \in \{s,\, si\} \\ > & \text{if } g = < \\ < & \text{if } g = > \end{cases}$$

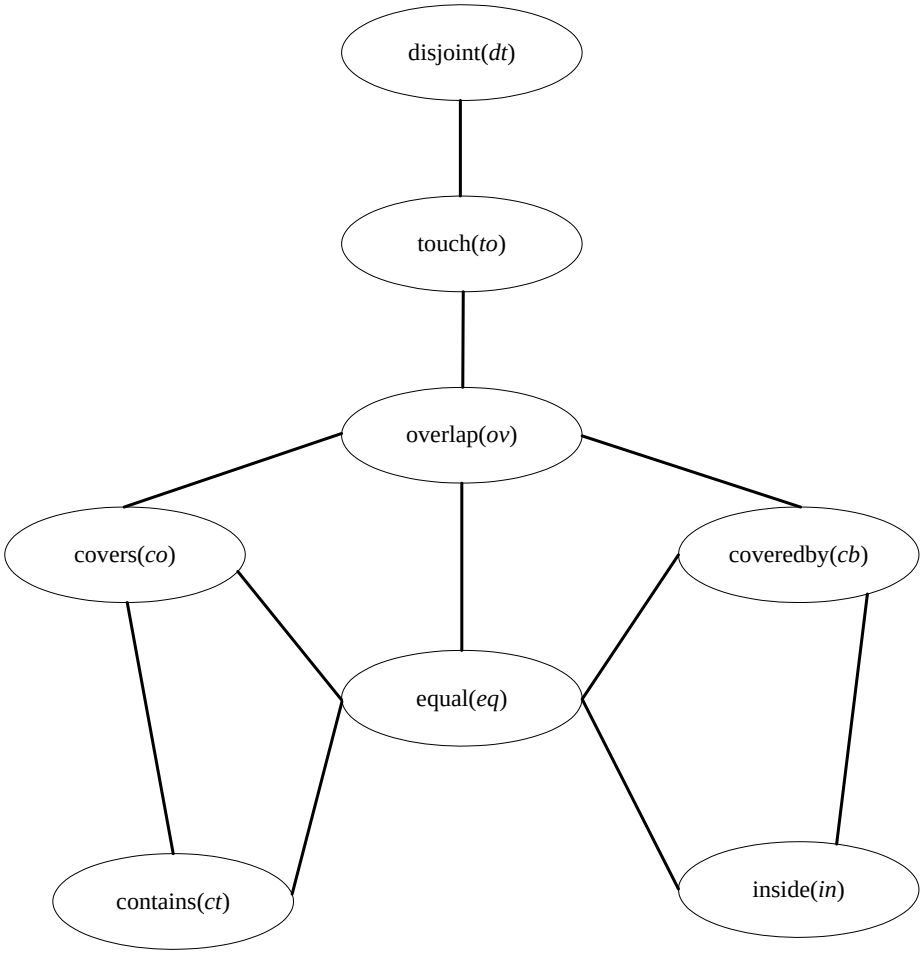


Figure 2.10 Topological neighborhood graph

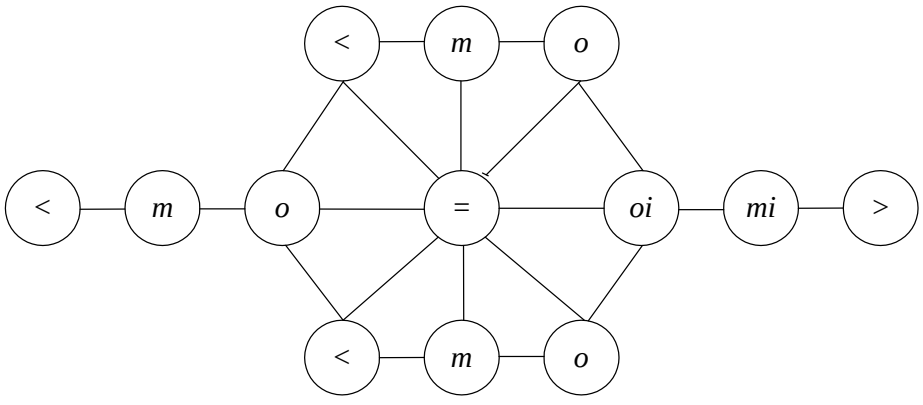
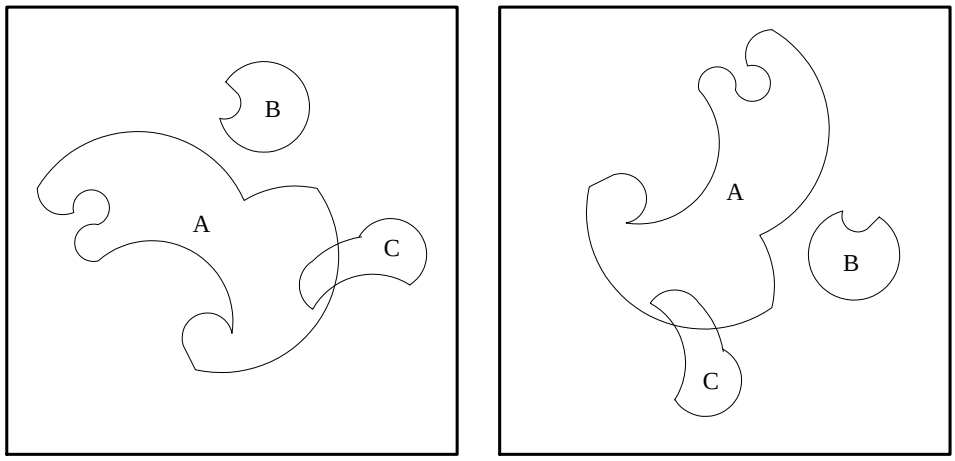


Figure 2.11 Interval neighborhood graph



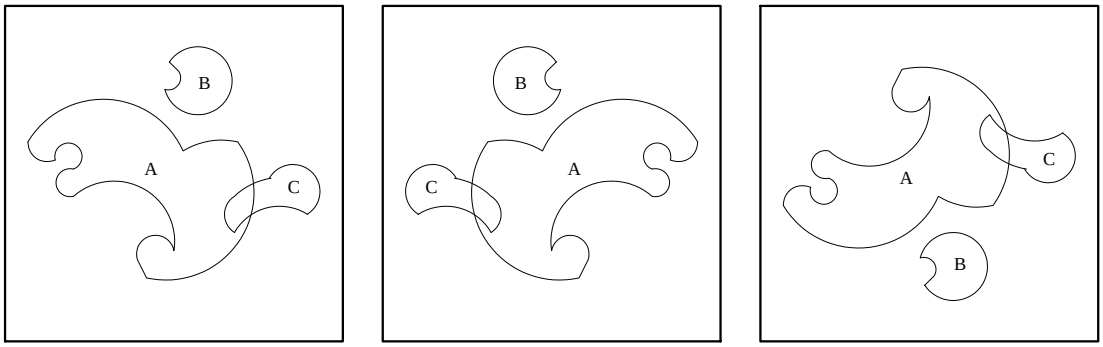
(a) (b)

Figure 2.12 Example of the rotation transformation: (a) original image (I); (b) the 270° rotated version (Q).

The following statement holds: If (δ, χ, ψ) is a 2D-PIR relationship between two spatial objects in an image then $(\delta, \xi(\psi), \chi)$ is the 2D-PIR relationship between objects in the 90° counterclockwise rotated version of the image. For any rotation of a picture which is a multiple of 90° counterclockwise rotation, the interval relationships can be computed as a composition of 90° rotations.

For example, Figure 2.12 shows an image (I) and a version of the image that has undergone a 270° counterclockwise rotation. The 2D-PIR relationships between AB, AC, and BC in I are $(dt, di, <)$, (ov, o, di), and $(dt, <, >)$, respectively. Transform these relationships via a 270° rotation transformation, *i.e.*, do the 90° transformation three times, the 2D-PIR relationships are $(dt, <, di)$, (ov, di, oi), and $(dt, >, >)$ which are the 2D-PIR relationships between AB, AC, and BC in Q, respectively.

With respect to reflection, the following statement holds: If (δ, χ, ψ) is a 2D-PIR relationship between two spatial objects in an image, then $(\delta, \chi, \xi(\psi))$ is the 2D-PIR relationship between the objects in a version of the image that has been reflected in the x-axis. $(\delta, \xi(\chi), \psi)$ is the 2D-PIR relationship between objects in the version of the same image reflected in the y-axis.



(a) (b) (c)

Figure 2.13 Example of the reflection transformation: (a) original image (I); (b) vertical reflection (I_v); (c) horizontal reflection (I_h).

For example, Figure 2.13 shows an image I and its vertical (I_v) and horizontal (I_h) reflected versions. The 2D-PIR relationships between A and B, A and Z, and B and C in I are the same as the those shown in Figure 2.12-(a). If we transform these relationships by vertical reflection, the 2D-PIR relationships are $(dt, di, <)$, (ov, oi, di), and $(dt, >, >)$, which are the 2D-PIR relationships between A and B, A and C, and B and C in I_v, respectively. Similarly, if we transform these relationships by horizontal reflection, the 2D-PIR relationships are $(dt, di, >)$, (ov, o, di), and $(dt, <, <)$ which are the 2D-PIR relationships between A and B, A and C, and B and C in I_h, respectively.

2.5 Zhou and Ang's *DT* Approach

The approach of iconic indexing by a 2D string for spatial reasoning was proposed by S. K. Chang *et al.* [20]. To represent a picture symbolically, the objects in the original image must be recognized first. These objects are usually then enclosed by minimum boundary rectangles (*MBR*s). However, sometimes, it is hard to describe the spatial relationship of the objects in terms of that between their corresponding *MBR*s. In order to avoid the ambiguity problem in visualization caused by *MBR*s, Zhou and Ang proposed a new spatial knowledge representation and a hashing table

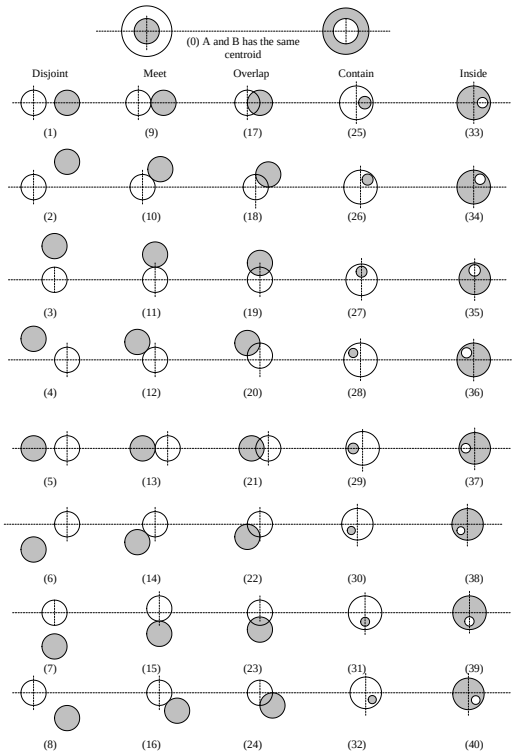


Figure 2.14 The 41 types of spatial relations in two dimensions

by combining the directional and topological information into one representation [80]. We call this approach, the DT approach.

There are 41 types of spatial relationships in 2D space displayed in Figure 2.14. In this approach, a pairwise spatial relationship $r_{i,j}$ (range between 0 to 40) between two objects O_i and O_j is represented as a triplet $(O_i, O_j, r_{i,j})$. Then, the algorithm calculates $h(O_i, O_j) = V_i + V_j$ and $hr(O_i, O_j, r_{i,j}) = h(O_i, O_j) + (r_{i,j}/100)$, where hr is called *symbol hashing value* (SHV). Finally, a hashing table is used to organize the pictures containing the pairs (O_i, O_j). The hashing function h is used to be the offset into the table of an entry that points to a tree-like structure with 41 branches corresponding to the 41 relationships. The branch for $r_{i,j}$ in the tree-like structure is a linked list of those pictures containing $(O_i, O_j, r_{i,j})$. The hashing table is shown in Figure 2.15.

We give an example to illustrate the generation of symbol hashing value and demonstrate how the similarity retrieval can be done based on it. The symbol set

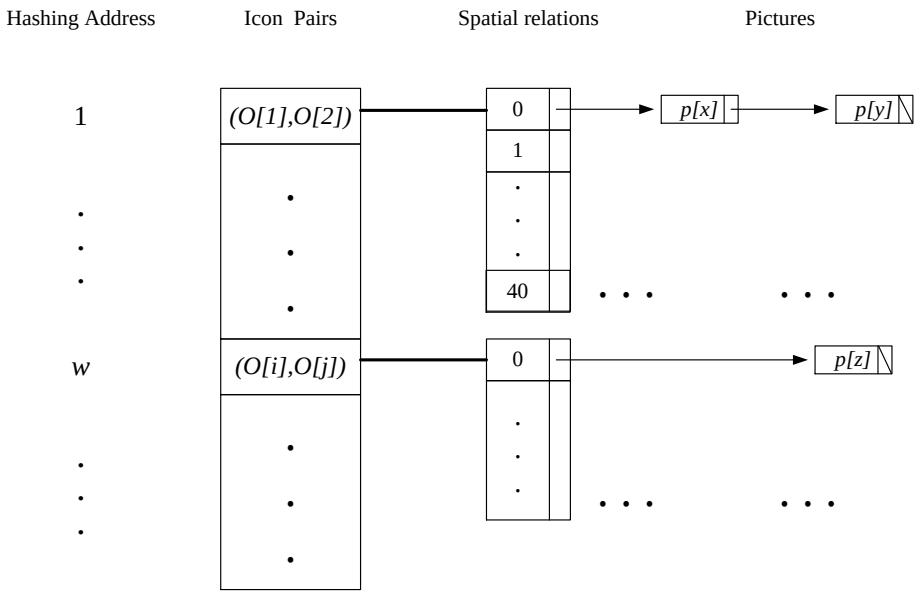


Figure 2.15 The hashing table for a pictorial database

of the picture P in Figure 2.16-(a) is $\{A, B, C, D, E, F\}$. The triplets of P are { *(A,B,3)*, *(A,C,3)*, *(B,C,17)*, *(A,D,3)*, *(B,D,9)*, *(C,D,33)*, *(A,E,3)*, *(B,E,3)*, *(C,E,3)*, *(D,E,3)*, *(A,F,3)*, *(B,F,3)*, *(C,F,3)*, *(D,F,3)*, *(E,F,18)* }. The values for symbols O_i, h and hr are shown in Table 2.4.

Assume there are 6 pictures in pictorial database. The hashing table of the database is shown in Figure 2.17, where picture P has the picture number 1. In our example, we only show a part of this structure to be referenced in the similarity retrieval. Given a query Q which is shown in Figure 2.16-(b), its symbol hashing value set is $\{5.09, 8.03, 11.03\}$. According to the algorithm discussed in the DT approach, for each hr_i in the above set, let $h_i = \lfloor hr_i \rfloor$, and $r'_i = (hr_i - h_i) \times 100$, and access through the table entry corresponding to h_i and the r'_ith branch to obtain the list $L[(h_i, r'_i)]$. Then, we have $L[(5, 9)] = \{1, 4, 6\}$, $L[(8, 3)] = \{1, 3, 5\}$, and $L[(11, 3)] = \{1\}$. Finally, we have $Answer = \{1, 4, 6\} \cap \{1, 3, 5\} \cap \{1\} = \{1\}$. So only picture P that matches the query picture Q will be retrieved from the database.

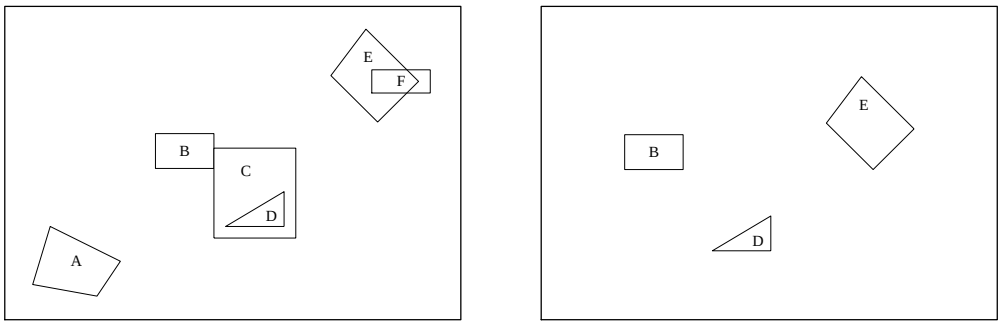


Figure 2.16 An example : (a) picture P; (b) query picture Q.

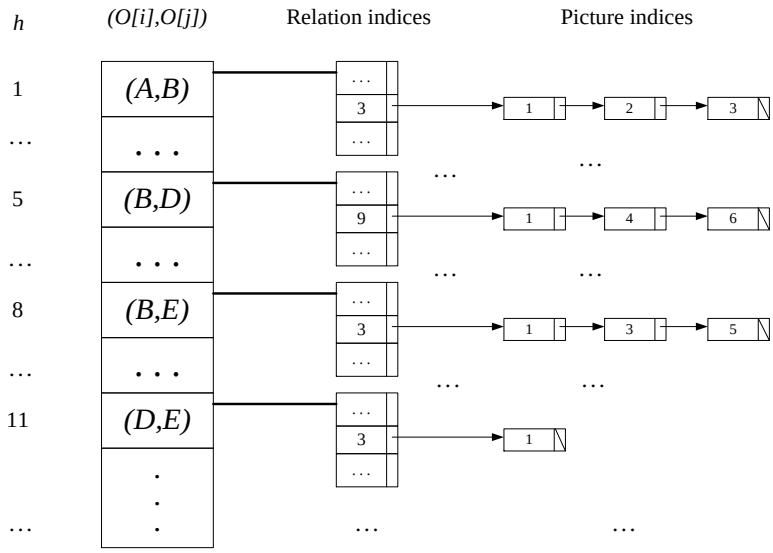


Figure 2.17 Part of the hashing table of the database

Table 2.4 Symbol hashing value construction

Symbols	Symbol value (V_m)	Icon pairs	r_{ij}	h	hr
A	0	*(A,B)*	3	1	1.03
B	1	*(A,C)*	3	2	2.03
C	2	*(B,C)*	17	3	3.17
D	4	*(A,D)*	3	4	4.03
E	7	*(B,D)*	9	5	5.09
F	12	*(C,D)*	33	6	6.33
		(A,E)	3	7	7.03
		(B,E)	3	8	8.03
		(C,E)	3	9	9.03
		(D,E)	3	11	11.03
		(A,F)	3	12	12.03
		(B,F)	3	13	13.03
		(C,F)	3	14	14.03
		(D,F)	3	16	16.03
		(E,F)	18	19	19.18

$h = V_i + V_j$

$hr = h + (r_{i,j}/100)$

```
Function CATEGORY(uid^x_{A,B}, uid^y_{A,B})
1       if (uid^x_{A,B} > 4) and (uid^y_{A,B} > 4) then
2           if (7 ≤ uid^x_{A,B} ≤ 10) and (7 ≤ uid^y_{A,B} ≤ 10) then
3               return ('C')
4           else if (10 ≤ uid^x_{A,B} ≤ 13) and (10 ≤ uid^y_{A,B} ≤ 13) then
5                   return ('B')
6               else return ('P')
7       else if (uid^x_{A,B} > 2) and (uid^y_{A,B} > 2) then
8               return ('J')
9           else return ('D')
```

Figure 2.18 The CATEGORY function

2.6 An Unique-ID-Based Matrix Strategy

Chang *et al.* [23] proposed an efficient iconic indexing strategy called unique-ID-based matrix (UID matrix) for symbolic pictures, in which each spatial relationship between any two objects is assigned with a unique identifier (ID) and is recorded in a matrix. Basically, the proposed strategy can represent those complex relationships that are represented in 2D C-strings in a matrix, and an efficient range checking operation can be used to support pictorial query, spatial reasoning and similarity retrieval.

Chang *et al.* assigned each of those 13 spatial operators a unique identifier denoted by *uid* as shown in Table 2.5. Thus, the 169 spatial relationships can be arranged in a way that relationships of the same category are grouped together as shown in Table 2.6. In this way, given two *uid*s ($uid^x_{A,B}$, $uid^y_{A,B}$), to efficiently determine a category, we can use the algorithm as shown in Figure 2.18. The corresponding decision tree is shown in Figure 2.19.

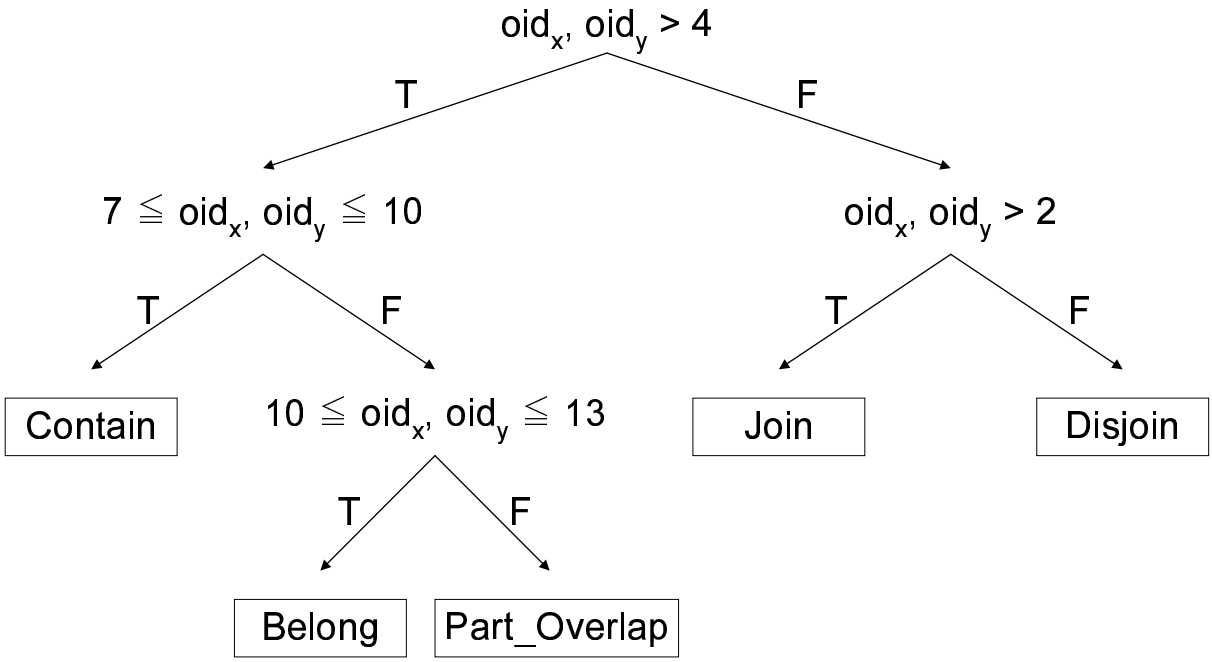
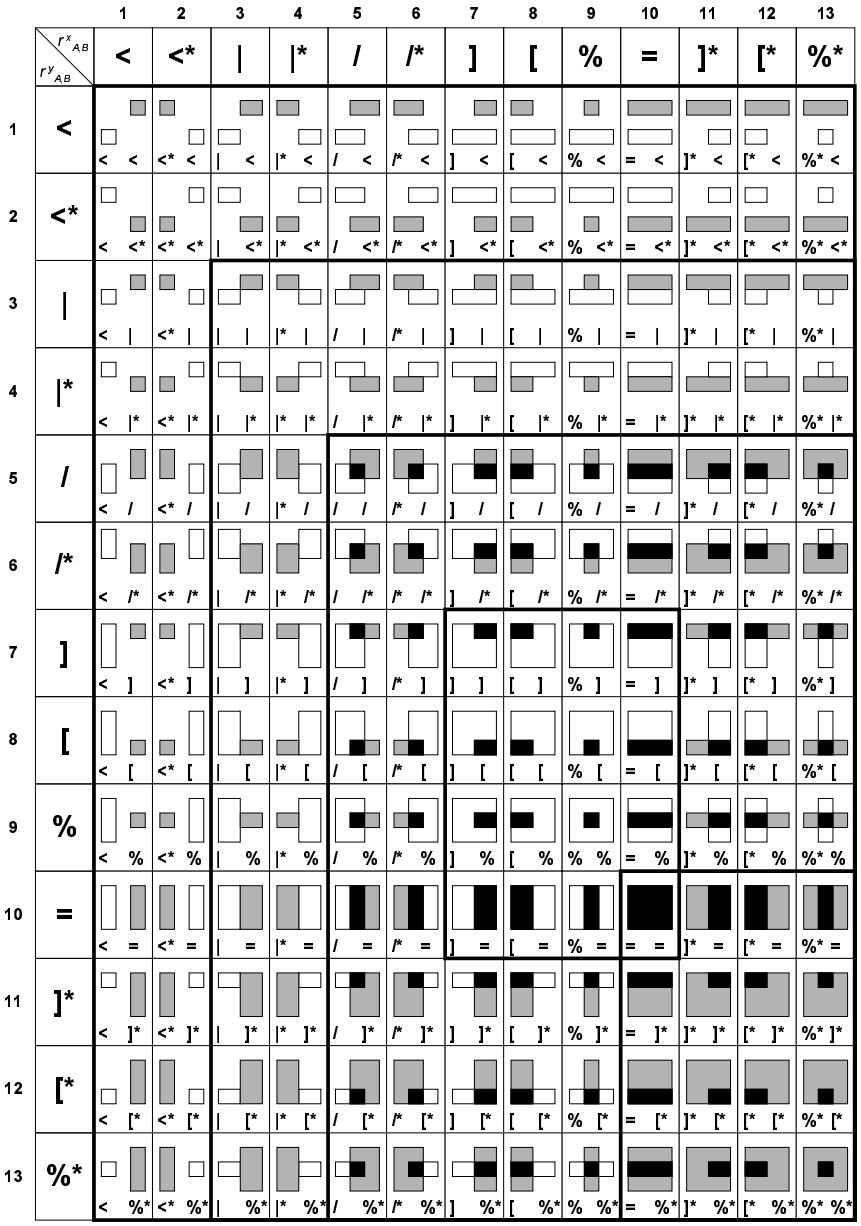


Figure 2.19 Decision tree of the CATEGORY function

Table 2.5 UIDs of 13 spatial operators

operator	$<$	$<^*$	$\|$	$\|^*$	$/$	$/^*$	$]$	$[$	$\%$	$=$	$]^*$	$[^*$	$\%^*$
UID	1	2	3	4	5	6	7	8	9	10	11	12	13

Table 2.6 Category table



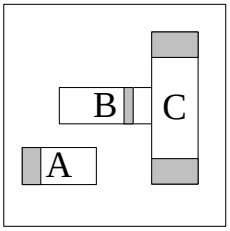


Figure 2.20 An example of a symbolic image presented by MBRs

Suppose an image I contains m objects and let $V = \{v_1, v_2, ..., v_m\}$. Let R be the set of 13 spatial operators $\{<, <^*, |, |^*, [, [^*,],]^*, \%, \%^*, /, /^*, =\}$. A $m \times m$ *spatial matrix* S of image I is defined as follows:

$$S = \begin{array}{c} \\ v_1 \\ v_2 \\ \vdots \\ v_{m-1} \\ v_m \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & v_2 & \cdots & v_{m-1} & v_m \end{array} \\ \begin{bmatrix} 0 & r^y_{1,2} & \cdots & \cdots & r^y_{1,m} \\ r^x_{1,2} & 0 & \ddots & & \vdots \\ \vdots & \ddots & 0 & \ddots & \vdots \\ \vdots & & \ddots & 0 & r^y_{m-1,m} \\ r^x_{1,m} & \cdots & \cdots & r^x_{m-1,m} & 0 \end{bmatrix} \end{array}$$

where the lower triangular matrix stores the spatial information along the x-axis, and the upper triangular matrix stores the spatial information along the y-axis.

For the image shown in Figure 2.20,the corresponding spatial matrix S is shown as follows.

$$S = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} 0 & < & [^* \\ / & 0 & \%^* \\ < & | & 0 \end{bmatrix} \end{array}$$

According to the assignments of the *uid* values for those 13 spatial operators shown in Table 2.5, the spatial matrix S of I can be transformed into a *UID matrix* T as follows:

$$T = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \left[\begin{array}{ccc} 0 & 1 & 12 \\ 5 & 0 & 13 \\ 1 & 3 & 0 \end{array}\right] \end{array}$$

The lower left triangular area of the matrix M records the spatial relationships among the x-axis, and the upper right one records the spatial relationships among the y-axis. For example, the unique identifiers for the objects A and B among the x- and y-axes are 5 and 1, respectively. Thus, this means that the object B is above the object A.

2.7 Virtual Images for Similarity Retrieval

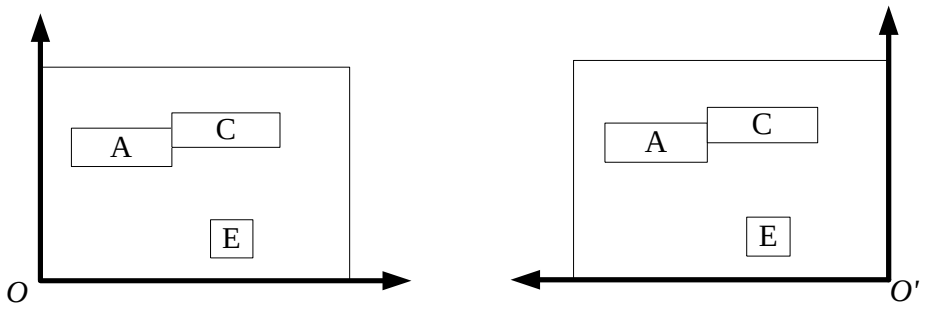
Petraglia *et al.* [63] introduced the concept of a virtual image description of an image, called *virtual image*. The virtual image of a real image consists of a set of objects and a set of binary spatial relationships over the objects. Moreover, a transformation law was proposed to derive the transformation of the virtual image.

Given a real image S, the virtual image P associated with S is a pair (Ob, Rel), where:

- $Ob = \{ob_1, ob_2, \ldots, ob_n\}$ is a set of objects
- $Rel = (Rel_x, Rel_y)$ is a pair of sets of binary spatial relationships over Ob, where Rel_x contains the mutually disjoint subsets of $Ob \times Ob$ that express the relationships as shown in Table 2.2 holding between pairs of objects of S along the x-projection. The Rel_y has the same definition of Rel_x except for the relationships are along the y-projection.

For example, in Figure 2.21-(a), the corresponding virtual image is:

$$Ob = A, C, E;$$
$$Rel_x = \{A|C, A < E, C\%E\};$$
$$Rel_y = \{A/C, E < A, E < C\}.$$



(a) (b)

Figure 2.21 Image S with different observation viewpoint: (a) O' viewpoint; (b) O viewpoint.

In real applications, it would be important to deal with the transformation of images, especially for the change of spatial relationships among objects. As an example, in Figure 2.21-(a), suppose O is the observer viewpoint. The spatial relationships between the objects in Figure 2.21-(a) are different from those in Figure 2.21-(b) where the observer viewpoint is changed to O'. This is because the conventional indexing methods based on orthogonal projections do not provide simple ways to recognize similarity between two indexes corresponding to an image and one of its possible manipulations, say translation, reflection, change of point of view, *etc.*

When reversing the scanning direction, the begin point of an object is exchanged with its ending point. This strategy derives the atomic relationships in terms of relationships between couple of objects. Thus, the atomic relationships of a picture reversed along a given direction can be easily obtained from those of the original image. Table 2.7 summarizes the correspondences existing between the atomic relationships of an image f along the x-direction and the corresponding atomic relationships in an image g obtained by reversing f along the y-axis.

Let f and g be two images such that g is obtained by reversing f along the x-axis, and let $f_{vi} = (Ob_f, Rel_f)$ with $Rel_f = (Rel_{f_x}, Rel_{f_y})$ and $g_{vi} = (Ob_g, Rel_g)$ with $Rel_g = (Rel_{g_x}, Rel_{g_y})$ be their corresponding virtual images. Then, the virtual image g_{vi}

Atomic relationship in f	Atomic relationship in g
A < B	B < A
A = B	B = A
A | B	B | A
A % B	A % B
A [B	A] B
A] B	A [B
A / B	B / A

Table 2.7 The TX-6 transformation law

is given by

$$g_{vi} = (Ob_g, Rel_g) \text{ with } Rel_g = (Rel_{f_x}, TX\text{-}6(Rel_{f_y}))$$

For example, suppose an image S as shown in Figure 2.21-(b) whose virtual image with respect to the O' viewpoint is (Ob, Rel) where

$$Ob = \{A, C, E\};$$
$$Rel'_x = \{C|A, E < A, C\%E\};$$
$$Rel'_y = \{A/C, E < A, E < C\}.$$

When moving the viewpoint from O' toward O as shown in Figure 2.21-(a), the virtual image corresponding to the image S can be obtained by applying the TX-6 transformation law to the Rel_x set as follows:

$$\begin{aligned} Ob &= A, C, E; \\ Rel_x &= TX\text{-}6(Rel'_x) = TX\text{-}6(\{C|A, E < A, C\%E\}) \\ &= \{A|C, A < E, C\%E\}; \\ Rel_y &= \{A/C, E < A, E < C\}. \end{aligned}$$

2.8 2D Z-string

In order to reduce the storage space, Lee *et al.* [53] proposed the 2D Z-string strategy. In this strategy, there is no cutting between objects. Thus, the integrity of objects is preserved and the storage space is bounded to $O(n)$, where n is the number of objects in an image. The shorter the generated string is, the less execution time is required for string generation and image reconstruction.

The objects in an image are projected onto the x- and y-axes to form two strings to represent the spatial relationships between the projections. The locations and sizes of objects, and the distances between objects are recorded in the 2D Z-string representation.

There are 13 spatial relationships in the 2D Z-string strategy. The definition and the corresponding operators of those spatial relationships are the same as which defined in the 2D C-string strategy. Some metric measurements to the knowledge structure of 2D Z-string is added. For examples, in Figure 2.22-(a), A_6 means the size of object A is 6. In Figure 2.22-(b), $A <_3 B$ denotes the distance associated with the spatial operator "<" between objects A and B is 3. In Figure 2.22-(c), $A\%_4B$ denotes the distance associated with the spatial operator "%" between the objects A and B is 4. In Figure 2.22-(d), $A/_7B$ denotes the distance associated with the spatial operator "/" between the objects A and B is 7.

The knowledge structure of 2D Z-string is a 5-tuple (O, T, R_g, R_l, "( )") where

1. O is the set of objects of interest;
2. T is the set of attributes to describe the objects in O;
3. $R_g = \{$"<", "|"$\}$ is the set of global spatial operators;
4. $R_l = \{$"=", "[", "]", "%", "/"$\}$ is the set of local spatial operators;
5. "( )" is a pair of separators which is used to describe a set of objects as a template object.

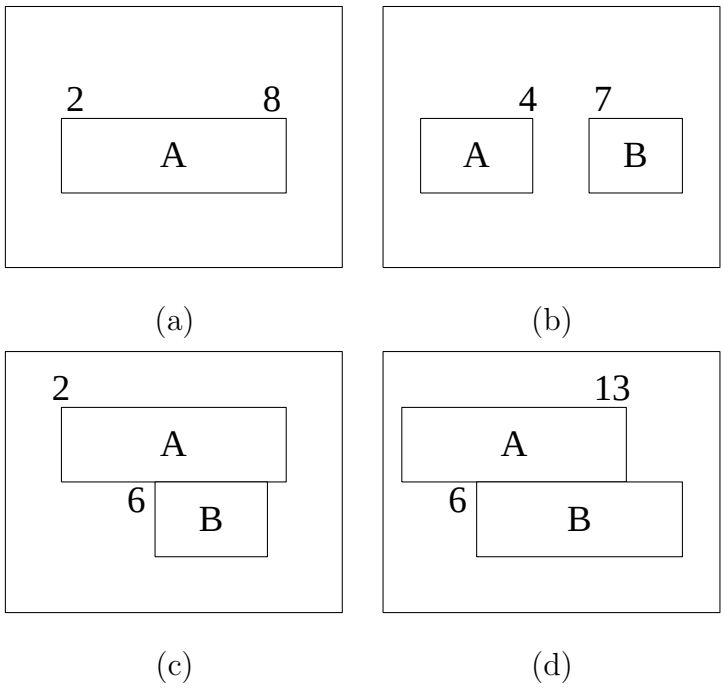


Figure 2.22 Examples of the metric measurements: (a) A_6; (b) $A <_3 B$; (c) $A \%_4 B$; (d) $A /_7 B$.

For example, in Figure 2.23, the image contains 6 partly overlapping objects. The corresponding 2D Z-string is shown as follows:

x-string: $(((A_2/_{0.9}B_{2.7})/_{0.9}C_3) <_{1.4} ((D_{3.4}/_{0.3}E_{4.2})|F_3))$

y-string: $((((A_{3.5}/_{1.5}B_4)/_{4.5}D_6)/_7(E_{7.2}]F_7))/_{2.8}C_{4.6})$

2.9 9D-SPA Representation for Spatial Relationships

Using centroid to represent the position of an object is too sensitive in spatial reasoning. For example, the spatial relationships between the two objects shown in Figures 2.24-(a), 2.24-(b), and 2.24-(c) are all different based on the centroid of the objects. However, they seem not too much different in human visual perception. This is because objects are often nonzero-sized. In order to overcome the above problem,

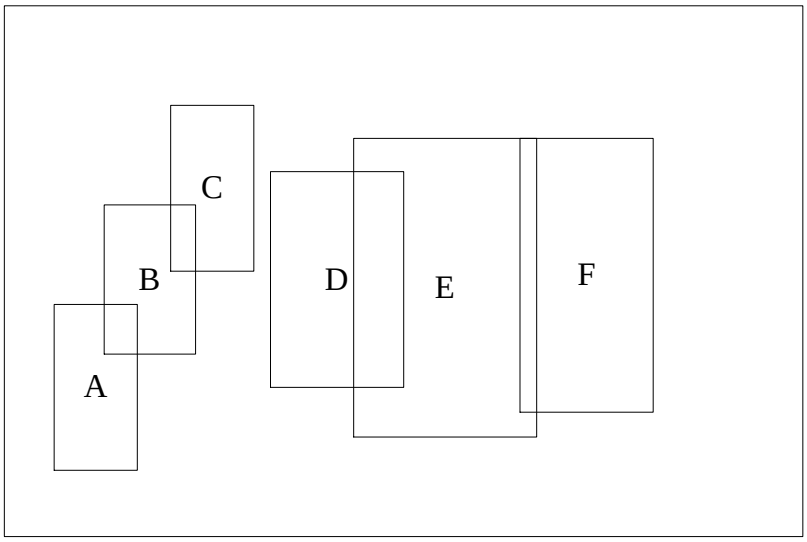
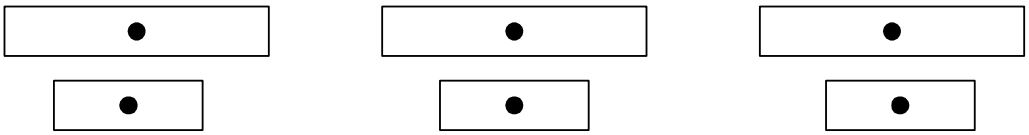


Figure 2.23 Example of the partly overlapping objects



(a) (b) (c)

Figure 2.24 Too sensitive spatial relationship based on centroid: (a) south-west; (b) south; (c) south-east.

Hunag *et al.* proposed the 9-Direction Spanning Area (9D-SPA) strategy [49].

Suppose that there are m objects $(O_1, O_2, \ldots, O_m)$ in an image S. Then, the 9D-SPA representation of S can be encoded as a set of 4-tuples:

$$R = \{(O_{ij}, D_{ij}, D_{ji}, T_{ij}) | \forall O_i, O_j \in S, \text{and } 1 \leq i < j \leq m\}$$

where O_{ij} is the code for object-pair (O_i, O_j), D_{ij} is the code for the direction-relation between objects O_i, O_j with O_j as the reference object, D_{ji} is the code for the direction-relation between O_i and O_j with O_i as the reference object, and T_{ij} is the code for the topological relation between O_i and O_j.

Let O_i be the ith object in the image database $(1 \leq i \leq m)$. The integer i is assigned to the object O_i as its object number. Then, O_{ij} is called the *object-pair code* for object-pair (O_i, O_j). O_{ij} is computed as

$$O_{ij} = \frac{(j-1)(j-2)}{2} + i$$

D_{ij} represents the value assigned to the directional relationship between objects O_i and O_j with O_j as the reference object. The value of D_{ij} is calculated by the following procedure. First, the Minimum Bounding Rectangle (MBR) for reference object O_j is recognized. The four boundaries of this MBR are extended horizontally and vertically until they cut the whole picture into 9 neighborhood areas. Each area is assigned to a binary code as shown in Table 2.8. The value of D_{ij} is determined by the formula

$$D_{ij} = \sum_{k=1}^{8} b_k w_k$$

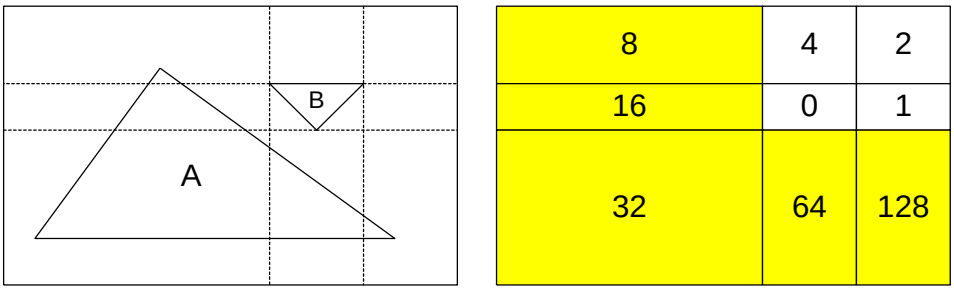
where w_k is the binary code of neighborhood area k; $b_k = 1$ if object O_i overlaps area k, otherwise, $b_k = 0$.

The value of T_{ij} indicates the topological relationship between objects O_i and O_j. Values assigned to the topological relationships are as follows:

- 0 stands for *disjoint*;
- 1 stands for *meet*;
- 2 stands for *partly overlap*;

Area 4: $(00001000)_2 = 8$	Area 3: $(00000100)_2 = 4$	Area 2: $(00000010)_2 = 2$
Area 5: $(00010000)_2 = 16$	Area 0: $(00000000)_2 = 0$	Area 1: $(00000001)_2 = 1$
Area 6: $(00100000)_2 = 32$	Area 7: $(01000000)_2 = 64$	Area 8: $(10000000)_2 = 128$

Table 2.8 Codes for 9 neighborhood areas of MBR of O_j



(a) (b)

Figure 2.25 Example of 9D-SPA representation strategy: (a) the symbolic image; (b) overlapping areas.

- 3 stands for *cover*;
- 4 stands for *contain* or *inside*;

For example, in Figure 2.25-(a), the code for D_{AB} is

$$(00001000 + 00010000 + 00100000 + 01000000 + 10000000)_2 = 248$$

and the code for T_{AB} is 0. Moreover, from $D_{AB} = 248 = (11111000)_2$, we can easily determine that object A spans five neighborhood areas of object B, *i.e.*, the northwest, the west, the southwest, the south, and the southeast neighborhood areas as shown in Figure 2.25-(b).

A two-level index structure to facilitate image retrieval is incorporated into the 9D-SPA strategy. An example of such an index structure is shown in Figure 2.26. There are two levels of indexes in this index structure based on 9D-SPA representation. The first-level index, called the *level*-1 *index array* containing two pointers. One points

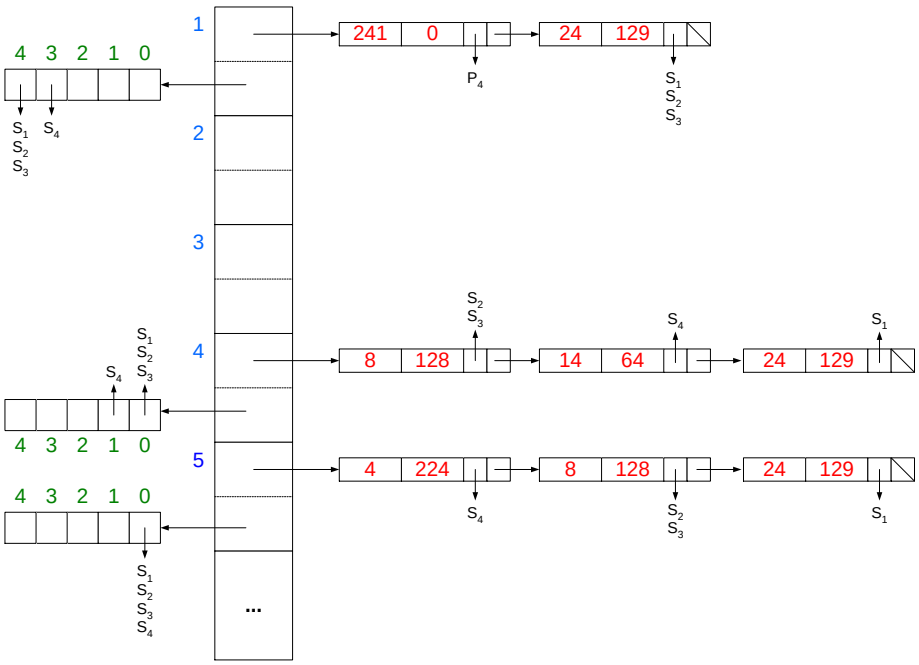


Figure 2.26 Example of index structure for a pictorial database

to a list of directional relation-codes, the other points to an array, called "topological relation array". For example, in Figure 2.26, suppose the tuple (4, 14, 64, 1) of the 9D-SPA representation belongs to the image S_4. Then, there are links to S_4 in the direction-code list (14, 64) and the topological relation array pointed by the 4th entry of the level-1 index array.

2.10 9DLT Matrix

In 2D string representation, the problem of picture query turns out to be the matching of 2D subsequences. Later, Lee *et al.* [57] suggest that by finding the 2D strings' longest common sequences, "similar" pictures can be retrieved. The algorithm still takes non-polynomial time complexity. This makes their picture retrieval method inappropriate for implementation, especially when the number of objects in an image is large. In [6], C. C. Chang *et al.* propose an approach of *iconic indexing* by a *nine direction lower-triangular (9DLT) matrix*.

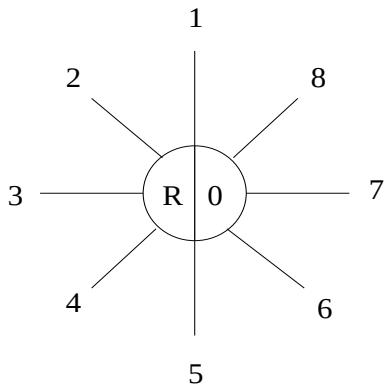
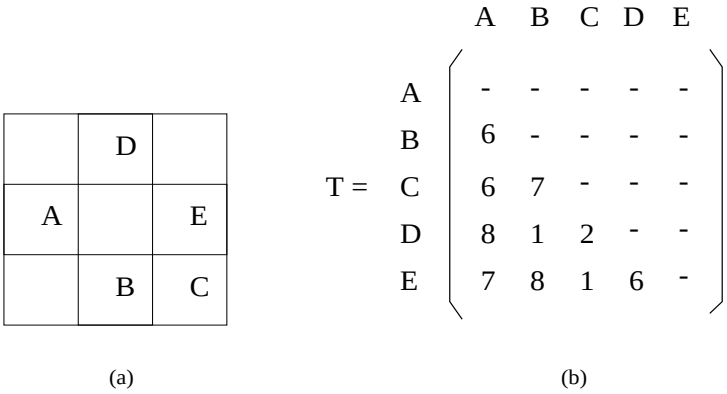


Figure 2.27 The direction codes



(a) (b)

Figure 2.28 9DLT representation: (a) a symbolic picture; (b) the related 9DLT matrix.

Given a well-constructed symbolic picture, our main concern is how to design a data structure to store the symbolic picture. Let there be nine direction codes as shown in Figure 2.27 used to represent relative spatial relationships among objects. In Figure 2.27, R denotes the referenced object, 0 represents "at the same spatial location as R", 1 represents "north of R", 2 represents "north west of R", 3 represents "west of R" and so on. Let us consider a symbolic picture shown in Figure 2.28-(a). Figure 2.28-(b) is the 9DLT matrix of this picture.

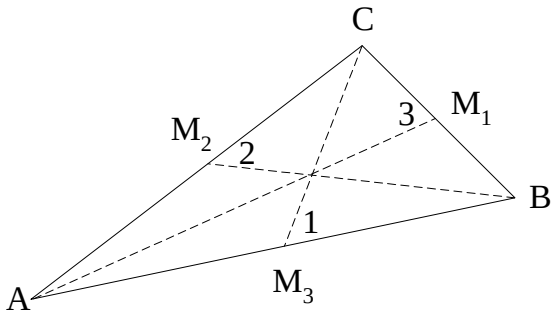


Figure 2.29 Triangular spatial relationship

2.11 Triangular Spatial Relationship

The triangular spatial relationship (TSR) proposed by Guru *et al.* [41] is invariant to object transformations, such as translation, rotation, scaling, and flipping. A TSR is defined by connecting three non-collinear objects in a symbolic image. For example, in Figure 2.29, connecting the centroids of the objects, A, B, and C, forms a triangle. M_1, M_2, and M_3 are the midpoints of the sides of the triangle. The angles labelled as 1, 2, and 3, are the smaller angles subtended at M_1, M_2, and M_3, respectively. The quadruple, (A, B, C, 3), is the triangle spatial relationship among the objects, A, B, and C. In a symbolic image, the triangular spatial relationships existing among every possible combination of three non-collinear objects are computed and represented by a set S of quadruples. In general, the size of S is $O(C_3^T - N_c)$, where T is the total number of objects and N_c is the number of triples of objects which are collinear. In order to minimize the storage requirement, the first principal component vector of the set S is used to represent the symbolic image. These vectors representing the symbolic images are stored in a sorted sequence. Thus, this strategy requires only $O(\log n)$ search time in the worst case, where n is the number of symbolic images.

2.12 Archival and Retrieval Based on The B-Tree Structure

Guru *et al.* [43] proposed a strategy based on triangular spatial relationships to make the representation invariant to image transformations. A unique and distinct number called *key* is generated as the representative of the corresponding quadruple to be stored in the B-tree. Thus, the problem of symbolic image representation is reduced to the problem of storing those quadruples such that the retrieval task becomes effective and efficient.

If (A, B, C, θ) is a quadruple, then the corresponding key K is computed as

$$K = D_\theta(A-1)m^2 + D_\theta(B-1)m + D_\theta(C-1) + (C_\theta - 1),$$

where D_θ is the number of slices of the continuous interval $[0^\circ \ldots 90^\circ]$ associated with θ is split into, and C_θ is the slice number to which a specific value of θ belongs. Let N be the total number of distinct quadruples generated due to all n symbolic images and let $\{K_1, K_2, \ldots, K_N\}$ be the set of corresponding keys. All these N keys are stored in a B-tree. Each key value is then attached with a list of image indexes. The image indexes that are attached to a key value are the indexes of images, which have the key K as one of the keys in their corresponding key set.

For example, let us consider $n = 5$ symbolic images shown in Figure 2.30. According to the TSR strategy, the number of quadruples associated with S_1 is 4 and they are, $\{(3, 2, 1, 67.238292), (4, 3, 1, 71.400327), (4, 3, 2, 84.925637), (4, 2, 1, 48.009121)\}$. The fourth component, which is a real value, of each quadruple is mapped into its slice index. In this example, the interval $[0^\circ \ldots 90^\circ]$ has been split into 18 slices of size 5°. Thus, the quadruples become, $\{(3, 2, 1, 14), (4, 3, 1, 15), (4, 3, 2, 17), (4, 2, 1, 10)\}$. For each distinct quadruple, a distinct and unique key is generated. Hence, S_1 can be described by a set of key values as $\{2461, 3758, 3778, 3609\}$. A B-tree of order $r = 4$ is constructed to store the distinct keys as shown in Figure 2.31. For the key value, 3578, the list obtained is the list containing only the image indexes 1, 2, and 4.

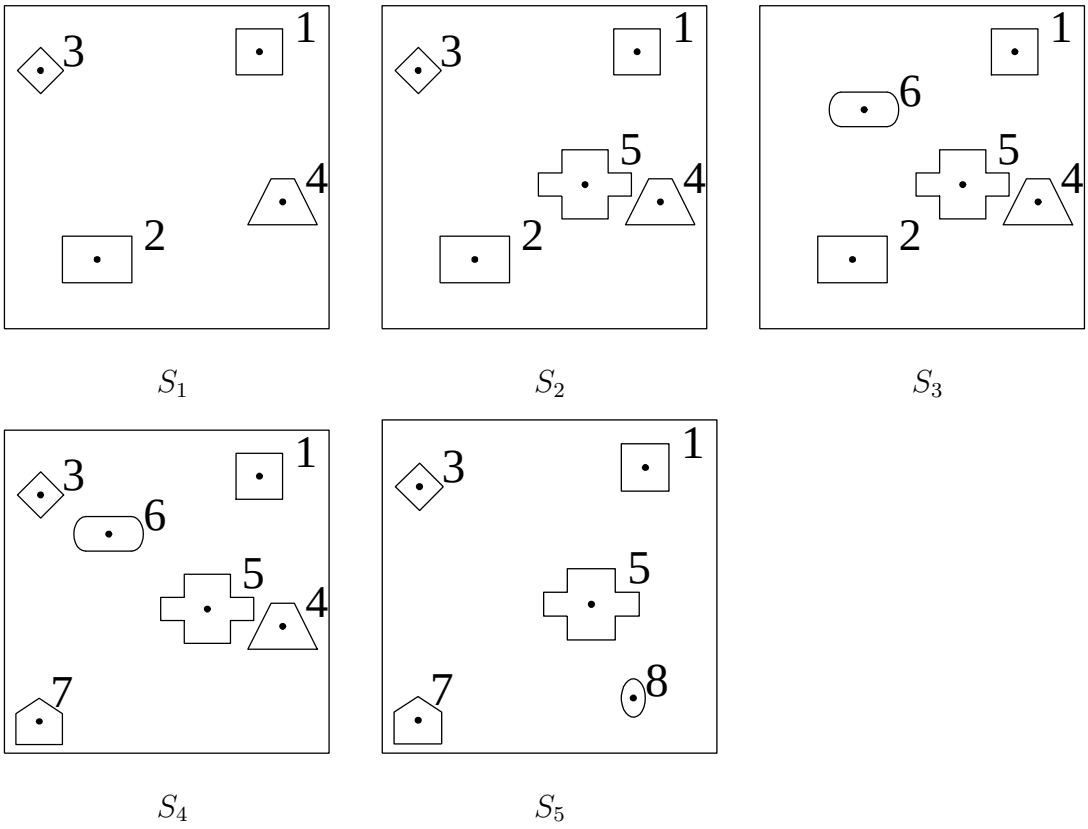


Figure 2.30 Five symbolic images

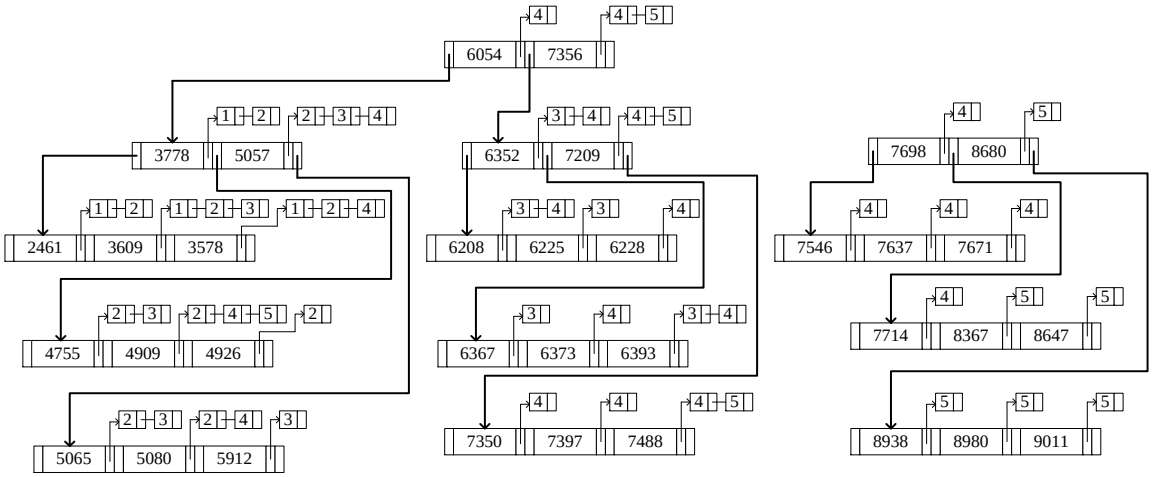


Figure 2.31 B-tree representation

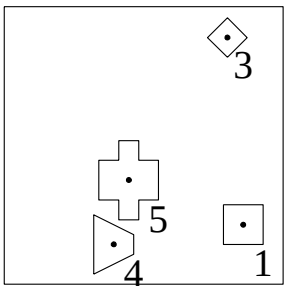


Figure 2.32 Query image

In similarity retrieval,let Q be a query image and q_Q be the number of keys associated with Q. Accessing through the B-tree to obtain the list of image indexes corresponding to each query key and then computing the intersection of all those lists would produce the list of indexes of images similar to the query image Q. It is required $O(q_Q \log_r N)$ search time to obtain all those lists, where N is the total number of keys stored in the B-tree and r is the order of the tree.

For example, consider a query image shown in Figure 2.32. The corresponding key set to describe the query image is {3758, 5075, 5080, 4909}. In order to retrieve the images similar to the query image, we access through the B-tree in search of each key and then we extract the list of image indexes attached to them. They are, {1, 2, 4}, {2, 3, 4}, {2, 4}, and {2, 4, 5}. The intersection of the extracted lists give us only images S_2 and S_4 that match the query image.

2.13 Archival and Retrieval Based on Statistic Measurements

Punitha *et al.* [65] proposed a strategy invariant to image transformations based on the triangular spatial relationships, useful for exact match retrieval. A distinct and unique key called TSR key is computed. The mean and standard deviation of the set of TSR keys computed for an image are stored as the representative of that image. This strategy requires $O(\log n)$ search time, where n is the number of images in the database.

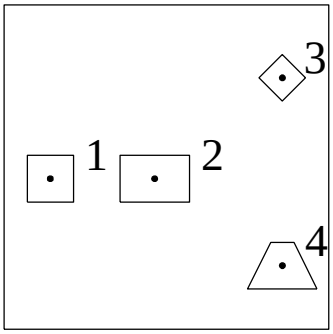


Figure 2.33 Example of statistic measurement-based strategy

Suppose there are three non-collinear objects, A, B, and C, in the image S. (A, B, C, θ) is the triangular spatial relationship among these tree objects. Then the key K corresponding to the quadruple is computed as

$$K = D_\theta(A-1)m^2 + D_\theta(B-1) + D_\theta(C-1) + \theta,$$

where D_θ is the allowable maximum value for θ and here D_θ is 90. The equation generated the key K is a mapping from the set of TSR quadruples to the set of TSR keys. Moreover, this mapping is one-one. That is, the TSR keys associated with two different quadruples are distinct and unique.

Let $P_q = \{q_1, q_2, \ldots, q_N\}$ be the set of N distinct quadruples and $P_K = \{K_1, K_2, \ldots, K_N\}$ be the set corresponding TSR keys generated for symbolic image S. The size of the set P_K is $O(m^3)$ in the worst case, where m is the number of objects in S. To reduce the storage requirement, the mean μ and the standard deviation σ of the set P_K are computed. Then, the triplet (N, μ, σ) is stored as the representative vector of S.

For example, the set of quadruples preserving the triangular spatial relationship among the objects shown in Figure 2.33 is $P_{q0} = \{(4, 2, 1, 26.02), (3, 2, 1, 26.02), (4, 3, 1, 90.00), (4, 3, 2, 90.00)\}$. For each quadruple in P_{q0}, the set of computed TSR key is $P_{k0} = \{18026.021, 12266.021, 18810.00, 18900.00\}$. The corresponding triplet is (4, 17000.511, 3180.6383).

For all n images to be archived in the database, the triplets (N, μ, σ) are computed and stored in a sorted sequence so that binary search can be employed during retrieval.

This strategy not only reduces the size of the database, but also enhances the efficacy of the retrieval process requiring only $O(\log n)$ search time in the worst case.

2.14 A Logarithmic Search Time Strategy for Exact Match Retrieval

Punitha *et al.* [66] a strategy for exact match retrieval of $O(\log n)$ search time in the worst case, where n is the total number of images in the database. The spatial relationships among objects are perceived with respect to the direction of reference [42] and preserved by a set of triples. A distinct and unique key is computed for each distinct triple. The mean and standard deviation of the set of keys computed for an image are stored as the representative of the corresponding image.

Let S be an image consisting of m number of objects. Suppose O_p and O_q to be two objects at the farthest distance, *i.e.*,

$$dist(O_p, O_q) = max\{dist(O_i, O_j) | \forall i, j \in \{1, 2, \ldots, m\}\},$$

where $dist()$ is a distance function which computes the Euclidean distance between the centroids of two objects. The line joining the objects O_p and O_q is the *line of reference* and its direction from the object O_p to the object O_q is the *direction of reference* for S. If (x_p, y_p) and (x_q, y_q) are the coordinates of the centroids of O_p and O_q, respectively, then we defined

$$\alpha = \tan^{-1}\left(\frac{y_q - y_p}{x_q - x_p}\right)$$

and

$$\beta = \sin^{-1}\left(\frac{y_q - y_p}{dist(O_p, O_q)}\right)$$

The direction θ of the line joining O_p and O_q is given by

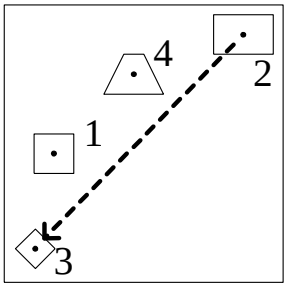


Figure 2.34 Image with direction of reference

$$\theta = \begin{cases} \alpha + \pi & \text{if } \alpha < 0 \text{ and } \beta > 0; \\ \alpha - \pi & \text{if } \alpha > 0 \text{ and } \beta < 0; \\ \alpha & \text{otherwise.} \end{cases}$$

Figure 2.34 shows an image consisting of 4 objects with the direction of reference from O_p to O_q, where $p = 2$ and $q = 3$.

Suppose O_i and O_j are two objects, then the spatial relationship θ_{ij} of the contrast object O_j to the reference object O_i is the relative direction of the line segment joining O_i and O_j, with respect to the direction of reference θ, and it is preserved by the a triple (O_i, O_j, θ_{ij}). Thus, there are $m(m-1)/2$ number of triples generated to represent the image consisting of m number of objects. With the triple (O_i, O_j, θ_{ij}), the corresponding key K is computed as

$$K = D_\theta(O_i - 1)m + D_\theta(O_j - 1) + (C_\theta - 1),$$

where D_θ is the number of slices the continuous interval type domain $[0 \ldots 359^\circ]$ associated with θ_{ij} is split into, and C_θ is the slice number to which a specific value of θ_{ij} belongs.

For an image S with m objects, there are $O(m^2)$ key values to represent this image. In order to reduce to storage requirement, the mean μ and the standard deviation σ of the set of triples are computed, and then to store the vector (N, μ, σ) as the representative vector of S, where N is the number of triples generated according to S. Suppose there are n images in the database, the representative vectors are stored

in a sorted order. For exact match retrieval, the desired image can be retrieved in $O(\log n)$ search time, by employing the modified binary search technique [44].

For example, in Figure 2.33, the direction of reference obtained for the image is 0°. There are two candidates for the computation of direction of reference, one connecting the objects 1 and 3 with 45^circ direction and the other connecting the objects 1 and 4 with -45° direction. Upon the triangular law of direction, we have the direction of reference as 0°. In this example, the θ_{ij}-domain is split into 360 slices of size 1°. The set of triples preserving the pairwise spatial relationship among the objects with respect to the direction of reference is

$$P_t = \{(1, 2, 0), (1, 3, 21), (1, 4, 339), (2, 3, 35), (2, 4, 325), (3, 4, 270)\}$$

The the set of the corresponding keys is

$$P_k = \{360, 741, 1419, 2195, 2485, 4320\}$$

Finally, the vector $D = (6, 1965.00, 1437.3477)$ is stored as the representative of the image.

CHAPTER 3

The Unique Bit Pattern Matrix Strategy

This chapter is organized as follows. First, the rules of transformation of the spatial relationships among objects will be described. Next, the concept of the transpose of a matrix and propose a bit operation will be presented. Then, special bit patterns to represent the spatial relationships will be proposed. Finally, our proposed index structure and the relevant algorithms will be presented.

3.1 Rules of Image Rotation and Reflection

To deal with the change of spatial relationships between objects in rotation and reflection, Nabil *et al.* [61] introduced the condition function as shown in Figure 3.1. In Figure 3.1, $g(h)$ is a spatial relationship, and $gi(hi)$ is the inverse of $g(h)$. On the other hand, Petraglia *et al.* [63] presented a mapping table as shown in Table 3.1 to deal with the linear transformation. For example, the transformed operator of the spatial operator "<" is "<*". The operator "]*" is the transformed operator of the spatial operator "[*". Comparing the condition function with the mapping table, they obtain the same transformed operator.

Table 3.1 Transformed operators

operator	<	<*	|	|*	/	/*	]	[	%	=	]*	[*	%*
transformed operator	<*	<	|*	|	/*	/	[	]	%	=	[*	]*	%*

$$\xi(g) = \begin{cases} gi & \text{if } g \in \{<, |, /\} \\ g & \text{if } g \in \{\%, \%^*, =\} \\ h & \text{where } g = hi \text{ and } g \in \{<^*, |^*, /^*\} \\ k & \text{where } g \in \{[, [^*\} \text{ and } k \in \{],]^*\} \\ l & \text{where } g \in \{],]^*\} \text{ and } l \in \{[, [^*\} \end{cases}$$

Figure 3.1 The condition function for linear transformations

Table 3.2 Transformed operators divided into 3 cases

	Case 1			Case 2		Case 3		
operator	$<$	$\|$	$/$	$]$	$]^*$	$\%$	$=$	$\%^*$
transformed operator	$<^*$	$\|^*$	$/^*$	$[$	$[^*$	$\%$	$=$	$\%^*$

From Table 3.1, we observe that the transformed operators can be classified into three cases as shown in Table 3.2. In Case 1, the related transformed operator is the same as the related inverse operator. For example, in Figure 3.2, "$<^*$" is both of the transformed operator and the inverse operator of the spatial operator "$<$". In Case 2, the related transformed operator is different from the related inverse operator. For example, in Figure 3.3, the transformed operator of the spatial operator "]" is "[", but the inverse operator is "]*". In Case 3, the related transformed operators are themselves. For example, in Figure 3.4, the transformed operator of the spatial operator "%" is itself, *i.e.*, "%".

When we rotate an image in the clockwise direction as shown in Figure 3.5-(a), the spatial relationships between objects A and B in the x- and y-axes are mutually exchanged. Moreover, the beginning and the ending points of any object in the y-axis are mutually exchanged. Thus, the crossed arrows in Figure 3.5-(a) show the exchange of the spatial operators in the x- and y-axes. Moreover, the gray arrow shows the spatial operator, *i.e.*, "$A <^* B$", changes to the related transformed operator, *i.e.*, "$A < B$", in the rotated image. When we flip an image horizontally as shown in Figure 3.5-(b), the beginning and ending points of any object in the

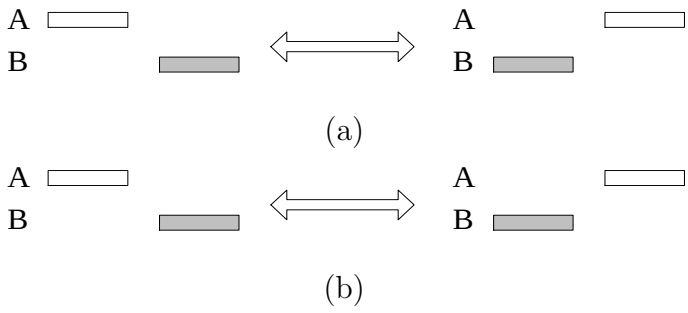


Figure 3.2 Example of Case 1: (a) $A < B$ *vs.* $A <^* B$ (transformed); (b) $A < B$ *vs.* $A <^* B$ (inverse).

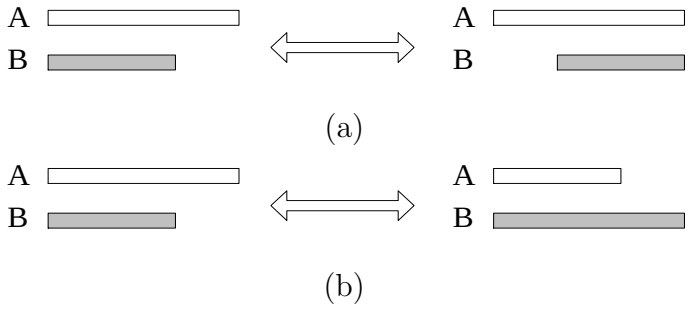


Figure 3.3 Example of Case 2: (a) $A[B$ *vs.* $A]B$ (transformed); (b) $A[B$ *vs* $A[^*B$ (inverse).

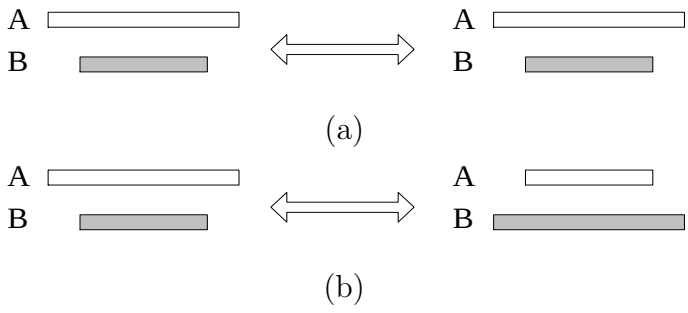
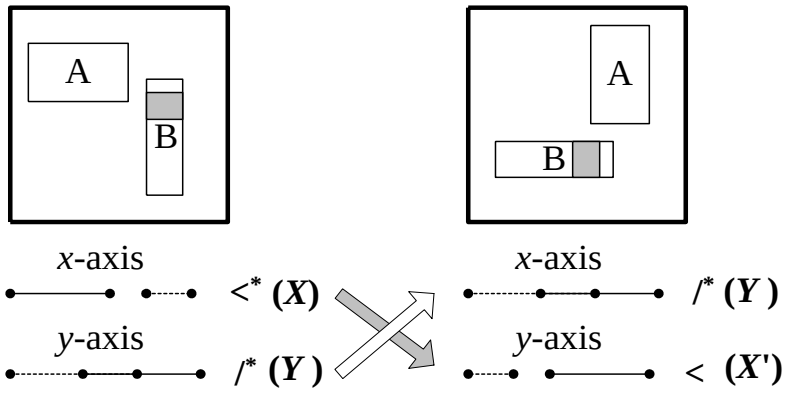
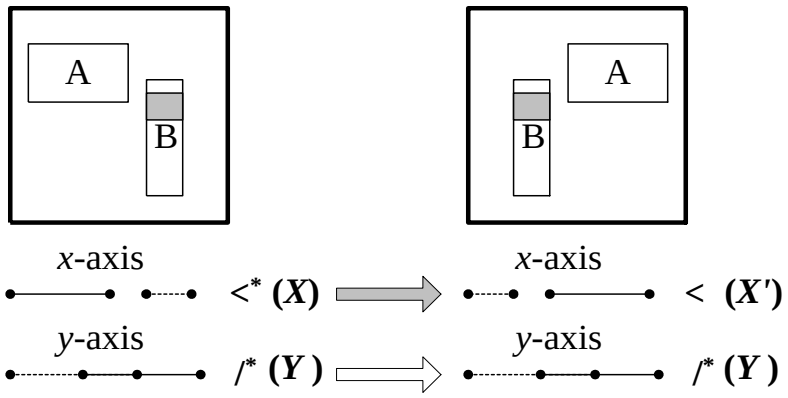


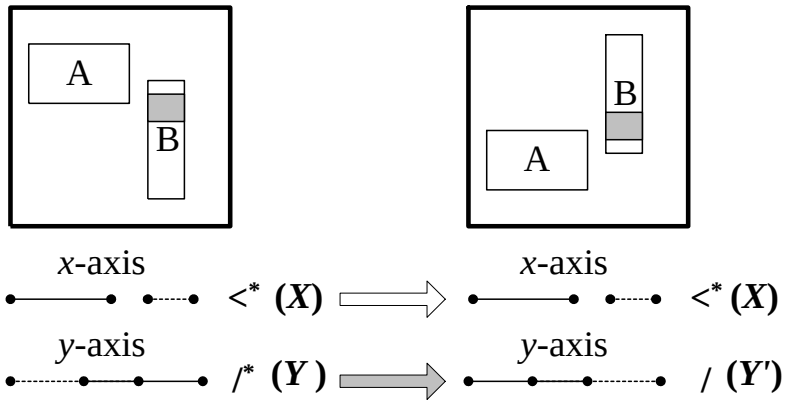
Figure 3.4 Example of Case 3: (a) $A\%B$ *vs.* $A\%B$ (transformed); (b) $A\%B$ *vs.* $A\%^*B$ (inverse).



(a)



(b)



(c)

X' (Y'): The transformed operator of X (Y)

Figure 3.5 Examples of rotation and reflection: (a) rotating 90° clockwise; (b) flipping horizontally; (c) flipping vertically.

Table 3.3 Rules of transformation: X' and Y' are the transformed operators related to X and Y, respectively.

	operator	
	x-axis	y-axis
functions	X	Y
Rotate 90°	Y	X'
Rotate 180°	X'	Y'
Rotate 270°	Y'	X
Flip horizontally	X'	Y
Flip vertically	X	Y'

x-axis are mutually exchanged. Thus, the gray arrow in Figure 3.5-(b) shows the spatial operator, *i.e.*, "$A <^* B$", changes to the related transformed operator, *i.e.*, "$A < B$", in the horizontally flipped image. When we flip an image vertically as shown in Figure 3.5-(c), the beginning and ending points of any object in the y-axis are mutually exchanged. Thus, the gray arrow in Figure 3.5-(c) shows the spatial operator, *i.e.*, "$A/^*B$", changes to the related transformed operator, *i.e.*, "A/B", in the vertically flipped image. According to the above observation, we present rules for transformation (rotation and reflection) as shown in Table 3.3. It is obvious that rotating an image by 180° and 270° clockwise are equivalent to rotating the image by 90° clockwise twice and three times, respectively. Thus, it is trivial to derive the rules of rotating images by 180° and 270° clockwise.

procedure *Transpose*(M) /* M is an $m \times m$ UBP matrix */

1: **for** $i := 1$ to m **do**

2: **for** $j := i + 1$ to m **do**

3: begin

4: $temp := M[i, j]$;

5: $M[i, j] := M[j, i]$;

6: $M[j, i] := temp$;

7: end

8: return (M);

end procedure

Figure 3.6 Procedure *Transpose*

3.2 The Matrix Manipulation and the Proposed Bit Operation

In Table 3.3, we observe that the spatial relationships in x- and y-axes will be mutually exchanged, when the image is rotated by 90° or 270° clockwise. Similar to the UID matrix strategy [23], we will use a matrix, called *unique bit pattern* matrix (UBP matrix), to record the spatial relationships among objects. The entry m_{ij} in a UBP matrix is the spatial relationship between objects i and j in x-axis when $i > j$. Otherwise, m_{ij} is the spatial relationship in y-axis when $i < j$. Because we use a matrix to be as the index structure, we introduce the concept of the *transpose* of a matrix to deal with the mutual exchange of the spatial relationships in x- and y-axes. Definition 1 describes the meaning of the transpose of a matrix. Figure 3.6 shows the procedure to obtain the transpose of a matrix.

Definition 1. *The matrix B is the* ***transpose*** *of the matrix A, written $B = A^T$, if each entry b_{ij} in B is the same as the entry a_{ji} in A, and conversely [33].*

Suppose M is a UBP matrix, and N is the transpose of the matrix M, *i.e.*, $N = M^T$. According to the definition of the transpose of a matrix, m_{ij} in M is the same as n_{ji} in N. m_{ij} is the spatial relationship between the objects i and j in x-axis when $i > j$. Then, n_{ji} which is the same as m_{ij} is to be the spatial relationship between objects i and j in y-axis. This means that the spatial relationships between the two objects in x-axis in M are to be the spatial relationships in y-axis in N. As the same result, the spatial relationships in y-axis in M are to be the spatial relationships in x-axis in N. In this way, the meaning of the transpose of a UBP matrix is to mutually exchange the spatial relationships in x- and y-axes.

When doing the rotation or flip operations, we need to change the spatial relationships to their related transformed spatial relationships by following the rules shown in Table 3.3. Thus, we propose a bit operation and use bit patterns to present those 13 spatial relationships to make the change more efficient.

There are several bit operations, *e.g.*, *bit shift*, *and*, *or*, and *exclusive or*, *etc.* We propose a bit operation, called *intra-exchange*, as shown in Figure 3.7. The bits B_i and B_{i+1} are mutually exchanged, where i is an even number. Figure 3.8 shows the procedure to do the *intra-exchange* operation. W_1 and W_2 are the bit patterns, where the odd and even bits are set to 1, respectively.

We use an example as shown in Figure 3.9 to describe the steps of Procedure *Intra_Exchange*. First, we shift left the bit string "0000 1001" by 1 to obtain T_{left}, *i.e.*, "0001 0010". Second, we shift right the bit string "0000 1001" by 1 to obtain T_{right}, *i.e.*, "0000 0100". To extract the values of the odd bits in T_{left}, we do the *AND* operation with T_{left} and W_1 to obtain T_{odd}, *i.e.*, "0000 0010". Similarly, to extract the values of the even bits in T_{right}, we do the *AND* operation with T_{right} and W_2 to obtain T_{even}, *i.e.*, "0000 0100". Finally, we do the *OR* operation with T_{even} and T_{odd} to obtain the intra-exchanged bit string, *i.e.*, "0000 0110".

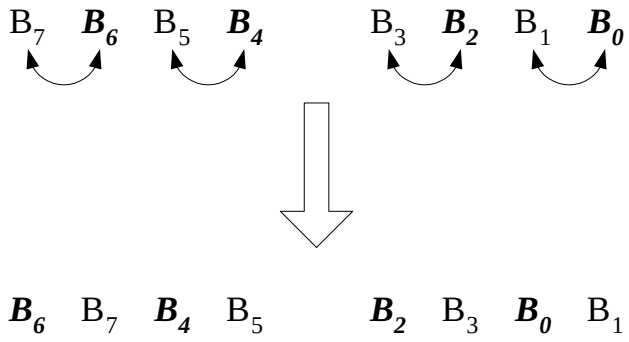


Figure 3.7 Intra-exchange of the odd and even bits

procedure *Intra_Exchange*(*bit_string*)

1: T_{left} := *bit_string* shift left by 1;

2: T_{right} := *bit_string* shift right by 1 ;

3: $T_{odd} := T_{left}$ AND W_1; /* $W_1 = 101010\ldots10$, where $|W_1| = |$*bit_string*$|$ */

4: $T_{even} := T_{right}$ AND W_2; /* $W_2 = 010101\ldots01$, where $W_2 = \overline{W_1}$ */

5: *result* := T_{even} OR T_{odd};

6: return (*result*);

end procedure

Figure 3.8 Procedure *Intra_Exchange*

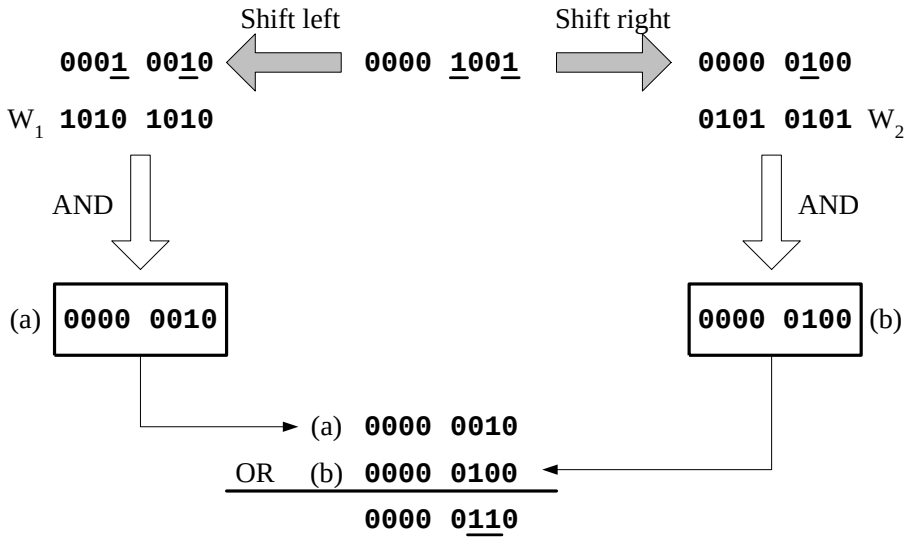


Figure 3.9 Example of applying Procedure *Intra_Exchange* to bit string 0000 1001

Table 3.4 New order of operators

	Case 1						Case 2		Case 3			Case 2	
operator	$<$	$<^*$	$\|$	$\|^*$	$/$	$/^*$	]	[	%	$=$	%*	]*	[*
order	1	2	3	4	5	6	7	8	9	10	**11**	**12**	**13**

3.3 Unique Bit Patterns

Chang *et al.* [23] assigned 13 different numbers, *i.e.*, from 1 to 13, to those 13 spatial operators as shown in Table 2.5. Those operators and their related transformed operators shown in Cases 1 and 2 in Table 3.1 are adjacent to each other. Although the related transformed operators of those three operators shown in Case 3 in Table 3.1 are themselves, the order of the last three operators shown in Table 2.5 are changed to make these three operators of Case 3, *i.e.*, "%", "=", "%*", be adjacent to each other. Table 3.4 shows the 13 spatial operators in the new order.

According to the order of the spatial operators shown in Table 3.4, We assign a unique 16-bit pattern, instead of a unique number, to each spatial operator as shown

in Table 3.5. The second column shows the spatial operators, and the third column shows their corresponding bit patterns. The last column shows the decimal numbers, when the bit patterns are viewed as binary numbers. There is only 1 bit set to 1 in the bit patterns of those operators in Case 1 and 2. However, the bit patterns of those operators in Case 3 are assigned with two of 1's. That is why it needs total $(10 \times 1 + 3 \times 2)$ bits to represent each spatial operator. In this assignment, the 1's in the bit patterns are adjacent to each other with the operator and the related transformed operator. For example, for the operator "$<$" and its related transformed operator "$<^*$", the bit which is set to 1 is in B_0 and B_1 bit, respectively, where B_0 is the right most bit of a bit pattern. In this way, by applying the *Intra_Exchange* procedure to the bit patterns of the spatial operators, the result is the bit pattern of the related transformed operators. For example, in Figure 3.10-(a), doing the *intra-exchange* operation on the bit pattern of the operator "$<$" results in the bit pattern of the transformed operator, "$<^*$". Figure 3.10-(b) shows the example of doing *intra-exchange* operation on the operators "]" and "[" which belong to Case 2. In Figure 3.10-(c), the transformed operator of "%" is itself, and doing the *intra-exchange* operation on the bit pattern of "%" also generates the bit pattern of itself.

3.4 Spatial Categories

According to the order of our proposed unique bit patterns, Those 169 spatial relationships in 2D space are arranged into a *Category* table as shown in Table 3.6. Those 169 spatial relationships are grouped together into 5 different categories, highlighted by the bold lines. For example, the spatial relationships in the first, second rows and the first, second columns are the disjoin spatial category. Thus, we can use a range checking algorithm as shown in Figure 3.11 to distinguish the spatial category between each two objects. The parameters ubp_x and ubp_y are the unique bit patterns among x- and y-axes, respectively. The corresponding decision tree is shown in Figure 3.12.

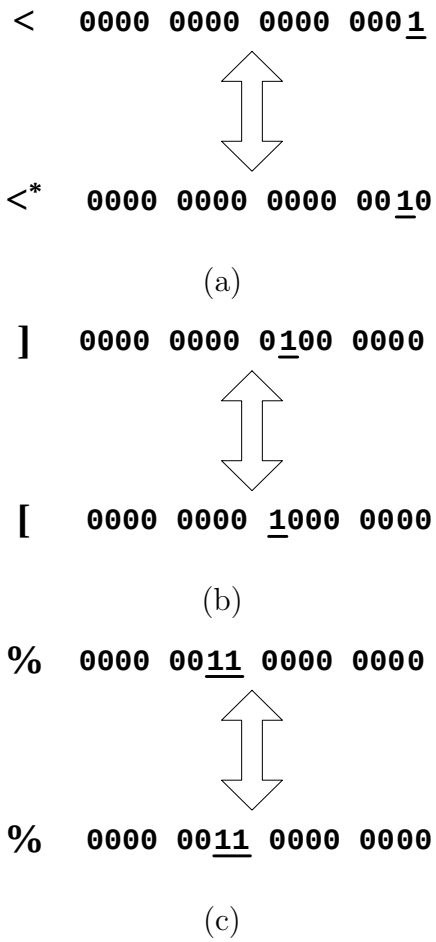


Figure 3.10 Three cases of bits intra-exchange: (a) Case 1; (b) Case 2; (c) Case 3.

procedure *Category*(ubp_x, ubp_y)

1: **if** ($ubp_x > 8$) and ($ubp_y > 8$) **then**

2: **if** ($64 \leq ubp_x \leq 3072$) and ($64 \leq ubp_y \leq 3072$) **then**

3: return ('contain')

4: **else if** ($3072 < ubp_x \leq 32768$) and ($3072 < ubp_y \leq 32768$) **then**

5: return ('belong')

6: **else** return ('partial overlapping')

7: **else if** ($ubp_x > 2$) and ($ubp_y > 2$)**then**

8: return ('join')

9: **else** return ('disjoin');

end procedure

Figure 3.11 Procedure *Category*

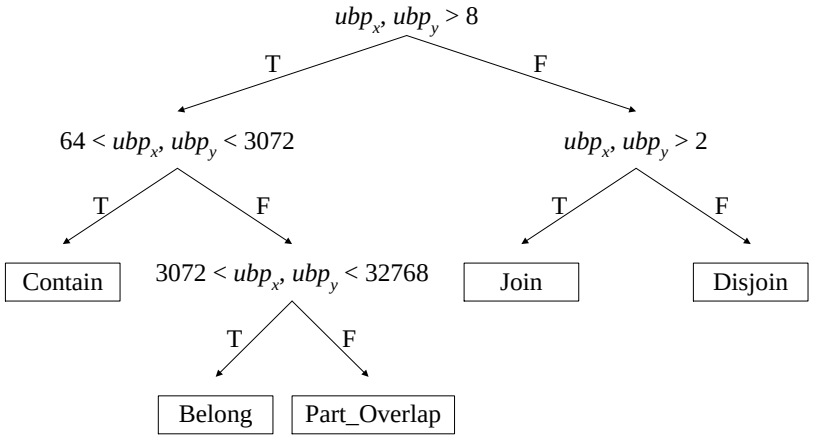


Figure 3.12 Decision tree of procedure *Category*

Table 3.5 Bit patterns of operators

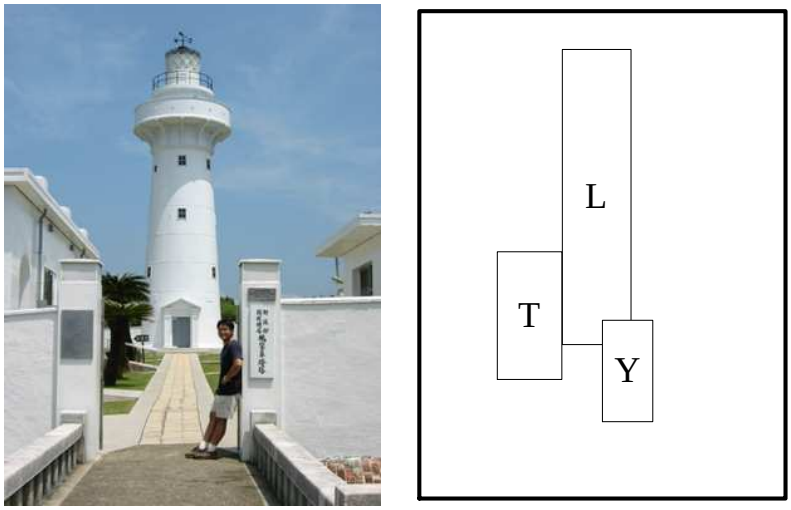
Case	Operator	Unique bit pattern	Decimal number
1	$<$	0000 0000 0000 000**1**	1
	$<^*$	0000 0000 0000 00**1**0	2
	$\|$	0000 0000 0000 0**1**00	4
	$\|^*$	0000 0000 0000 **1**000	8
	$/$	0000 0000 000**1** 0000	16
	$/^*$	0000 0000 00**1**0 0000	32
2	$]$	0000 0000 0**1**00 0000	64
	$[$	0000 0000 **1**000 0000	128
3	$\%$	0000 00**11** 0000 0000	768
	$=$	0000 **11**00 0000 0000	3072
	$\%^*$	00**11** 0000 0000 0000	12288
2	$]^*$	0**1**00 0000 0000 0000	16384
	$[^*$	**1**000 0000 0000 0000	32768

3.5 The Unique Bit Pattern Matrix

Similar to previous iconic index strategies [15], we assume that there are at least two objects in an image. The spatial information between any two objects can be derived. Thus, we propose a *unique-bit-pattern* matrix (UBP matrix) to preserve the spatial information of the objects in an image. Suppose an image I contains m objects and let $O = \{o_1, o_2, ..., o_m\}$. Let A be the set of 13 spatial operators $\{<, <^*, |, |^*, [, [^*,],]^*, \%, \%^*, /, /^*, =\}$. An $m \times m$ *spatial matrix* S [23] of the image I is defined as follows:

Table 3.6 Category table

$r^y_{A,B}$ \ $r^x_{A,B}$	1 `<`	2 `<*`	4 `|`	8 `|*`	16 `/`	32 `/*`	64 `]`	128 `[`	768 `%`	3072 `=`	12288 `%*`	16384 `]*`	32768 `[*`
1 `<`	`<`	`<*`	`|`	`|*`	`/`	`/*`	`]`	`[`	`%`	`=`	`%*`	`]*`	`[*`
2 `<*`	`<*`	`<*`	`|<*`	`|*<*`	`/<*`	`/*<*`	`]<*`	`[<*`	`%<*`	`=<*`	`%*<*`	`]*<*`	`[*<*`
4 `|`	`<|`	`<*|`	`||`	`|*|`	`/|`	`/*|`	`]|`	`[|`	`%|`	`=|`	`%*|`	`]*|`	`[*|`
8 `|*`	`<|*`	`<*|*`	`||*`	`|*|*`	`/|*`	`/*|*`	`]|*`	`[|*`	`%|*`	`=|*`	`%*|*`	`]*|*`	`[*|*`
16 `/`	`</`	`<*/`	`|/`	`|*/`	`//`	`/*/`	`]/`	`[/`	`%/`	`=/`	`%*/`	`]*/`	`[*/`
32 `/*`	`</*`	`<*/*`	`|/*`	`|*/*`	`//*`	`/*/*`	`]/*`	`[/*`	`%/*`	`=/*`	`%*/*`	`]*/*`	`[*/*`
64 `]`	`<]`	`<*]`	`|]`	`|*]`	`/]`	`/*]`	`]]`	`[]`	`%]`	`=]`	`%*]`	`]*]`	`[*]`
128 `[`	`<[`	`<*[`	`|[`	`|*[`	`/[`	`/*[`	`][`	`[[`	`%[`	`=[`	`%*[`	`]*[`	`[*[`
768 `%`	`<%`	`<*%`	`|%`	`|*%`	`/%`	`/*%`	`]%`	`[%`	`%%`	`=%`	`%*%`	`]*%`	`[*%`
3072 `=`	`<=`	`<*=`	`|=`	`|*=`	`/=`	`/*=`	`]=`	`[=`	`%=`	`==`	`%*=`	`]*=`	`[*=`
12288 `%*`	`<%*`	`<*%*`	`|%*`	`|*%*`	`/%*`	`/*%*`	`]%*`	`[%*`	`%%*`	`=%*`	`%*%*`	`]*%*`	`[*%*`
16384 `]*`	`<]*`	`<*]*`	`|]*`	`|*]*`	`/]*`	`/*]*`	`]]*`	`[]*`	`%]*`	`=]*`	`%*]*`	`]*]*`	`[*]*`
32768 `[*`	`<[*`	`<*[*`	`|[*`	`|*[*`	`/[*`	`/*[*`	`][*`	`[[*`	`%[*`	`=[*`	`%*[*`	`]*[*`	`[*[*`



(a) (b)

Figure 3.13 Example: (a) an image; (b) the symbolic representation.

$$S = \begin{array}{c} \\ o_1 \\ o_2 \\ \vdots \\ o_{m-1} \\ o_m \end{array} \begin{array}{c} \begin{array}{ccccc} o_1 & o_2 & \cdots & o_{m-1} & o_m \end{array} \\ \begin{bmatrix} 0 & r^y_{1,2} & \cdots & \cdots & r^y_{1,m} \\ r^x_{1,2} & 0 & \ddots & & \vdots \\ \vdots & \ddots & 0 & \ddots & \vdots \\ \vdots & & \ddots & 0 & r^y_{m-1,m} \\ r^x_{1,m} & \cdots & \cdots & r^x_{m-1,m} & 0 \end{bmatrix} \end{array}$$

where the lower triangular matrix stores the spatial information along the x-axis, and the upper triangular matrix stores the spatial information along the y-axis. That is, $S[o_i, o_j] = r^x_{j,i}$ if $i > j$; $S[o_i, o_j] = r^y_{i,j}$ if $i < j$; $S[o_i, o_j] = 0$ if $i = j$, $\forall o_i, o_j \in S$, $\forall r^x_{j,i}, r^y_{i,j} \in A$, $1 \leq i, j \leq m$, where $r^x_{j,i}$ and $r^y_{i,j}$ are the spatial operators between objects o_i and o_j along the x- and y-axes, respectively. In this representation, the relationships between two objects o_i and o_j are recorded from the viewpoint of object o_i either along the x-axis or along the y-axis, where $i < j$. Thus, $S[o_i, o_j] = r^x_{j,i}$ when $i > j$. For the image shown in Figure 3.13, the corresponding spatial matrix S is shown as follows:

$$S = \begin{array}{c} \\ L \\ T \\ Y \end{array} \begin{array}{c} \begin{array}{ccc} L & T & Y \end{array} \\ \begin{bmatrix} 0 & /^* & /^* \\ |^* & 0 & /^* \\ / & < & 0 \end{bmatrix} \end{array}$$

Objects L, T, and Y stand for the lighthouse, the tree, and the person, respectively. According to the assignments of the unique bit patterns for those 13 spatial operators shown in Table 3.5, we can transform the spatial matrix S of the image p into a *UBP matrix* M by replacing each spatial operator with its corresponding unique bit pattern (in decimal representation) as follows:

$$M = \begin{array}{c} \\ L \\ T \\ Y \end{array} \begin{array}{c} \begin{array}{ccc} L & T & Y \end{array} \\ \begin{bmatrix} 0 & 32 & 32 \\ 8 & 0 & 32 \\ 16 & 1 & 32 \end{bmatrix} \end{array}$$

Then, the UBP matrix M records the relative position of all objects. Moreover, the storage space of the matrix M is $3^2 \times 16$ bits. In other words, if an image I contains m objects, then we use an $m \times m$ UBP matrix with $m^2 \times 16$ bits of storage space to index the image I. Note that although the strategies recording spatial information in a matrix, *e.g.*, [6, 26], require more storage space than the strategies recording spatial information in strings, *e.g.*, [20, 56], the matrix-based strategies do similarity retrieval more efficiently than the string-based strategies [23].

3.6 Deriving Indices of The Rotated and Flipped Images

We propose algorithms to derive the UBP matrices of the rotated and flipped images from the UBP matrix of the original image. Table 3.7 shows the definition of the symbols, M^{90}, M^{180}, M^{270}, M^H, and M^V. In our proposed algorithms, we

will use those two procedures as shown in Figures 3.14 and 3.15. Procedures *Upper_Right_Triangular* and *Lower_Left_Triangular* do the *intra_exchange* operation on each entry in the upper right triangular and the lower left triangular area of a matrix, respectively. According to the rules of transformation shown in Table 3.3, we describe the algorithms to derive the matrices M^{90}, M^{180}, M^{270}, M^{H}, and M^{V} as follows.

1. Matrix M^{90}: When an image is rotated by 90°, first, the spatial relationships between any two objects along x- and y-axes are exchanged. The spatial information in the matrix M^{T} represents the exchange. Second, the original spatial relationships along the x-axis are changed to their related transformed spatial relationships. Because the upper right triangular matrix of M^{T} records the original spatial relationships along the x-axis, Procedure *Intra_Exchange* is applied to each entry in the *upper right triangular* matrix of M^{T} to change the spatial information to the related transformed spatial information. The details are described in Procedure *M_90* as shown in Figure 3.16.

2. Matrix M^{180}: When an image is rotated by 180°, the spatial relationships between any two objects along x- and y-axes are changed to their related transformed spatial relationships. Thus, Procedure *Intra_Exchange* is applied to each entry of M to change the spatial information to the related transformed spatial information. The details are described in Procedure *M_180* shown in Figure 3.17.

3. Matrix M^{270}: When an image is rotated by 270°, first, the spatial relationships between any two objects along x- and y-axes are exchanged. The spatial information in the matrix M^{T} represents the exchange. Second, the original spatial relationships along the y-axis are changed to their related transformed spatial relationships. Because the lower triangular matrix of M^{T} records the original spatial relationships along the y-axis, Procedure *Intra_Exchange* is applied to each entry in the *the lower left triangular* matrix of M^{T} to change the spatial information to the related transformed spatial information. The details are described in Procedure *M_270* shown in Figure 3.18.

Table 3.7 The definition of the matrix symbols

Symbol	Definition
I	a database image
M	the UBP matrix of I
M^T	the transpose of the matrix M
M^{90}	The UBP matrix of I rotated by 90° clockwise
M^{180}	The UBP matrix of I rotated by 180° clockwise
M^{270}	The UBP matrix of I rotated by 270° clockwise
M^H	The UBP matrix of I flipped horizontally
M^V	The UBP matrix of I flipped vertically

4. Matrix M^H: When an image is flipped horizontally, the spatial relationships along the x-axis are changed to their related transformed spatial relationships. Because the lower left triangular matrix of M records the spatial relationships along the x-axis, Procedure *Intra_Exchange* is applied to each entry in the *lower left triangular* matrix of M to change the spatial information to the related transformed spatial information. The details are described in Procedure *M_H* shown in Figure 3.19.

5. Matrix M^V: When an image is flipped vertically, the spatial relationships along the y-axis are changed to their related transformed spatial relationships. Because the upper right triangular matrix of M records the spatial relationships along the y-axis, Procedure *Intra_Exchange* is applied to each entry in the *upper right triangular* matrix of M to change the spatial information to the related transformed spatial information. The details are described in Procedure *M_V* shown in Figure 3.20.

We use five examples to show how the above matrix-deriving algorithms work. In Figure 3.21, the UBP matrix M is the index of the image on the top of Figure 3.21. Then, the rotated image by 90° clockwise is shown on the bottom of Figure 3.21.

procedure *Upper_Right_Triangular*(M) /* M is an $m \times m$ UBP matrix */

1: **for** $i := 1$ to m **do**

2: **for** $j := i + 1$ to m **do**

3: $M[i, j] :=$ *Intra_Exchange*($M[i, j]$);

4: return (M);

end procedure

Figure 3.14 Procedure *Upper_Right_Triangular*

procedure *Lower_Left_Triangular*(M) /* M is an $m \times m$ UBP matrix */

1: **for** $i := 1$ to m **do**

2: **for** $j := i + 1$ to m **do**

3: $M[j, i] :=$ *Intra_Exchange*($M[j, i]$);

4: return (M);

end procedure

Figure 3.15 Procedure *Lower_Left_Triangular*

procedure *M_90*(M) /* M is a UBP matrix */

1: $M^T :=$ *Transpose*(M);

2: $M^{90} :=$ *Upper_Right_Triangular*(M^T);

3: return (M^{90});

end procedure

Figure 3.16 Procedure *M_90*

procedure *M_180*(M) /* M is a UBP matrix */

1: $M^{180} :=$ *Upper_Right_Triangular*(M);

2: $M^{180} :=$ *Lower_Left_Triangular*(M^{180});

3: return (M^{180});

end procedure

Figure 3.17 Procedure *M_180*

procedure *M_270*(M) /* M is a UBP matrix */

1: $M^T :=$ *Transpose*(M);

2: $M^{270} :=$ *Lower_Left_Triangular*(M^T);

3: return (M^{270});

end procedure

Figure 3.18 Procedure *M_270*

procedure *M_H*(M) /* M is a UBP matrix */

1: $M^H :=$ *Lower_Left_Triangular*(M);

2: return (M^H);

end procedure

Figure 3.19 Procedure *M_H*

procedure $M\_V(M)$ /* M is a UBP matrix */

1: $M^V := Upper\_Right\_Triangular(M)$;

2: return (M^V);

end procedure

Figure 3.20 Procedure M_V

According to Table 3.3, the spatial relationships along x- and y-axes are mutually exchanged in the rotated image. Moreover, the spatial relationships along the y-axis of the rotated image are the transformed spatial relationships along the x-axis of the original image. Thus, first, the transpose of the matrix, M^T, is obtained to exchange the spatial relationships along the x- and y-axes. Then, procedure *Intra_Exchange* is employed to the upper right triangular area of the matrix M^T to change the original spatial relationships to their corresponding transformed spatial relationships. Finally, the index of the rotated image is generated, *i.e.*, the matrix M^{90}.

In Figure 3.22, the UBP matrix M is the index of the image on the top of Figure 3.22. Then, the rotated image by 180° clockwise is shown on the bottom of Figure 3.22. According to Table 3.3, the spatial relationships along x- and y-axes are changed to the related transformed spatial relationships, respectively. Thus, procedure *Intra_Exchange* is employed to the upper right triangular and lower left triangular area of the matrix M to change the original spatial relationships along the x and y-axes to their corresponding transformed spatial relationships. Finally, the index of the $180^c irc$-rotated image is generated, *i.e.*, the matrix M^{180}.

In Figure 3.23 to describe the steps of deriving the matrix M^{270}. The UBP matrix M is the index of the image on the top of Figure 3.23. Then, the rotated image by 270° clockwise is shown on the bottom of Figure 3.23. According to Table 3.3, the spatial relationships along x- and y-axes are mutually exchanged in the 270°-rotated image. Moreover, the spatial relationships along the x-axis of the 270°-rotated image are the transformed spatial relationships along the y-axis of the original image. Thus, first,

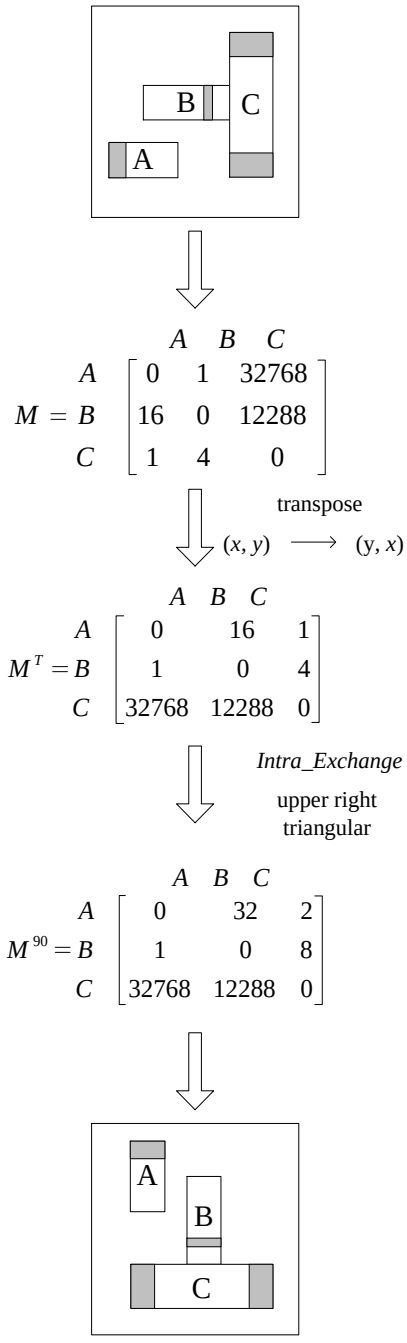


Figure 3.21 Process of deriving the matrix of rotated image by 90°

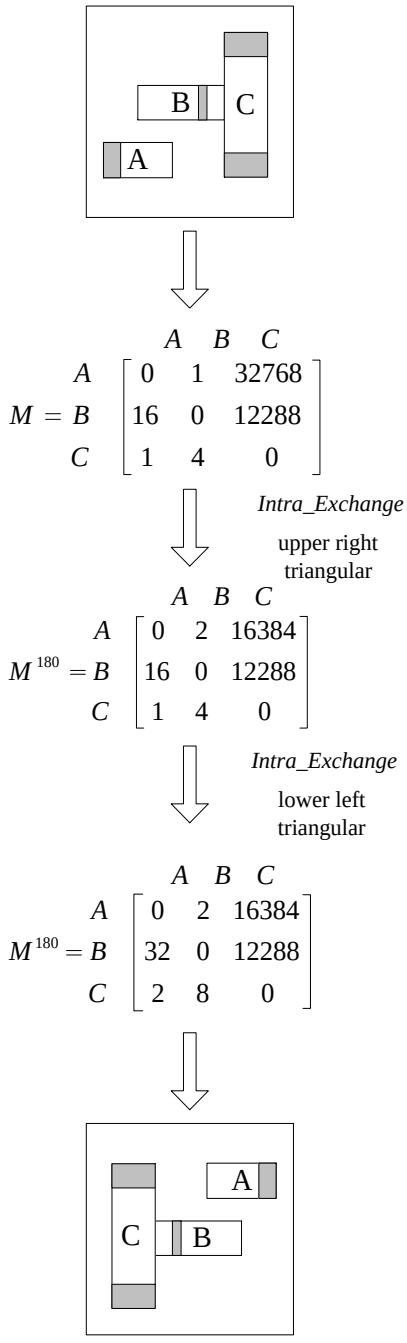


Figure 3.22 Process of deriving the matrix of rotated image by 180°

the transpose of the matrix, M^T, is obtained to exchange the spatial relationships along the x- and y-axes. Then, procedure *Intra_Exchange* is employed to the lower left triangular area of the matrix M^T to change the original spatial relationships to their corresponding transformed spatial relationships. Finally, the index of the 270°-rotated image is generated, *i.e.*, the matrix M^{270}.

In Figure 3.24, the UBP matrix M is the index of the image on the top of Figure 3.24. Then, the horizontal-flipped image is shown on the bottom of Figure 3.24. According to Table 3.3, the spatial relationships along x-axis are changed to the related transformed spatial relationships. Thus, procedure *Intra_Exchange* is employed to the lower left triangular area of the matrix M to change the original spatial relationships along the x-axis to their corresponding transformed spatial relationships. Finally, the index of the horizontal-flipped image is generated, *i.e.*, the matrix M^H.

In Figure 3.25, the UBP matrix M is the index of the image on the top of Figure 3.25. Then, the horizontal-flipped image is shown on the bottom of Figure 3.25. According to Table 3.3, the spatial relationships along y-axis are changed to the related transformed spatial relationships. Thus, procedure *Intra_Exchange* is employed to the upper right triangular area of the matrix M to change the original spatial relationships along the y-axis to their corresponding transformed spatial relationships. Finally, the index of the vertival-flipped image is generated, *i.e.*, the matrix M^V.

3.7 Similarity Retrieval

Similar retrieval is to retrieve the images that are similar to the query image. In this Section, we will describe the similar types and the corresponding similarity retrieval algorithm based on our UBP Matrix strategy. We apply the same definition of the similarity measures [56], which is described in Definition 2, to determine the similarity degrees among images.

Definition 2. *Picture f' is a type-i unit picture of f, if*

1. *all objects in f' are also in f,*

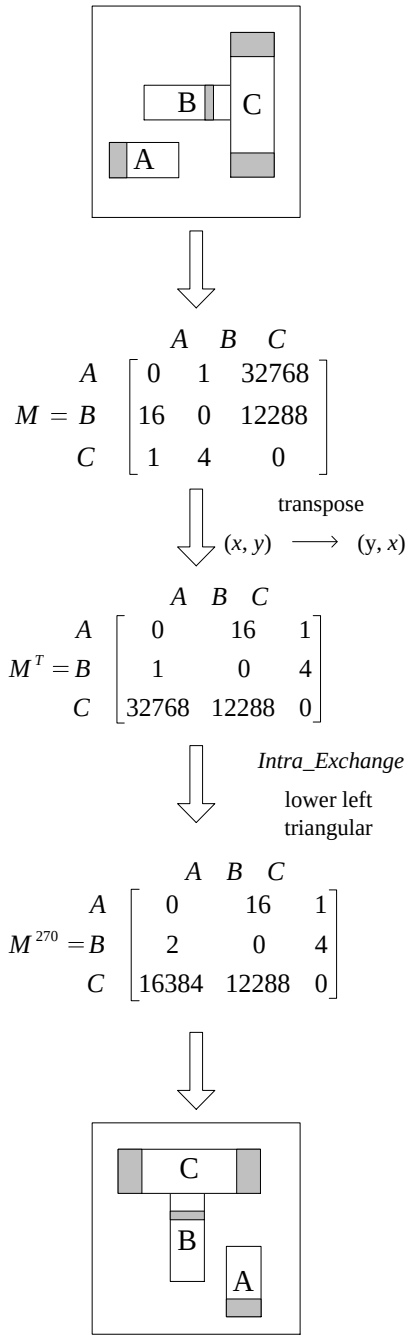


Figure 3.23 Process of deriving the matrix of rotated image by 270°

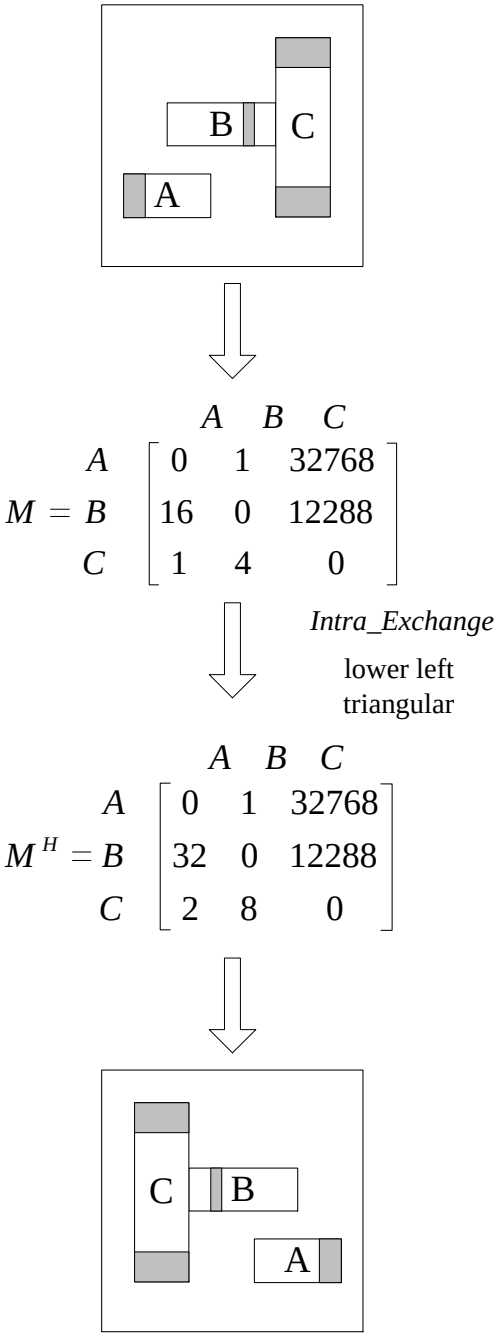


Figure 3.24 Process of deriving the matrix of horizontal-flipped image

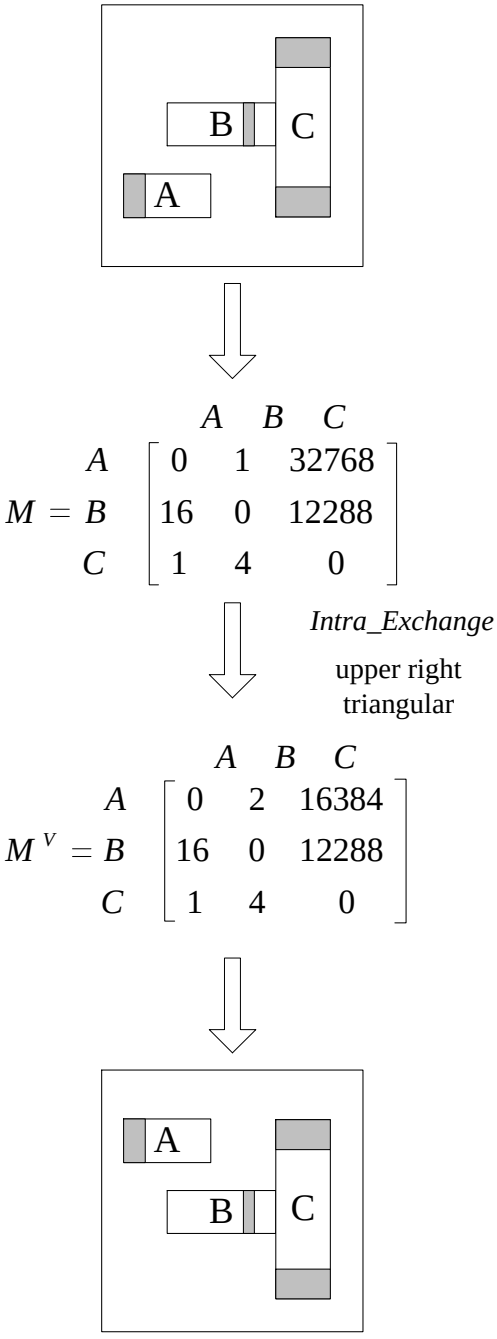


Figure 3.25 Process of deriving the matrix of vertical-flipped image

2. *for any two objects A and B, the spatial bit patterns among x- and y-axes between the objects A and B in f and f′ are represented as* (ubp_x, ubp_y) *and* ($ubp_{x'}$, $ubp_{y'}$), *respectively, then*

   type-0: *Category*(ubp_x, ubp_y) = *Category*($ubp_{x'}$, $ubp_{y'}$);

   type-1: *(type-0) and* ($ubp_x = ubp_{x'}$ *or* $ubp_y = ubp_{y'}$);

   type-2: $ubp_x = ubp_{x'}$ *and* $ubp_y = ubp_{y'}$.

where $Category(ubp_x, ubp_y)$ denotes the spatial category.

Then, some definitions which will be employed in our similarity retrieval algorithm are given as follows.

Definition 3. *Given a* $m \times m$ *matrix* M, *a matrix operator* P *is defined as follows:*
Let $M^* = (M)^P$, *where* $M^P(i,j) = M(i,j) * M(j,i) \quad ,\forall \;\; 1 \le i \le m, 1 \le j < i$.

Definition 4. *Given two* $m \times m$ *matrices* M_1 *and* M_2, *a matrix operator* $\otimes$ *is defined as follows:*

$$\text{Let } M = M_1 \otimes M_2, \text{ where } \begin{cases} M(i,j) = 0 & ,\forall \;\; M_1(i,j) = M_2(i,j) \\ M(i,j) = 1 & ,\forall \;\; M_1(i,j) \neq M_2(i,j) \end{cases}$$

Definition 5. *Given a* $m \times m$ *UBP Matrix* M, *the corresponding spatial category matrix* C *is defined as follows.*

$$\begin{cases} C[i,j] = 1, & if\ Category(M[i,j], M[j,i]) = \text{“Disjoin”}, & where\ 1 \le i < j \le m; \\ C[i,j] = 2, & if\ Category(M[i,j], M[j,i]) = \text{“Join”}, & where\ 1 \le i < j \le m; \\ C[i,j] = 3, & if\ Category(M[i,j], M[j,i]) = \text{“Contain”}, & where\ 1 \le i < j \le m; \\ C[i,j] = 4, & if\ Category(M[i,j], M[j,i]) = \text{“Belong”}, & where\ 1 \le i < j \le m; \\ C[i,j] = 5, & if\ Category(M[i,j], M[j,i]) = \text{“Part\_Overlap”}, & where\ 1 \le i < j \le m; \end{cases}$$

That is, $C[i,j] = 1, 2, 3, 4, 5$ if the relationship between objects o_i and o_j is of the disjoin, join, contain, belong and part_overlap category, respectively, by calling procedure *Category*.

Based on the two new matrix operators, P and $\otimes$, the following three algorithms, *type-0*, *type-1*, *type-2* are used to determine whether two images are of type-0, type-1, type-2 similarity, respectively, given two UBP Matrices M_1 and M_2.

Algorithm (type-0)

1. Following the procedure Category, find the category matrix C_1 and C_2 corresponding to the two matrices M_1 and M_2, respectively.
2. $C = C_1 \otimes C_2$. If C is zero in the lower triangular matrix, these two images are of type-0 similarity; otherwise, there is no match.

Algorithm (type-1)

1. Algorithm (type-0) passed.
2. $M = M_1 \otimes M_2$.
3. $M^* = (M)^P$.
   If M^* is zero in the lower triangular matrix, these two images are of type-1 similarity; otherwise, there is no match.

Algorithm (type-2)

1. $M = M_1 \otimes M_2$. If M is a zero matrix, these two pictures are of type-2 similarity; otherwise, there is no match.

For example, to find an image where there is a tree beside a lighthouse (type-0 similarity), the query may be issued by the symbolic representation as shown in Figure 3.26. The corresponding UBP matrix Q generated by the system is as follows:

$$Q = \begin{array}{c} \\ L \\ T \end{array} \begin{array}{c} \begin{array}{cc} L & T \end{array} \\ \begin{bmatrix} 0 & 32768 \\ 8 & 0 \end{bmatrix} \end{array}$$

Following procedure *Category* as shown in Figure 3.11, the spatial category between the tree and the lighthouse in the matrix Q is $Category(8, 32768)$, *i.e.*, *join*.

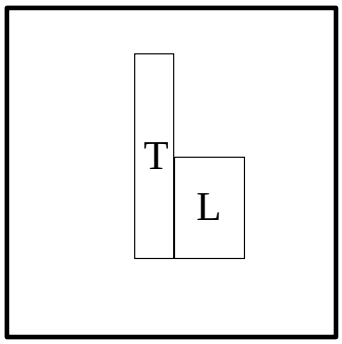


Figure 3.26 Symbolic representation of a query

Then, the spatial category between the two objects in the corresponding matrix M of the image shown in Figure 3.13 is $Category(8, 32)$, *i.e.*, *join*. Thus, the image shown in Figure 3.13 qualifies the query in type-0 similarity based on the Definition 2.

Now, we take one more complex example to show how the type-0, type-1, and type-2 algorithms work. Consider the images as shown in Figure 3.27.

(Step 1) Find the spatial matrices S_1 and S_2 and the UBP matrices M_1 and M_2 representing the two images I_1 and I_2, respectively.

$$S_1 = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & /^* & <^* & /^* \\ <^* & 0 & /^* & \% \\ \%^* & < & 0 & < \\ <^* & \%^* & <^* & 0 \end{bmatrix} \end{array}$$

$$M_1 = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & 32 & 2 & 32 \\ 2 & 0 & 32 & 768 \\ 12288 & 1 & 0 & 1 \\ 2 & 12288 & 2 & 0 \end{bmatrix} \end{array}$$

$$S_2 = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & \%^* & <^* & \% \\ <^* & 0 & \% & \% \\ \%^* & < & 0 & < \\ <^* & / & <^* & 0 \end{bmatrix} \end{array}$$

$$M_2 = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & 12288 & 2 & 768 \\ 2 & 0 & 768 & 768 \\ 12288 & 1 & 0 & 1 \\ 2 & 16 & 2 & 0 \end{bmatrix} \end{array}$$

(Step 2) Following the category rules, compute the corresponding category matrices, where 1, 2, 3, 4 and 5 mean the *disjoin*, *join*, *contain*, *belong* and *part_overlap* relationship, respectively.

$$C_1 = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} & & & \\ 1 & & & \\ 1 & 1 & & \\ 1 & 5 & 1 & \end{bmatrix} \end{array} \qquad C_2 = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} & & & \\ 1 & & & \\ 1 & 1 & & \\ 1 & 5 & 1 & \end{bmatrix} \end{array}$$

(Step 3) Check type-0 similarity. Since $C = 0$ in the lower triangular matrix, these two pictures are of type-0 similarity.

$$C = C_1 \otimes C_2 = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} & & & \\ 0 & & & \\ 0 & 0 & & \\ 0 & 0 & 0 & \end{bmatrix} \end{array}$$

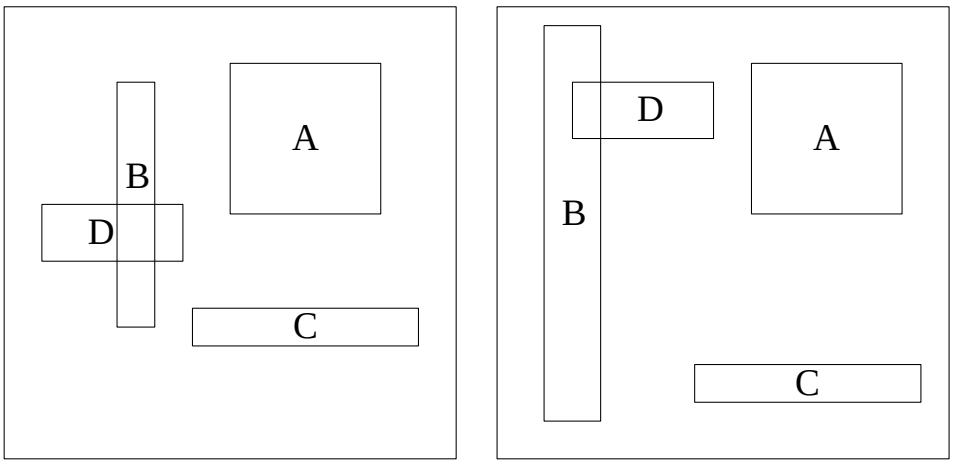
(Step 4) Check type-1 similarity. Since $T^* = 0$ in the lower triangular matrix, these two pictures are of type-1 similarity.

$$M = M_1 \otimes M_2 = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

$$M^* = (M)^R = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} & & & \\ 0 & & & \\ 0 & 0 & & \\ 0 & 0 & 0 & \end{bmatrix} \end{array}$$

(Step 5) Check type-2 similarity. Since $M \neq 0$, these two pictures are not of type-2 similarity.

$$M = M_1 \otimes M_2 \neq \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$



(a) (b)

Figure 3.27 Example of Images: (a) I_1; (b) I_2.

CHAPTER 4

Performance Study

In this chapter, first, we will analyze the time complexity of our proposed strategy. Then, we show the simulation results and discuss the properties of the results. In this simulation, we consider the CPU-time and the average search time as our performance measures. The CPU-time is the time to generate the corresponding index of the rotated or flipped image. The average search time is the time to find all qualified database for one query. Finally, we present a system prototype based on our proposed strategy.

4.1 Analysis

First, we analyze the complexity of generating the matrices, M^{90}, M^{180}, M^{270}, M^H, and M^V. Then, we analyze the complexity of searching a database image, when the query is issued in the orientation different from that of the database image.

There are two basic operations to generate the above matrices. One is procedure *Intra_Exchange*, the other is to obtain the transpose matrix, M^T. In procedure *Intra_Exchange*, from step 1 to step 5, each step is a CPU bit operation, the complexity is $O(1)$. Thus, the complexity of procedure *Intra_Exchange* is $5 \times O(1) = O(1)$. Suppose M is an $m \times m$ matrix, there are m^2 entries. To get the transpose matrix M^T, we go through a *loop*, as shown in Figure 3.6, to exchange the content of the entries, m_{ij} and m_{ji}. The complexity of the exchange is $O(1)$. The number of the exchange is $m^2/2$. Thus, the complexity of obtaining the matrix M^T is $m^2/2 \times O(1) = O(m^2)$.

According to the algorithms to obtain the matrices M^{90}, M^{180}, M^{270}, M^H, and M^V, we list the complexity as follows.

1. M^{90}: First, to obtain the transpose matrix M^T, the complexity is $O(m^2)$. Then, the complexity of employing procedure *Intra_Exchange* to each entry in the upper right triangular matrix of M^T is $m^2 \times O(1) = O(m^2)$. Finally, the total complexity is $O(m^2) + O(m^2) = O(m^2)$

2. M^{180}: The complexity of employing procedure *Intra_Exchange* to each entry of the matrix M is $m^2 \times O(1) = O(m^2)$.

3. M^{270}: First, to obtain the transpose matrix M^T, the complexity is $O(m^2)$. Then, the complexity of employing procedure *Intra_Exchange* to each entry in the lower left triangular matrix of M^T is $m^2 \times O(1) = O(m^2)$. Finally, the total complexity is $O(m^2) + O(m^2) = O(m^2)$

4. M^H: The complexity of employing procedure *Intra_Exchange* to each entry in the lower left triangular matrix of M is $m^2/2 \times O(1) = O(m^2)$.

5. M^V: The complexity of employing procedure *Intra_Exchange* to each entry in the upper right triangular matrix of M is $m^2/2 \times O(1) = O(m^2)$.

Chang *et al.* [23] have proved that to answer a query of similarity retrieval, the complexity of matching the index of the query with the index of a database image is $O(m^2)$, where m is the number of objects common to the database image and the query. Similarly, if the query is issued in the different orientation, in order not to miss the qualified database images, we need to compare another five indexes, *i.e.*, the matrices M^{90}, M^{180}, M^{270}, M^H, and M^V, with the index of an image. Thus, the complexity is $O(m^2) + 5 \times O(m^2) = O(m^2)$.

4.2 Simulation Results

We set the number of different objects appearing in the database be 100. For each object, the width and height of which are bounded between 1 and 100,000 units.

Table 4.1 The percentage of the improvement of our proposed algorithm as compared to the condition function

	10 objects	20 objects	30 objects	60 objects	90 objects
rotated by 180°	23.56%	36.45%	44.21%	47.29%	53.23%
flipped horizontally	13.64%	22.45%	28.82%	31.83%	36.72%

We generate images with 10, 20, 30, 60, and 90 objects which are randomly chosen from those 100 different objects. In each case, 100,000 images are generated. Thus, we use 500,000 images to calculate the average CPU-time of obtaining the index of the rotated and flipped images. Nabil *et al.* [61] and Petraglia *et al.* [63] used a condition function as shown in Figure 3.1 to deal with the linear transformation. This function does not show a straightforward way to change the spatial relationship to its related transformed spatial relationship. When we implement this function, we need to use conditional statements, such as *if-then-else*, to deal with the transformation. Thus, when the number of spatial relationships which need to be changed to its transformed one is huge, the process of this function is time-consuming. Instead of using the condition function, our strategy employs the *intra-exchange* bit operation to change the spatial relationship directly. In the following five cases, *Rotate in* 90°, *Rotate in* 180°, *Rotate in* 270°, *Flip horizontally*, and *Flip vertically*, we will compare the average CPU-time with different number of objects in an image. We show the two cases of the improvement of our proposed algorithm in Table 4.1. The more the number of objects in an image, the better improvement of our proposed algorithm is. Our proposed algorithm can reach more than 50% improvement as compared to those strategies based on the condition function, when there are 90 objects in an image.

The average search time is the time to search all the qualified images for one query. To evaluate the average search time, we prepare 100,000 images for each of the following cases in the database. We consider cases of 10, 20, 30, 60, and 90 different objects randomly chosen with the uniform distribution to appear in each image, respectively. There are 1,000 query images which contains 2 different objects. Each

Table 4.2 The average search time (in seconds) of different combinations

	10%	20%	50%
10 objects	6.80	6.85	6.85
20 objects	17.53	18.34	18.30
30 objects	32.80	32.62	32.76
60 objects	105.63	103.28	104.80
90 objects	219.45	217.08	219.50

query is a sub-image randomly chosen from the database images. Each query image with 10%, 20%, and 50% probability to be given in the rotation ordination or the reflection direction. Then, we compare the average search time of each combination.

Table 4.2 shows the simulation results. For the same number of objects in a database image, we observe that no matter what the value of the probability is, the average search time is almost the same. This is because we have to check the five cases to prevent from missing the qualified database images. We also observe that the more the number of objects in a database image, the longer the search time is. We have shown that the time complexity to do the similarity is $O(m^2)$, where m is the number of objects common to the query and the database image. Thus, the query time should be the same, when m is 2 in this simulation. However, before comparing the UBP matrix of the query with that of a database image, we need to obtain the UBP matrices of the rotated and flipped versions of the original image. Then, we have to obtain the sub-UBP matrix of the database image with the same objects information as the query. This is the reason why the more the number of objects in a database image, the longer the search time is. Note that, in fact, in the image database searching, the signature strategy [77] will be applied to prune off many unsatisfied images before comparing the indices of the candidate images with the index of the query one by one. Thus, the search time will be much less than that shown in Table 4.2.

CHAPTER 5

Conclusion

In this dissertation, we proposed an iconic indexing strategy for similarity retrieval in image database. Our proposed strategy can find out those qualified images in the database, even though the query is issued in a different spatial orientation of the qualified images. In this chapter, we give a summary of the dissertation and point out some future directions.

5.1 Summary

An image database system is concerned with the problem of storage, retrieval, and manipulation of pictorial data in an efficient manner. There are two major categories of features: *primitive* and *logical*. Logical features are abstract representations of images at various levels of detail, which contains the spatial relationship features. Spatial relationships are important ingredients for expressing constrains in retrieval systems for the image database. Thus, the concept of the iconic index was introduced and many iconic indexing strategies for iconic image database have been proposed.

In Chapter 2, we have given a survey of previous proposed representation strategies for symbolic pictures, including *2D strings* [20], *2D C-strings* [54], *2D B-strings* [58], *2D-PIR* [61], Zhou and Ang's strategy [80], *UID matrix* [23], *Virtual images* [63], *2D Z-string* [53], *9D-SPA* [49] *9DLT matrix* [6], *Triangular spatial relationship* (TSR) [41]

and it's subsequent extensions [43, 65, 66], respectively. These strategies could be classified into two different categories. One views objects with extents, the other views objects as points based on the objects' centroid.

In Chapter 3, we have presented an efficient iconic indexing strategy, *i.e.*, unique bit pattern matrix (UBP matrix), to derive the index of rotated and flipped images from the original index directly. In this way, our proposed strategy will not miss the qualified images when the query is issued in the different orientation as compared to the database images. By observing the order of the unique identifiers of the spatial relationships proposed in the UID matrix strategy, we have proposed a new order for those 13 spatial relationships to make the spatial relationship and its related transformed spatial relationship be adjacent to each other. Then, we have designed special 16-bit patterns to represent those spatial relationships. Based on these carefully designed bit patterns and our proposed bit operation, *i.e.*, *intra-exchange*, each bit pattern can be easily changed to another bit pattern of the related transformed spatial relationship. Although we reorder the spatial relationships, by viewing the bit patterns as digital numbers, we can distinguish the spatial category between each two objects by range checking algorithm.

In Chapter 4, we have shown that our proposed strategy makes a great improvement as compared to the strategies based on the condition function, when deriving the index of the rotated or flipped image from the original index.

5.2 Future Research Direction

In our proposed strategy, we can answer queries with at least two objects with their relative position. Future work includes dealing with the query with only one object and its relative position in the whole image. For example, finding all images which have a circle at the border.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] J. F. Allen, "Maintaining Knowledge about Temporal Intervals," *Comm. of the ACM*, Vol. 26, No. 11, pp. 832–843, Nov. 1983.

[2] S. K. Bhatia and C. L. Sabharwal, "A Fast Implementation of a Perfect Hash Function for Picture Objects," *Pettern Recognition*, Vol. 27, No. 3, pp. 365–376, March 1994.

[3] A. Blaser, *Database Techniques for Pictorial Applications, Lecture Notes in Computer Science*, Vol. 81. Springer Verlag GmbH, 1979.

[4] H. Burkhardt and S. Siggelkow, *Nonlinear Model-Based Image Video Processsing and Analysis*, ch. Invariant Features for Discriminating Between Equivalence Classes. John Wiley and Sons, 2000.

[5] A. E. Cawkill, "The British Library's Picture Research Projects: Image, Word, and Retrieval," *Advanced Imaging*, Vol. 8, No. 10, pp. 38–40, Oct. 1993.

[6] C. C. Chang, "Spatial Match Retrieval of Symbolic Pictures," *Journal of Information Science and Engineering*, Vol. 7, No. 3, pp. 405–422, Sept. 1991.

[7] C. C. Chang and C. F. Lee, "Relative Coordinates Oriented Symbolic String for Spatial Relationship Retrieval," *Pattern Recognition*, Vol. 28, No. 4, pp. 563–570, April 1995.

[8] C. C. Chang and C. F. Lee, "A Spatial Match Retrieval Mechanism for Symbolic Pictures," *The Journal of Systems and Software*, Vol. 44, No. 1, pp. 73–83, Dec. 1998.

[9] C. C. Chang and S. Y. Lee, "Retrieval of Similar Pictures on Pictorial Databases," *Pattern Recognition*, Vol. 24, No. 7, pp. 675–680, July 1991.

[10] C. C. Chang and D. C. Lin, "A Spatial Data Representation: An Adaptive 2D-H String," *Pattern Recognition Letters*, Vol. 17, No. 2, pp. 175–185, Feb. 1996.

[11] N. S. Chang and K. S. Fu, "A Relational Database System for Images," *Technical Report TR-EE*, pp. 79–82, May 1797.

[12] N. S. Chang and K. S. Fu, "Query by Pictorial Example," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 6, No. 6, pp. 519–524, Nov. 1980.

[13] S. K. Chang and A. Hsu, "Image Information Systems: Where Do We Go from Here?," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 5, No. 5, pp. 431–442, Oct. 1992.

[14] S. K. Chang and E. Jungert, "Pictorial Data Management Based upon The Theory of Symbolic Projection," *Journal of Visual Language and Computing*, Vol. 2, No. 3, pp. 195–215, Sept. 1991.

[15] S. K. Chang and E. Jungert, *Symbolic Projection for Image Information Retrieval and Spatial Reasoning*. London, U.K.: Academic Press, April 1996.

[16] S. K. Chang, E. Jungert, and Y. Li, "Representation and Retrieval of Symbolic Pictures Using Generated 2D Strings," *Proc. of Visual Communications and Image Processing*, pp. 1360–1372, 1989.

[17] S. K. Chang, E. Jungert, and G. Tortora, *Intelligent Image Database Systems*. Singapore: World Scientific Press, July 1996.

[18] S. K. Chang and T. L. Kunii, "Pictorial Database Systems," *IEEE Computer Magazine*, Vol. 14, No. 11, pp. 13–21, Nov. 1981.

[19] S. K. Chang and Y. Li, "Representation of Multi-Resolution Symbolic and Binary Pictures Using 2D-H Strings," *Proc. IEEE Workshop on Languages for Automata*, pp. 190–195, 1988.

[20] S. K. Chang, Q. Y. Shi, and C. W. Yan, "Iconic Indexing by 2D Strings," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 9, No. 3, pp. 413–428, May 1987.

[21] S. K. Chang, C. W. Yan, D. C. Dimitroff, and T. Arndt, "An Intelligent Image Database System," *IEEE Trans. on Software Engineering*, Vol. 14, No. 5, pp. 681–688, May 1988.

[22] Y. I. Chang and H. Y. Ann, "A Note on Adaptive 2D-H Strings," *Pattern Recognition Letters*, Vol. 20, No. 1, pp. 15–20, Jan. 1999.

[23] Y. I. Chang, H. Y. Ann, and W. H. Yeh, "A Unique-ID-Based Matrix Strategy for Efficient Iconic Indexing of Symbolic Pictures," *Pattern Recognition*, Vol. 33, No. 8, pp. 1263–1276, Aug. 2000.

[24] Y. I. Chang and B. Y. Yang, "A Prime-Number-Based Matrix Strategy for Efficient Iconic Indexing of Symbolic Pictures," *Pattern Recognition*, Vol. 30, No. 10, pp. 1745–1757, Oct. 1997.

[25] Y. I. Chang, B. Y. Yang, and W. H. Yeh, "A Generalized Prime-Number-Based Matrix Strategy for Efficient Iconic Indexing of Symbolic Pictures," *Pattern Recognition Letters*, Vol. 22, No. 6, pp. 657–666, May 2001.

[26] Y. I. Chang, B. Y. Yang, and W. H. Yeh, "A Bit-Pattern-Based Matrix Strategy for Efficient Iconic Indexing of Symbolic Pictures," *Pattern Recognition Letters*, Vol. 24, No. 1–3, pp. 537–545, Jan. 2003.

[27] Y. Chen, J. Z. Wang, and R. Krovetz, "An Unsupervised Learning Approach to Content-Based Image Retrieval," *Proc. of the IEEE Int. Symp. on Signal Processing and Its Applications*, pp. 197–200, July 2003.

[28] J. Dowe, "Content-Based Retrieval in Multimedia Imaging," *Proc. SPIE Storage and Retrieval for Image and Video Database*, 1993.

[29] J. Eakins and M. Graham, "Content-Based Image Retrieval," tech. rep., University of Northumbria at Newcastle, 1999.

[30] M. J. Egenhofer, "Point-Set Topological Spatial Relations," *Int. Journal of Geographical Information Systems*, Vol. 5, No. 2, pp. 161–174, 1991.

[31] M. J. Egenhofer and R. D. Franzasa, "On The Equivalence of Topological Relations," *Journal of Geographic Information Systems*, Vol. 9, No. 2, pp. 133–152, 1995.

[32] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz, "Efficient and Effective Querying by Image Content," *Journal of Intelligent Information Systems*, Vol. 3, No. 3–4, pp. 231–262, July 1994.

[33] J. B. Fraleigh and R. A. Beauregard, *Linear Algebra*. Addison Wesley, third ed., 1995.

[34] B. Furht, S. W. Smoliar, and H. J. Zhang, *Video and Image Processing in Multimedia Systems*. Kluwer Academic, 1995.

[35] Y. Gong, H. J. Zhang, and T. C. Chua, "An Image Database System with Content Capturing and Fast Image Indexing Abilities," *Proc. IEEE Int. Conf. on Multimedia Computing and Sytems*, pp. 121–130, May 1994.

[36] V. N. Gudivada, "On Spatial Similarity Measures for Multimedia Applications," *Proc. of SPIE Storage and Retrieval for Still Images and Video Databases III*, pp. 363–372, 1995.

[37] V. N. Gudivada, "θR-String: A Geometry-Based Representation for Efficient and Effective Retrieval of Images by Spatial Similarity," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 10, No. 3, pp. 504–512, May 1998.

[38] V. N. Gudivada and G. S. Jung, "An Algorithm for Content-Based Retrieval in Multimedia Databases," *Proc. of the Int. Conf. on Multimedia Computing and Systems*, pp. 90–97, 1995.

[39] V. N. Gudivada and V. V. Raghavan, "Design and Evaluation of Algorithms for Image Retrievals by Spatial Similarity," *ACM Trans. on Information Systems*, Vol. 13, No. 2, pp. 115–144, April 1995.

[40] A. Gupta and R. Jain, "Visual Information Retrieval," *Comm. of the ACM*, Vol. 40, No. 5, pp. 70–79, 1997.

[41] D. S. Guru and P. Nagabhushan, "Triangular Spatial Relationship: A New Approach for Spatial Knowledge Representation," *Pattern Recognition Letters*, Vol. 22, No. 9, pp. 999–1006, July 2001.

[42] D. S. Guru and P. Punitha, "An Invariant Scheme for Exact Match Retrieval of Symbolic Images Based Upon Principal Component Analysis," *Pattern Recognition Letters*, Vol. 25, No. 1, pp. 73–86, Jan. 2004.

[43] D. S. Guru, P. Punitha, and P. Nagabhushan, "Archival And Retrieval of Symbolic Images: An Invariant Scheme Based on Triangular Spatial Relationship," *Pattern Recognition Letters*, Vol. 24, No. 14, pp. 2397–2408, Oct. 2003.

[44] D. S. Guru, H. J. Raghavendra, and M. G. Suraj, "An Adaptive Binary Search Based Sorting by Insertion: An Efficient and Simple Algorithm," *Statistics and Applications*, Vol. 2, pp. 85–96, 2000.

[45] T. Y. Hou, P. Lui, and M. Y. Chui, "A Content-Based Indexing Technique Using Relative Geometry Features," *Proc. of SPIE The Int. Society for Optical Engineering*, Vol. 1662, 1992.

[46] F. J. Hsu and S. Y. Lee, "Similarity Retrieval by 2D C-Trees Matching in Image Databases," *Journal of Visual Communication and Image Representation*, Vol. 9, No. 1, pp. 87–100, March 1998.

[47] F. J. Hsu, S. Y. Lee, and B. S. Lin, "2D C-Tree Spatial Representation for Iconic Image," *Journal of Visual Languages and Computing*, Vol. 10, No. 2, pp. 147–164, April 1999.

[48] P. W. Huang and Y. R. Jean, "Using 2D C^+-String as Spatial Knowledge Representation for Image Database Systems," *Pattern Recognition*, Vol. 27, No. 9, pp. 1249–1257, Sept. 1994.

[49] P. W. Huang and C. H. Lee, "Image Database Design Based on 9D-SPA Representation for Spatial Relations," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 16, No. 12, pp. 1486–1496, Dec. 2004.

[50] R. Jain, ed., *Proc. US NSF Workshop Visual Information Management Systems*, 1992.

[51] E. Jungert, "Extended Symbolic Projections as a Knowledge Structure for Spatial Reasoning," *Proc. of the 4th Int. Conf. on Pattern Recognition*, Vol. 301, pp. 343–351, 1988.

[52] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, "Rotaion Invariant Spherical Harmonic Representation of 3D Shape Descriptions," *Proc. of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pp. 156–164, 2003.

[53] A. J. T. Lee and H. P. Chiu, "2D Z-String: A New Spatial Knowledge Representation for Image Databases," *Pattern Recognition Letters*, Vol. 24, No. 16, pp. 3015–3026, Dec. 2003.

[54] S. Y. Lee and F. J. Hsu, "2D C-String: A New Spatial Knowledge Representation for Image Database Systems," *Pattern Recognition*, Vol. 23, No. 10, pp. 1077–1087, Oct. 1990.

[55] S. Y. Lee and F. J. Hsu, "Picture Algebra for Spatial Reasoning of Iconic Images Represented in 2D C-String," *Pattern Recognition Letters*, Vol. 12, No. 7, pp. 425–435, July 1991.

[56] S. Y. Lee and F. J. Hsu, "Spatial Reasoning and Similarity Retrieval of Images Using 2D C-String Knowledge Representation," *Pattern Recognition*, Vol. 25, No. 3, pp. 305–318, March 1992.

[57] S. Y. Lee, M. K. Shan, and W. P. Yang, "Similarity Retrieval of Iconic Image Database," *Pattern Recognition*, Vol. 22, No. 6, pp. 675–682, June 1989.

[58] S. Y. Lee, M. C. Yang, and J. W. Chen, "2D B-String: A Spatial Knowledge Representation for Image Database Systems," *Proc. of the Second Int. Computer Science Conf.*, pp. 609–615, 1992.

[59] Y. Liu, D. Zhang, G. Lu, and W. Y. Ma, "A Survey of Content-Based Image Retrieval with High-Level Semantics," *Pattern Recognition*, Vol. 40, No. 1, pp. 262–295, Jan. 2007.

[60] W. Y. Ma and B. Manjunath, "Netra: A Toolbox for Navigating Large Image Databases," *Proc. of the IEEE Int. Conf. on Image Processing*, pp. 568–571, 1997.

[61] M. Nabil, A. H. H. Ngu, and J. Shepherd, "Picture Similarity Retrieval Using the 2D Projection Interval Representation," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 8, No. 4, pp. 533–539, Aug. 1996.

[62] A. Pentland, R. W. Picard, and S. Scaroff, "Photobook: Content-Based Manipulation for Image Database," *Int. Journal of Computer Vision*, Vol. 18, No. 3, pp. 233–254, 1996.

[63] G. Petraglia, M. Sebillo, M. Tucci, and G. Tortora, "Virtual Images for Similarity Retrieval in Image Databases," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 13, No. 6, pp. 951–967, Nov./Dec. 2001.

[64] E. G. M. Petrakis, "Design and Evaluation of Spatial Similarity Approaches for Image Retrieval," *Image and Vision Computing*, Vol. 20, No. 1, pp. 59–76, Jan. 2002.

[65] P. Punitha and D. S. Guru, "An Invariant Scheme for Exact Match Retrieval of Symbolic Images: Triangular Spatial Relationship Based Approach," *Pattern Recognition Letters*, Vol. 26, No. 7, pp. 893–907, May 2005.

[66] P. Punitha and D. S. Guru, "An Effective and Efficient Exact Match Retrieval Scheme for Symbolic Image Database Systems Based on Spatial Reasoning: A Logarithmic Search Time Approach," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 18, No. 10, pp. 1368–1381, Oct. 2006.

[67] Y. Rui, T. S. Huang, and S. F. Chang, "Image Retrieval: Current Techniques, Promising Directions and Open Issues," *Journal of Visual Comm. and Image Representation*, Vol. 10, pp. 39–62, 1999.

[68] C. L. Sabharwal and S. K. Bhatia, "Image Databases and Near-Perfect Hash Table," *Pattern Recognition*, Vol. 30, No. 11, pp. 1867–1876, Nov. 1997.

[69] H. Samet, "The Quadtree and Related Hierarchical Data Structure," *ACM Computing Surveys*, Vol. 16, No. 2, pp. 187–260, June 1984.

[70] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-Based Image Retrieval at The End of The Eearly Years," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 12, pp. 1349–1380, Dec. 2000.

[71] J. R. Smith and S. F. Chang, "Visualseek: A Fully Automatic Content-Based Query System," *Proc. of the Fourth ACM International Conf. on Multimedia*, pp. 87–98, 1996.

[72] H. Tamura and N. Yokoya, "Image Database Systems: A Survey," *Pattern Recognition*, Vol. 17, No. 1, pp. 29–43, 1984.

[73] S. L. Tanimoto, "An Iconic/Symbolic Data Structuring Scheme," *Proc. of the Joint Workshop on Pattern Recognition and Artificial Intelligence*, pp. 452–471, June 1976.

[74] J. Z. Wang, J. Li, and G. Wiederhold, "SIMPLIcity: Semantics-Sensitive Integrated Matching for Picture Libraries," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 23, No. 9, pp. 947–963, 2001.

[75] J. C. Wong and A. Datta, "Animating Real-time Realistic Movements in Small Plants," *Proc. of the 2nd Int. Conf. on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, pp. 182–189, 2004.

[76] T. C. Wu and C. C. Chang, "Applications of Geometric Hashing to Iconic Database Retrieval," *Pattern Recognition Letters*, Vol. 15, No. 9, pp. 871–876, Sept. 1994.

[77] W. H. Yeh and Y. I. Chang, "An Efficient Signature Extraction Method for Image Similarity Retrieval," *Journal of Information Science and Engineering*, Vol. 22, No. 1, pp. 63–94, Jan. 2006.

[78] A. Yoshitaka and T. Ichikawa, "A Survey on Content-Based Retrieval for Multimedia Databases," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 11, No. 1, pp. 81–93, Jan./Feb. 1999.

[79] H. J. Zhang and D. Zhong, "A Scheme for Visual Feature-Based Image Indexing," *Proc. of SPIE Conf. on Storage and Retrieval for Image and Video Databases III*, pp. 36–46, Feb. 1995.

[80] X. M. Zhou and C. H. Ang, "Retrieving Similar Pictures from a Pictorial Database by an Improved Hashing Table," *Pattern Recognition Letters*, Vol. 18, No. 8, pp. 751–758, Aug. 1997.

[81] X. M. Zhou, C. H. Ang, and T. W. Ling, "Image Retrieval Based on Object's Orientation Spatial Relationship," *Pattern Recognition Letters*, Vol. 22, No. 5, pp. 469–477, April 2001.

[82] X. S. Zhou and T. S. Huang, "CBIR: From Low-Level Features to High-Level Semantics," *Proc. of the SPIE Image and Video Communication and Processing*, Vol. 3974, pp. 426–431, 2000.