Modelling and Retrieving Multimedia Data Using Spatial and Temporal Information

by

Mohammad Nabil

Thesis submitted in fulfillment

of the requirements for the degree of

Doctor of Philosophy

School of Computer Science and Engineering

University of New South Wales

Sydney, NSW 2052, Australia

July 1997

Acknowledgements

There many people I should thank in regard to this work and no doubt I will omit some. To these I can but beg forgiveness.

I wish to thank my supervisor Dr. Anne H.H. Ngu and my co-supervisor Dr. John Shepherd for their guidance, encouragement, enthusiasm, technical comments and constructive criticism through all years of my candidature. I really feel indebted to them, and their friendship is deeply valued.

I would like to thank to AusAID for financial support during all years of my research. Without this support this work would not have been possible.

I would also like to thank The Government of The Republic of Indonesia, that encouraged me to take further education with financial and moral support so that this work can be accomplished.

To my friends in the "boat", especially Banchong Harangsri, I am indebted with their moral support with friendly faces and smiles. To Mohammed Waleed Kadus, I really appreciate your proof reading.

Finally I feel deeply indebted to my dear wife, Syarifah Chadijah, for her moral support and patience during the course of this endeavour. I also thank my parents who constantly supported and encouraged me to accomplish this work.

I dedicated this thesis to my parents, my wife, my children, Hasan, Fathimah, Ahmad and Zainab for their love and moral support.

Abstract

This thesis presents a new symbolic model for static and dynamic multimedia data. A novel graph-based approach is developed to model objects and their relationships in these data. The model is extensible in the sense that object features (such as colour, texture and shape) can be added when needed.

In this system, the nodes in the graph represent objects and edges represent relationships between objects. For static multimedia data, such as pictures, the relationships are spatial relationships called 2D Projection Interval Relationships (2D-PIR), which combine topological and directional relationships. For dynamic multimedia data, such as moving objects, the model integrates the representations of individual moving objects using tracks and the time-varying relationships between multiple moving objects in a scene, which are temporal sequences of 2D-PIR.

To support content-based similarity retrieval of pictures, a similarity measure has been developed based on distance between components of the 2D-PIR representation. We compare the 2D-PIR similarity matching algorithm with keyword-based matching and Gudivada's image similarity matching method by conducting experiments on several collections of real images. The results of our experiments confirm that the 2D-PIR similarity matching algorithm is more accurate than keyword-based matching and performs better than Gudivada's method. We also compare our model with several other models that use spatial relationships. Our model is more powerful since it is extensible and provides more accurate spatial relationships with a well defined similarity measure. In addition, we also provide a formalism for 2D-PIR to support (a limited form of) reflection and rotation invariance, which is not addressed in other models (except Gudivada's).

Our model for moving objects is supported by several powerful operations. Using those operations one can retrieve moving objects on scenes both with exact matching and similarity matching. Using the projection operation, one can create a new scene by isolating an object(s) from a scene. A new scene can also be created by overlaying existing scenes using the overlay operation. A query on moving objects can be posed

by an animation sketch. In addition, traditional queries using a query language are supported.

We compare our model for moving objects with several related models. So far, our model provides more features than other models. For example, our model supports both tracks/trajectories and spatiotemporal relationships. Our model also supports operations such as the projection and overlay operations which are not available in others models.

Prologue

Information retrieval is now witnessing a booming interest in the light of the ever-growing amount of information repositories available on both local and, especially, distributed platforms, including the Web and the Internet at large. The availability of multimedia documents has given a new twist to information retrieval research, unfortunately setting even farther in the future the time in which generalised, completely automatic indexing methods will be available that allow the answering of content-based queries. A system really capable of fully automatic recognition of "Authoritative sources on the social impact of nuclear waste disposal" seem currently out of reach, even more so for needs of "piano sonatas in styles influenced by art nouveau", "cubist paintings representing violins" or "scenes on sieges from war movies". While in a few decades these information needs might be satisfied by fully automatic systems, for today and tomorrow we will have to make do with systems relying on a mixture of automatic, semi-automatic and manual indexing methods, as it is only through manual intervention that it is possible to inform a retrieval system of the ins and outs of a document.

(Quoted from the call for position papers: Workshop on Conceptual Modelling in Multimedia Information Seeking, 1997).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Development of Multimedia Information

In recent years, we have witnessed a rapid growth of multimedia applications in fields such as education, entertainment and so on. The growth has been intensified by the rapid development of computer technology, especially powerful workstations and high capacity digital storage systems (e.g. optical discs). Accordingly, the cost of the hardware required for storage, transmission and presentation of multimedia data is rapidly decreasing. On the other hand, the software that manages multimedia data is still far from satisfactory. The majority of current multimedia software deals primarily with the presentation aspect; efficient storage and retrieval, especially content-based retrieval, has not been well addressed. [8, 58].

The size of multimedia data is very large; for example, audio and video data may exceed gigabytes [51, 59]. Thus the need for very large storage is obvious. In addition, the need for "smarter" access methods for data stored is also important. Recent advances in optical storage technology provide the possibility of holding video images and other large data. What we need are techniques for managing those large data sets so that they can be used effectively [96].

One of the important challenges in the development of multimedia database systems is the *effective* and *efficient* retrieval of multimedia data. Traditional database retrieval methods, which are based on exact matching alone, are no longer appropriate. The approximate retrieval techniques from Information Retrieval are more suitable.

1.2 Motivation and Objectives

Current research on content-based image retrieval emphasises visual features such as colour, shape and texture. One of the major shortcomings of these features is that relationships among objects in an image cannot be expressed, hence the accuracy of retrieval is reduced. There is some research that explores spatial relationships for image retrieval. However, most of this research focuses on traditional spatial relationships, especially directional relationships.

Research on dynamic multimedia data such as moving objects or video is in an embryonic stage. Most current video clip retrieval systems use text annotation and hence retrieval is based on text. Some systems such as QBIC [32] use an image retrieval technique to retrieve video clips (that is a representative frame is chosen on a video clip and retrieval of an video clip is based on its representative frame).

A video clip is a sequence of frames which are images. If we can create a model for images, then the model should be able to be extended to cover moving objects stored in a video clip. In other words it is possible to create a model both for static and dynamic multimedia data using the same framework. From the model, it should be able to derive query facilities that can answer both exact and similarity retrieval queries. Therefore the objective of this thesis is to create a unified model for static and dynamic multimedia data.

We consider a graph-based model as appropriate for this objective. A node in a graph represents an object. Features of the object can be attached to the node. The edge between nodes represents a relationship(s) between objects. For an image, this relationship is a spatial relationship. For moving objects stored in a video clip, this relationship is a spatiotemporal relationship(s). Queries can be posed to objects (nodes), relationships (edges) or both. Image or video clips matching can be approached by graph matching.

1.3 Static Multimedia Data

Static multimedia data are data that do not change over time, such as text and images. In the traditional database approach, images are treated as *b*inary *l*arge *ob*jects (blobs). For the purpose of retrieval, each blob is associated with a manually-entered textual description (annotated blob). Thus, the retrieval of a blob is conducted via its textual description using known text retrieval methods [18, 90]. This approach has some advantages, primarily that it makes use of mature text retrieval technology. However, it also suffers from well-known shortcomings; for example the description is based on the subjective perception of the annotator and may potentially be incomplete.

Currently, there is considerable research underway to develop "content-based" image retrieval. Content-based retrieval works by matching the query against an (semi)automatically-generated representation of the content of the image. Note that queries no longer need to be specified in a textual form; they could, for example, be given in the form of a rough sketch of the desired image, or by selecting from a set of image templates. The retrieval engine returns a set of candidate matches which have high "similarity" (as defined by a matching function); the final decision on which candidates best satisfy the user's requirements is left to a second stage of filtering.

Current approaches in content-based retrieval of pictorial data (images and graphics) are based on the features of the corresponding picture. There are two kinds of features: visual features (such as colour, texture and shape) and relationship features (spatial relationships among the objects in a picture).

Both visual and relationship features of images can be used for content-based retrieval. Visual features could potentially be used as a basis for "coarse-grained" image similarity retrieval (e.g. "Find images which are predominantly blue"). If we extract visual features for individual objects, we can ask more "fine-grained" queries (e.g. "Find images containing a yellow butterfly and a red rose"). However, it is not until we introduce spatial relationships into queries that we can ask more precise queries such as "Find images containing a yellow butterfly *on top of* a red rose". While a general-purpose image query language will involve both visual and spatial relationship features, in this thesis we concentrate our attention towards spatial relationships.

An object occupies a region of space of an image can be defined by a set of points (pointset). Using the pointset theory we can derive a unified topological and directional relationships between objects in an image. We define the representation of these relationships, called 2D Projection Interval Relationships (2D-PIR), in [86, 87].

2D-PIR provides a symbolic representation of an image where each object in the image is represented by an identifying symbol and the description also includes spatial relationships (both directional and topological) among objects in the image. A symbolic image is a digraph where the nodes of the graph are symbols representing spatial objects and the edges of the graph are labelled by 2D-PIR relationships. Note that our representation can be easily extended to include visual features. Also, since in practice the symbol representing an object is either the name of the object or a concept that represents it, we can make use of concept-based retrieval mechanisms within the 2D-PIR scheme.

1.4 Dynamic Multimedia Data

Dynamic multimedia data are data that change over time, such as moving objects in video sequences or animations. Compared to static multimedia data such as images, dynamic multimedia data are far more complex. In images the main relationships among objects are spatial relationships. In dynamic multimedia data the main relationships are not only spatial relationships, but also temporal relationships and spatiotemporal relationships.

There is already a substantial body of research in spatial databases [46, 97] and temporal databases [61]; however research on the integration of space and time in the database area is still in its infancy [2]. As a consequence, research on the handling of large volumes of information about moving objects (e.g. for queries on video sequences) is also very limited [71]. On the other hand, motion has been intensively studied in computer vision [4, 94].

Most current video retrieval systems use textual descriptions (or video scripts) to retrieve video clips from a database [72, 91, 95]. A content-based video retrieval system based on visual features has also been addressed [118]. However, content-based video retrieval using spatiotemporal features (in particular, spatiotemporal relationships) has not yet been developed. Future multimedia database systems will be required to answer queries such as: "find an object whose trajectory is similar to the trajectory drawn in a window" or "find a scene when object A enters region B from the left". Thus, they will require the capability to answer queries on moving objects based on their spatiotemporal aspects. In order to develop such capabilities, we first require a model for moving objects.

A stationary object occupies a region of space that can be defined by a set of points (pointset). If we use a single point (e.g. centroid) to define the position of the object, this can be represented as a point in an n-dimensional space (we will consider primarily $n = 2$, but our model extends in a straightforward way to $n > 2$). For a moving object this position changes over time.

Any moving object can be considered as a sequence of points in $n+1$ dimensions where the n dimensions are spatial and the extra dimension is temporal. It is thus a kind of spatiotemporal object. In other words, a moving object is a sequence of spatial locations over time. The motion of such a moving object can be described by a sequence of spatiotemporal points, which we call a *track*. Each spatiotemporal point is of the form $(x_1, x_2, \ldots x_n, t)$, where the x_i's are spatial coordinates, and t is a temporal coordinate. Thus, for example, the spatiotemporal points for a two-dimensional moving object are of the form (x, y, t) where x is x-coordinate, y is y-coordinate and t is time coordinate. Note that this is a very low level representation of a track; we have also developed a more compact representation using *distance* and *symbolic direction*, which is described in Chapter 5.

A model for moving objects must be able to deal with both the tracks of individual moving objects and with relationships among objects. To achieve this objective, we extend our graph-based symbolic model of static multimedia data (image) to incorporate moving objects. In this model, each node of the graph is a symbol representing an object, a loop in a node is labelled by a track (for a moving object) and each edge is labelled with a relationship between two objects. These relationships are developed based on our previously developed concept of spatiotemporal relationships [84, 85].

1.5 Thesis Scope

The work in of this thesis includes representation and modelling of static and dynamic multimedia data, defining similarity measures for supporting retrieval of static and dynamic multimedia data and providing query model for retrieving dynamic multimedia data (especially moving objects). We consider a query model for static multimedia data is not necessary since retrieval of static multimedia data such as images is better addressed using a graphical user interface that supports similarity retrieval.

Our work relies on objects in images having been identified. At present, this is done manually. For video data, a preprocessing facility to separate cuts or clips is assumed. Objects in a clip will be identified manually. In general automatic object identification is beyond the scope of current image processing. We assume that image processing will eventually solve this problem, so that our techniques will find wide applications.

1.6 Results

Prototype image and moving-object retrieval systems have been implemented using the Java language. Some experiments have been conducted on image similarity retrieval and trajectory similarity retrieval.

Experiments on similarity image retrieval show that if we compare keyword-based retrieval with 2D-PIR similarity retrieval to retrieve images, the precision value in 2D-PIR is much higher. For example, the precision value increases by a factor of 2.7 and 3.0 for threshold values of 50 and 60 respectively. This is because 2D-PIR uses more information, such as directional relationships and topological relationships among objects. When we compare 2D-PIR similarity retrieval with Gudivada's similarity retrieval. 2D-PIR similarity retrieval performs better on most collections of images tested.

We also briefly compare our model on static multimedia data (2D-PIR) with other related models, so far our model is the most expressive one since spatial relationships in 2D-PIR are richer than other spatial relationships in other models (2D-PIR support a unified topological and directional relationships, while most of other model support only directional relationships).

The results of experiments conducted on trajectory similarity retrieval indicate that our measure for trajectory similarity is effective. So far no other experiments on trajectory similarity retrieval have been published.

Our major contribution is a graph-based model for representing an image (and moving objects). There are two important components in the model: spatial objects and relationships. Currently spatial objects are represented by a symbolic name and have no attributes. However adding attributes to a spatial object such as colour, shape etc. is straightforward. We introduce a unified spatial relationship called 2D-PIR to represent spatial relationships among spatial objects in an image. 2D-PIR can represent 169 spatial relationships robustly. We develop formal similarity measure on 2D-PIR which leads to an effective method for image similarity retrieval.

We extend our image model for representing moving objects in a scene. To capture how an object moves we develop spatiotemporal relationships based on direction and distance travelled of an object over time which we call a *track*. We also develop spatiotemporal relationships based on the changing 2D-PIR relationships between moving objects over time which we call Temporal Sequence of 2D-PIR (TS-2D-PIR). A set of powerful operations on moving objects are derived from this model. Based on these operations we build a query language for moving objects.

Both models (image and moving objects) support graphical queries (sketches). Each graphical query is transformed to a corresponding graph representation which is then compared to graph representations stored in a database.

1.7 Thesis Organisation

This thesis is composed of eight chapters. Chapter 2 provides a comprehensive summary of related work ranging from spatial relationships and temporal relationships to content-based image/picture, video sequence and moving object retrieval. This chapter also discuss some fundamental concepts of pointset theory. In Chapter 3, we develop the 2D-PIR representation and discuss how to deal with rotation and reflection of 2D-PIR pictures. A similarity measure for similarity retrieval of pictures using 2D-PIR is also provided in this chapter. Chapter 4 discusses a low level representation for 2D-PIR, called a *bi-array*. In this chapter we demonstrate a system prototype for picture retrieval. We also discuss some experiments on the prototype and we compare our system with other image retrieval systems. Chapter 5 discusses a model for moving objects such as *tracks* and spatiotemporal relationships in detail. We also discuss operations on moving objects and scenes. In Chapter 6, we discuss the representation of *tracks* and spatiotemporal relationships using an *extended bi-array* structure. We illustrate a system prototype for moving objects and describe experiments of trajectory similarity retrieval. Chapter 7 provides example of applications of moving-object data model. Finally, Chapter 8 provides a conclusion and remarks for future work.

Chapter 2

Fundamentals and Related Work

2.1 Introduction

Currently, multimedia information systems are becoming very popular. One possible definition of a multimedia system is a complete system of hardware and software which make it possible to create, control, integrate, retrieve and present material consisting of text, graphics, still images, animation, motion video, sound, and human-computer interactivity [23]. To support those functionalities, a multimedia system needs additional input/output devices besides the standard input/output devices of the computer. For example, still images can be input via a scanner, voice can be captured with a microphone, and motion pictures can be captured by video camera [7].

Current multimedia software mainly focuses on how to present multimedia data. This is the reason why issues such as the content of multimedia data, relationships in and between media have not been thoroughly addressed. Consequently, there are some quite severe restrictions on how users can interact with multimedia data.

On the other hand, multimedia data is extremely rich. For example an image contains far more information than its textual description. A sequence of images, such as a video clip, that contains vast information is another example. There is a great need to develop a representation of multimedia data so that the richness of its information can be explored and utilised as much as possible. Images and video clips carry visual information such as colours, shapes and textures of objects and relationships information among those objects. In images these relationships are mainly spatial relationships. In video clips these relationships can be spatial, temporal and spatiotemporal relationships.

Not only a representation of multimedia data is needed but also a data model should be developed so that database systems for these data that exploit their richness can be developed.

In the following sections we will discuss multimedia database systems in general. We then look at the roles of spatial, temporal and spatiotemporal relationships in multimedia data and discuss retrieval of multimedia data, especially retrieval of pictures or images and their sequences (video).

2.2 Multimedia Databases

Multimedia databases are an active research area. Researchers in the database community are likely to agree that multimedia data are data of the form text, sound, video, images, graphics and traditional alphanumeric data [7, 9, 73, 89]. Some earlier researchers [9, 114] did not include motion picture (video and animation). Jasmine [50], a system which is claimed as an object-oriented multimedia knowledge base management system does not support sound and video (also animation) data. On the other hand, Chen and Little [16] suggest that a multimedia object may consist of components of any data type.

A multimedia database system is a system capable of storing and retrieving multimedia data effectively. A multimedia database system is much more complex than a conventional database system. For example, in a multimedia database, many relationships should be considered including spatial and temporal relationships since multimedia data may contain image, graphics, audio, video or animation. A new set of query operators for each new media type, for example, "similar" for pictures and motion pictures should be considered as well. Equality is not a useful operator for picture retrieval. For example, it is not useful to ask "Give me all pictures in the database that are exactly the same as the picture I have here." "Similarity" is the most important matching operator for image data. In addition, data presentation becomes a major problem when many types of media are used. For example synchronisation of audio, video and text data in a motion picture with subtitles must be considered as well [39].

Furthermore, multimedia data consume abundant space and bandwidth compare to conventional data [16].

Multimedia data contain many relationships between media and between objects in a medium and some of these relationships spatial and temporal relationships. We will concentrate on spatial and temporal relationships in multimedia data since these relationships are important in multimedia data.

2.3 Spatial Relationships and Their Role in Multimedia Data Retrieval

Multimedia data include spatial data such as maps, graphics and images, therefore multimedia data retrieval should consider spatial relationships among spatial objects. For example, the following queries explicitly use directional relationships as constraints to retrieve objects: "find a gas station which is located west of a region in a map" or "find a picture containing a computer on the top of a table to the right of a bookcase". Another possible query is "give me images in which the objects arranged similar to this image (pointed using a mouse)."

Spatial relationships can be classified into two groups: quantitative spatial relationships in term of distance and qualitative spatial relationships. There are two well known qualitative spatial relationships: directional relationships such as *leftOf, rightOf, above, below,* etc., (or in terms of *east, west, north, south,* etc.) and topological relationships such as *disjoint, touches, overlaps, inside* etc. Since our concern is mainly on topological and directional relationships, the distance relationship is not discussed here.

2.3.1 Topological relationships

Pointset theory

Topological relationships are relationships which are invariant under topological transformation (that is, they are preserved if the objects are translated, rotated, or scaled) [21]. Before we discuss topological relationships in detail, let us review some important concepts in topology. These are standard concepts in topology, so any standard textbook (such as [57, 81]) can be referred to for more details.

Definition 1 *Let X be a set. A topology on X is a family of $\mathcal{A}$ of subsets of X that satisfies the following conditions: (1) each union of members of $\mathcal{A}$ is also a member of $\mathcal{A}$ (closed under arbitrary union); (2) each finite intersection of $\mathcal{A}$ is also a member of $\mathcal{A}$ (closed under finite intersection); and (3) $\emptyset$ and X are members of $\mathcal{A}$.*

The pair $(X, \mathcal{A})$ is called a topological space and elements of this space are called points. The members of $\mathcal{A}$ are called *open sets* and their complements in X are called *closed sets*. Note that the family of closed sets has the following properties: (1) closed under arbitrary intersection; (2) closed under arbitrary union; and (3) $\emptyset$ and X are members of the family.

Definition 2 *Let $(X, \mathcal{A})$ be a space and x be a point in X. A neighbourhood of x is an open set U such that $x \in U$.*

Definition 3 *Let $Y \subset X$. The interior of Y, denoted by Y^0, is the largest open set contained in Y or $Y^0 = \cup(U \mid U \text{ is open } \wedge U \subset Y)$.*

A more intuitive definition of interior is as follows: If Y is a set of points (pointset) on X, a point of p in Y is said to be an interior point of Y if there is a neighbourhood U of p such that $U \subset Y$, then the collection of all interior points of Y is the interior of Y (Y^0).

Definition 4 *Let $Y \subset X$. The* closure *of Y, denoted by $\bar{Y}$, is the smallest closed set containing Y or $\bar{Y} = \cap(F \mid F \text{ is closed} \wedge F \supset Y)$.*

Definition 5 *Let $Y \subset X$. The* boundary *of Y, denoted by ∂Y, is the intersection of the closure of Y and the closure of the complement of Y.*

Definition 6 *Let $Y \subset X$. The* exterior *of Y, denoted by Y^{-1}, is X which is not in Y^0 and ∂Y or $Y^{-1} = X - (Y^0 \cup \partial Y)$).*

The relationship among $Y^0, \bar{Y}$, and ∂Y is $Y^0 \subseteq Y \subseteq \bar{Y}$. If A is an open set then $Y = Y^0$, that is ∂Y is not part of Y. If Y is a closed set then $Y = \bar{Y}$, that is ∂Y is part of Y. For the sake of simplicity we assume that all pointsets are closed sets.

Topological relationships derived from the pointset theory

Egenhofer [27] uses the concepts of *boundary* and *interior* of pointsets to derive topological relationships between two-dimensional regions (pointsets) embedded on a two-dimensional plane (R^2). If A and B are two regions, A^0 and ∂A denote the interior and boundary of A respectively, then all of the possible topological relationships between A and B can be derived from the possible combinations of intersection between their boundaries and interiors; that is $\partial A \cap \partial B, A^0 \cap B^0, \partial A \cap B^0$, and $A^0 \cap \partial B$. Each of these intersections is either empty (0) or non-empty (1). This results in 16 possible combinations as shown in Table 2.1.

There are nine valid topological relationships between two regions. But r_5 and r_6, r_{13} and r_{14} are inverses of each other. There are two *overlaps* relationships (r_7 and r_{15}) since disconnected boundaries are allowed. Note that r_7 is overlaps with disjoint boundaries and r_{15} is overlaps with intersecting boundaries. If boundaries are assumed to be connected, then r_7 is excluded. Hence, there are eight topological relationships: *disjoint, contains, inside, overlaps, touches, equals, covers* and *coveredBy* as shown in Figure 2.1. Note that these relationships are mutually exclusive.

The above formalism for deriving topological relationships is called 4-*intersection* which is good only for co-dimension 0 (the object and the embedded space have the same dimension, in this case 2D). An extension of this formalism, called 9-*intersection* [28, 30], uses *exterior* to enhance the expressive power of topological

	$\partial A \cap \partial B$	$A^0 \cap B^0$	$\partial A \cap B^0$	$A^0 \cap \partial B$	Relationships' Name
r_0	0	0	0	0	A and B are *disjoint*
r_1	0	0	0	1	*
r_2	0	0	1	0	*
r_3	0	0	1	1	*
r_4	0	1	0	0	*
r_5	0	1	0	1	A *contains* B or B *inside* A
r_6	0	1	1	0	A *inside* B or B *contains* A
r_7	0	1	1	1	A *overlaps* B
r_8	1	0	0	0	A *touches* B
r_9	1	0	0	1	*
r_{10}	1	0	1	0	*
r_{11}	1	0	1	1	*
r_{12}	1	1	0	0	A *equals* B
r_{13}	1	1	0	1	A *covers* B
r_{14}	1	1	1	0	A *coveredBy* B
r_{15}	1	1	1	1	A *overlaps* B

*) indicates there is no topological relationship (proved in [27]).

Table 2.1: Topological relationships created from four intersections

relationships by considering relationships of objects (pointsets) with the embedding space.

Clementini, Felice and Oosterom [21] also use pointset theory like Egenhofer, but they take account of the dimension of intersection, which can be no dimension (empty set), 0D (a point) , 1D (a line) and 2D (a region). A calculus-based method is used to define topological relationships. A comparison of this work with Egenhofer's can be found in [20].

Other approaches for defining topological relationships are discussed in [22, 111]. Topological relationships are defined using a primitive dyadic relation: $C(x, y)$, x is connected with y which is defined if x share some parts with y or when x and y are in contact. The following are topological relationships defined in [22]: $DC(x, y)$, x is disconnected from y; $P(x, y)$, x is a proper part of y; $x = y$, x is identical with y; $O(x, y)$, x overlaps y; DR(x,y), x is discrete from y; $PO(x, y)$, x is partially overlaps y ; $EC(x, y)$, x is externally connected with y; $TPP(x, y)$, x is a tangential proper part of y; and $NTPP(x, y)$, x is non-tangential proper part of y. Note that the relationships P, PP, TPP and NTPP have inverses.

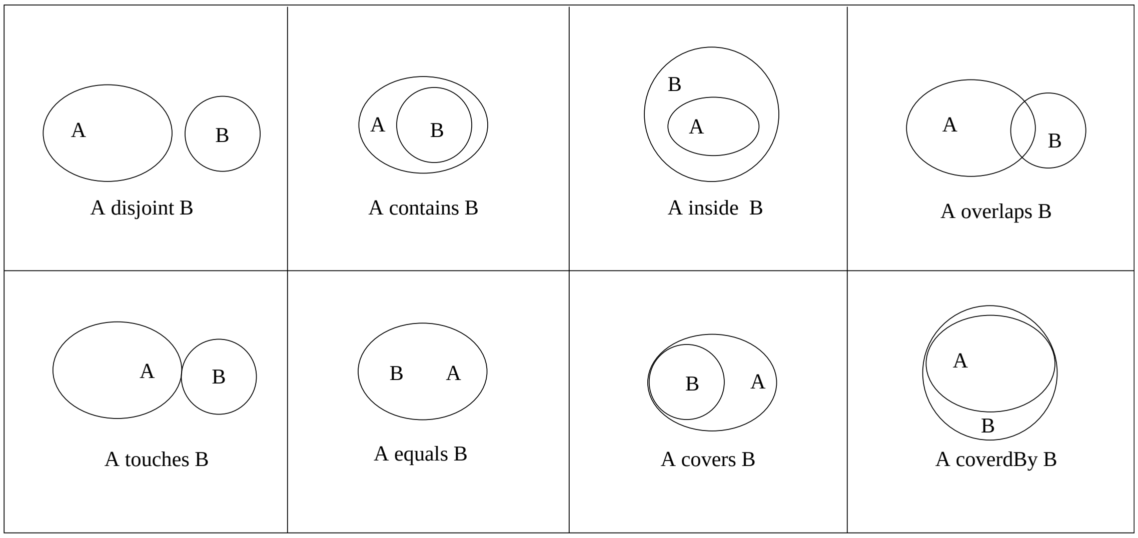


Figure 2.1: Mutually exclusive topological relationships

Unlike topological relationships defined in [27], not all topological relationships defined in [22] or [111] are mutually exclusive. The topological relationships DC, EC, PO, =, TPP, NTPP and the inverse of TPP and NTPP are mutually exclusive which are equivalent to *disjoint*, *touches*, *overlaps*, *equals*, *coveredBy*, *inside*, *covers*, and *contains* respectively.

Even though topological relationship defined by [22] and [111] are equivalent to the topological relationships defined by Egenhofer [27], in general the topological relationships defined by [27] has a strong theoretical foundation in pointset theory and in algebraic topology. This is the reason why the topological relationships defined by [27] and its extension [28, 30] are popular, especially in GIS community.

2.3.2 Directional relationships

Directional relationships are heavily used in everyday life. We frequently mention an object based on its directional relationship with another object. For example, a mouse is to the right of a keyboard or Brisbane is north of Sydney. According to [93] and [109] directional relationships are fuzzy concepts since they depend on human interpretation. Note also that directional relationships are influenced by the angle of view and orientation of objects.

Peuquet and Ci-Xiang [93] have developed a model for directional relationships between polygons by incorporating size, shape and distance. They also provide an algorithm to determine directional relationships between polygons in a 2D-space. Some characteristics of directional relationships are: they are binary, each direction is coupled with a semantic inverse, the area of acceptance for any given direction increases with distance. Some other characteristics are that the area of acceptance for any given direction increases with the dimension in the facing direction of the reference object in relation to the second object, and in the selection between a pair of objects as the reference, the one which is "perceptually prominent" (the larger one) is preferred.

In [109], fuzzy concepts were introduced into directional relationships, to account for the fact that such relationships are not necessarily exclusive. For example, if an object is above and to the left of another object, it may be perceived as being *above*, *leftOf*, or some degree of both of these.

Chang's 2D-strings [14] represent directional relationships among objects in a picture. The representation is constructed from the original picture by stating directional relationships relative to a grid superimposed on the picture. An example is shown in Figure 2.2, where the upper-case letters represent spatial objects lying in the region of the plane bounded by each grid cell. Note that object A is large enough that it spans two cells.

DE		
	B	C
A	A	

Figure 2.2: A symbolic picture

In Figure 2.2, we can straightforwardly determine directional relationships among objects. For instance, object B is on the *leftOf* object C, object D is in the same set as (i.e. "near to") object E, and object C is on *upperRight* of object A.

A 2D-string is a pair made up of the "symbolic projections" of a picture along the x-axis and y-axis. Construction of a 2D-string for a picture commences by laying a grid over the picture; the idea is that objects are partitioned over the cells of the grid (as in Figure 2.2). To generate a 2D-string from a picture, we scan along the x-axis, looking at the projections of objects on this axis, and describing the relationships between the objects using the operators "=", "<" and ":" ("=" represents the relationship "at the same spatial location as", "<" represents the relationship "to the left of" and ":" represents "in the same set as" relation). For example, in Figure 2.2, the x-axis relationships are: $A = D : E < A = B < C$. We then repeat this scan along the y-axis, to generate another string of relationships between objects (or, more precisely, their projections on the y-axis). In the example, this corresponds to: $A = A < B = C < D : E$. The 2D-string of the picture is the pair of object relationship strings.

More formally, a 2D-string is defined as follows: Let V be a set of symbols representing spatial objects. Let A be the set {"=", "<", ":"} of spatial relationships where $A \cap V = \emptyset$. A 1D-string over V is any string $x_1 x_2 ... x_n, n \geq 0$, where the x_i are in V. A 2D-string over V, written as (u, v) is defined to be $x_1 y_1 x_2 y_2 ... y_{n-1} x_n,\ x_{p(1)} z_1 x_{p(2)} z_2 ... z_{n-1} x_{p(n)}$, where $x_1 ... x_n$ is a 1D-string over V, $p : \{1, ..., n\} \to \{1, ...n\}$ is a permutation over $\{1, ...n\}$, $y_1, ...y_{n-1}$ is a 1D-string over A and $z_1, ...z_{n-1}$ is a 1D-string over A. For Figure 2.2, V = {A,B,C,D,E}. The 2D-string representing it is $(A = D : E < A = B < C, A = A < B = C < D : E)$. If the symbol "=" is omitted then the 2D representation becomes $(AD : E < AB < C, AA < BC < D : E)$ which is called *normal* 2D-string. If the symbol ":" is also omitted then the 2D-string becomes $(ADE < AB < C, AA < BC < DE)$ which is called *reduced* 2D-string. The *augmented* 2D-string is *reduced* 2D-string where a permutation function is included in 2D string representation. The *augmented* 2D-string of the above picture is $(ADE < AB < C, AA < BC < DE, 145623)$.

There are several extensions of 2D-string such as extension by Jungert [54] and [55, 56]. Another extension is 2D C-string [66, 67, 68] that enhances 2D-string in term of cutting lines for partitioning objects. However these extensions do not change the nature of the original 2D-string in representing directional relationships.

2.3.3 Integrated spatial relationships

There is not much research concerning the integration of various spatial relationships. Hernandez [47] used topological-orientation pairs (orientations are directions in term of *rightOf, leftOf* etc.) to determine qualitative positions of objects in 2D-space. He combined the neighbourhood graph structure of topological with directional relationships to improve the efficiency of the propagation algorithm in spatial reasoning. Zimmermann [120] proposed an approach to combine orientation, position and distance (qualitative) to enhance spatial reasoning. Cui [22] attempted to define directional relationships using a similar approach to that used in defining his topological relationships. Papadias and Sellis [92] introduced a 2D-array representation to represent topological relationships. It is shown that a 2D-array can also be used to represent directional relationships.

2.4 Temporal Relationships and Their Role in Time-based Multimedia Data

Since multimedia data such as motion pictures, audio and historical image data involve a time dimension, temporal relationships must be considered among these data. In addition the time dimension can be considered as a 1D-space, so spatial relationships among 1D space objects, especially straight lines, are same as temporal relationships.

Temporal relationships between objects in multimedia data may be implied, as in the simultaneous acquisition of voice and video, or may be explicitly formulated, as in the case of a multimedia document which possesses voice annotated text [7, 73].

Many temporal databases or multimedia databases have adopted time intervals as the temporal element. A time instance is defined as a zero-length moment in time, such as “4:00 PM.” On the other hand, a time interval is defined by two time instants (the endpoints) [73].

Allen [3] proposed an interval-based temporal representation and introduced a method to derive relationships between intervals. Intervals can be represented by their endpoints assuming that every interval consists of a fully ordered set of points of time. Thus an interval is represented as an ordered pair of points with the first point less than the second. Based on this representation, 13 relationships (six relationships have inverses, one relationship has no inverse) are derived as shown in Table 2.2.

Relationship	Symbol	Symbol for Inverse	Pictorial Example
X before Y	<	>	XXXXX YYYYY
X equals Y	=	=	XXXXX YYYYY
X meets Y	m	mi	XXXXXYYYYY
X overlaps Y	o	oi	XXXXX YYYYY
X during Y	d	di	XXXXX YYYYYYYYYY
X starts Y	s	si	XXXXX YYYYYYYYYY
X finishes Y	f	fi	XXXXX YYYYYYYYYY

Note: X and Y are intervals

Table 2.2: Allen's interval relationships

Temporal intervals can be used to model multimedia presentation by letting each interval represent the presentation time of some multimedia data element, such as a still image or an audio segment. These intervals represent the time component required for multimedia playout, and the relative positions represent their time dependency. For continuous media such as audio and video, an appropriate temporal representation is a sequence of intervals described by *meets* relations. In this case, intervals abut in time and are non-overlapping, by definition of a continuous medium [73].

The "reverse" relationship is a very useful and important one in multimedia data presentation for continuous media types such as voice, animation, and full motion video. Examples of the reverse relationship include viewing a movie backwards or reversing an animation of a series of image [64].

The thirteen temporal relationships between two intervals of time defined by Allen [3] are sufficient only for the temporal characterisation of simple multimedia presentation [64, 72]. They cannot represent complex n-ary relationships among several multimedia objects in a concise form [64].

Little and Ghafoor [72] introduced n-ary temporal relations as a generalisation of binary temporal relations. As in binary temporal relations, n-ary temporal relations also have thirteen possible temporal relations.

Lee and Nah [89] introduced a delay object to represent potential slack time between multiple media objects. They showed that by using parallel and sequential relations and the delay object, all thirteen temporal relationships can be expressed.

2.5 Attempts at Integrating Spatial and Temporal Information

A comprehensive discussion of spatiotemporal issues in Geographical Information Systems (GIS) has been provided by Langran [63]. She notes that even though spatiotemporal domains are considered very interesting, there are very few attempts to model spatiotemporal data.

Worboys [115, 116] has proposed a spatiotemporal data model based on the integration of *simplex* and *bitemporal* concepts. A simplex is either a single point, finite straight line segment or triangular area. Bitemporal refers to two axes of time: *transaction* time and *valid* time [53]. An *ST-simplex* is a pair of spatial object and bitemporal element (BTE) (that is, *ST-simplex* $= <S, T>$ where S is a *simplex* and T is a BTE). This model is supported by several operations such as temporal and spatial projections, equality and subset operations, topological operations, β-product, and spatial and temporal selections.

In [36, 37], an extension of the relational data model that supports both spatial and temporal dimensions is presented. In this model, spatial objects or regions are considered as sets of points (Z) and certain subsets of Z that are of interest to users are called *spatial elements*. A universe of time instants $[0, NOW] = \{0, 1, 2, \cdots NOW\}$ where NOW denotes the current time, is assumed. An interval is a subset of $[0, NOW]$ and a *temporal element* is a finite union of intervals. A spatiotemporal universe is $Z \times [0, NOW]$ and spatiotemporal elements are of the form $reg_1 \times \mu_1 \cup reg_2 \times \mu_2 \cup \cdots \cup reg_n \times \mu_n$ where the reg_i are the spatial elements and μ_i are the temporal elements. The authors claim that spatiotemporal elements are closed under union, intersection and complement and form a boolean algebra. A spatiotemporal assignment to an attribute A is a function from spatiotemporal elements into the domain of A. A spatiotemporal tuple is simply a concatenation of spatial assignments whose spatiotemporal domain are the same and a spatiotemporal relation r over R with $k \subseteq R$ as its key, is a finite set of non-empty tuples, such that no key attribute value of a tuple change from one spatiotemporal, and no two tuples agree on all their key attributes. This model is supported by an SQL-like query language called ParaSQL [37].

Li et al. [71] have used object-oriented concepts to model spatial, temporal and spatiotemporal objects. A spatial object $O_S = < O_{id}, Area >$, temporal object $O_T = < O_{id}, Range >$ and a spatiotemporal object $O_{ST} = < O_{id}, AreaRange >$ where O_{id} is an object identifier, *Area* is set of points occupied by O_S, *Range* is set of temporal points occupied by O_T and *AreaRange* is set of points occupied by O_S at a given time t. In this model a moving object is considered as a specialisation of a spatiotemporal object. The authors also introduced the concept of *trace*, a special kind of *AreaRange* for moving objects.

Galton [38] combined spatial logic defined in [22] and Allen temporal logic [3] to produce logic for space, time and motion. Several predicates have been defined so that motions of a spatial object with respect to another spatial object can be described using the logic.

Bimbo [10] used (point-based) temporal logic to derive spatial logic of projection of bounding rectangle (both on vertical and horizontal axes) of spatial objects. He extended (point-based) temporal logic to deal with intervals (lines) and also integrated temporal logic and spatial logic to form spatiotemporal logic that can be used to describe changes within images sequence (motion).

In multimedia data, especially time-based multimedia data such as motion pictures and animations both spatial and temporal relationships exist among objects. In fact spatial and temporal relationships are strongly coupled. For example in describing multiple moving objects, both spatial and temporal relationships are used at the same time forming spatiotemporal relationships.

2.6 Content Based Multimedia Data Retrieval

In the traditional database approach, multimedia data such as images or graphics are considered as *b*inary *l*arge *ob*jects (blobs). For retrieval purposes, each blob has a corresponding textual description entered manually (annotated blob). Thus retrieval of a blob is conducted via its textual description using known text retrieval methods [18, 90]. This approach has some advantages, as the retrieval technology of text is quite mature and widely available. However, it suffers from shortcomings; for example the description entered may be incomplete and may vary from person to person depending on his/her perception. According to Meghini *et al.* [76] it is difficult to capture the structure, properties, and relationships of multimedia data stored as a blob.

A current trend in multimedia retrieval is towards "content-based" retrieval [17, 45]. The content-based retrieval concept in multimedia database no longer uses text retrieval approach solely. In other words, a user may enter a query not in the form of text. For example when querying an image, a user may enter a query in the form of a sketch or use an existing picture. The retrieval engine uses the concept of similarity retrieval to answer the query; that is the retrieval engine will retrieve candidate answers which are similar to the query. The final decision as to which candidate is appropriate is left to the user in a second stage of filtering.

At present, interest in content-based similarity retrieval of pictorial data (images and graphics) is based on features of the corresponding picture. There are two kinds of features: visual related features such as colour, texture and shape and relationship feature, that is spatial relationships among objects in a picture.

2.6.1 Similarity retrieval using visual properties

QBIC (Query By Image Content) [6, 32, 90] uses colour, texture and shape to retrieve similar images. Similarity matching based on both average colour and colour histogram is used. For average colour, the similarity measure is based on the weighted Euclidean distance in the colour spectrum. Texture similarity matching is based on weighted Euclidean distance in a three dimensional space (coarseness, contrast, and directionality). Similarity matching of shape is based on the weighted Euclidean distance of several shape features such as circularity, eccentricity, major axis orientation and set algebraic moment invariants. For image sequence retrieval (in video database), the QBIC approach is to generate a representative frame. Once representative frames have been generated, video retrieval is based on the representative frames using the QBIC image retrieval techniques.

Hyperbook on birds [107, 108] uses the concept of similarity retrieval for retrieving images of birds. It has three query modes: silhouette, birdcall and sketch-of-scene. In silhouette retrieval, the user enters queries interactively by selecting a silhouette from a list of silhouette patterns presented by Hyperbook. The user then selects modifications to the pattern, and the modified pattern is then used as the basis for scanning the database of images. Modifications to the pattern include: focus on particular body region, adjust axis of body, adjust size of head, etc. The similarity metric is based on a linear sum of distances from each of the components of the pattern.

Jagadish [52] proposed shape similarity retrieval based on two-dimensional rectilinear (rectangular covers) shapes representation. In this representation an object (a polygon) is divided into several rectangular shape objects. The similarity measure uses the concept of area-difference. Two shapes are similar if the error area (where the two do not match) is small when one shape is placed on the top of the other. In other works [42, 43] the shape of an object is represented by ordered sequence of boundary components which are feature vectors. Similarity retrieval is based on the minimum cost of converting one feature to the other (error-correction substring matching). Mehrota [77, 78] used a set of structural components in the form of maximal curvature boundary points arranged as an ordered set. Shape features are represented as points in a multidimensional space. The similarity of any given two features is defined by the Euclidean distance between the corresponding pair of points in the multidimensional space.

Lu [74] proposed a representation of shape using normalised chain codes [33, 34] of an object's boundary so that the chain codes are invariant under scale and rotation. Using the normalised chain codes, a similarity measurement based on a normalised grid value (1 if a grid is covered by a boundary, 0 otherwise) is proposed.

Scassellati [98] compared several shape similarity measures, including algebraic moments, spline curve distances, cumulative turning angle, sign of curvature and Hausdorff distance. His results are quite interesting. The curve distance (parametric curve distance) is not a good similarity measure for any shape tested. The moment method performs either very well or very poorly on testbed shapes. This method is particularly good for rectangular and arch objects, but it performs very poorly for snake, triangular plane, letter-T and letter-S shapes. The curvature method worked well on shapes that involve protrusion from a central mass. It is not good for shapes that have no protrusion or when there is no central body and it is very bad for T-shapes, squares and ellipses. Hausdorff's distance method performs very well when the test shape is very close to tested shapes but not very good if some flexibility is needed. Scasssellati concluded that the turning angle seems the best one from the above methods since it performs well on any shape tested.

Image retrieval based on colour has been proposed by several researchers [18, 75, 79, 106]. Chua *at al.* [18] have used the colour-pairs technique for fuzzy object-level image retrieval. The basic idea of this technique is to extract a set of distinct colour pairs from an image. This set of colour-pairs is used to model closeness between regions and objects in the image. Similarity of two images can be computed based on sets of colour-pairs, that is if two images contain two similar sets of colour-pairs then the contents of the images are considered similar. Mehtre *et al* [79] have used colour histogram intersections and proposed a distance method and a reference colour method for measuring similarity. In [75] a quad-tree representation of colour called multi-level colour histograms was proposed. The quad-tree representation can handle positional information on colour which is its main advantage over ordinary colour histograms. The similarity between two images is computed based on colour histogram intersection.

2.6.2 Similarity retrieval using relationship properties

Similarity retrieval of images encoded in a 2D-string is based on subsequence matching techniques. In other words, the problem of determining the similarity between a query image and a database image is transformed into a 2D-string subsequence matching problem. Chang [14] defines 2D-string matching based on the rank of symbols in a 1D-string. Based on this, three types of matching, type-0, type-1, and type-2, are defined. The matching algorithm is based on subsequence matching. Lee [70] defines similarity matching on 2D-string based on the longest common subsequence in the string. He reduces 2D-string matching to clique finding in an undirected graph. In [15] a modification of Lee's algorithm for similarity matching of 2D-string is proposed which claims to perform better than Lee's algorithm.

In [109], Takahashi uses orientation (directional relationships) to retrieve pictures. Since directional relationships such as *right, left* or *west, east* are vague, a fuzzy representation of spatial relationships using a heuristic function of geometrical quantities is proposed. In retrieving images, there is no direct similarity measure between images since an exact matching retrieval method is used. However, since an image is also encoded based on spatial relationships, the spatial relationships which are specified in a query will be matched to spatial relationships in images stored in a database based on the estimated degree of fuzziness of the relationships.

The 9-DLT directional relationships representation [13] was proposed to retrieve pictures from a database. A 9-DLT is an ordered 3-tuple (O_i, O_j, r_{ij}) where O_i and O_j are icons representing spatial objects in a picture and r_{ij} is an integer 1..9 which represents directional relationships such as *north, northwest, west* etc. as shown in Figure 2.3. Note that R is the reference object. 1 stands for north of R, 2 stands for northwest of R, etc. and 9 stands for the same location as R.

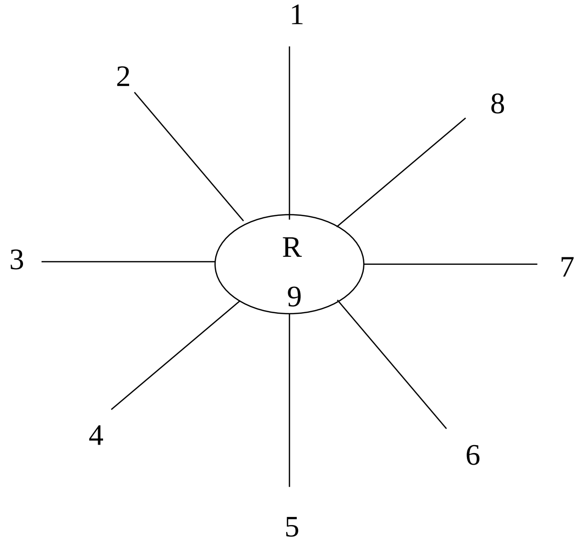


Figure 2.3: The 9-DLT spatial relationship codes.

The consecutive retrieval technique defined in [40] is used to construct a picture-tuple incidence matrix so that similar pictures are arranged closely. Retrieval of similar pictures is efficient since pictures that are similar in term of their directional relationships are basically clustered.

Yu [117] used an E-R model to represent an image. The entity part represents the objects in an image together with their attributes and the relationship part represents the relationships between objects in an image. A relationship can be in the form of a spatial relationship as well as a non-spatial relationship called an action relationship. Figure 2.4 shows en example of a picture represented using E-R model (taken from [117]).

Note that the E-R diagram corresponding to a picture is created manually. Picture retrieval is conducted by similarity retrieval. Given a query picture, this picture is compared to database pictures by comparing their E-R representations. Similarity measures are defined for entities (entity names and attributes) and relationships. The similarity of two pictures is the sum of the similarities of their entities and relationships.

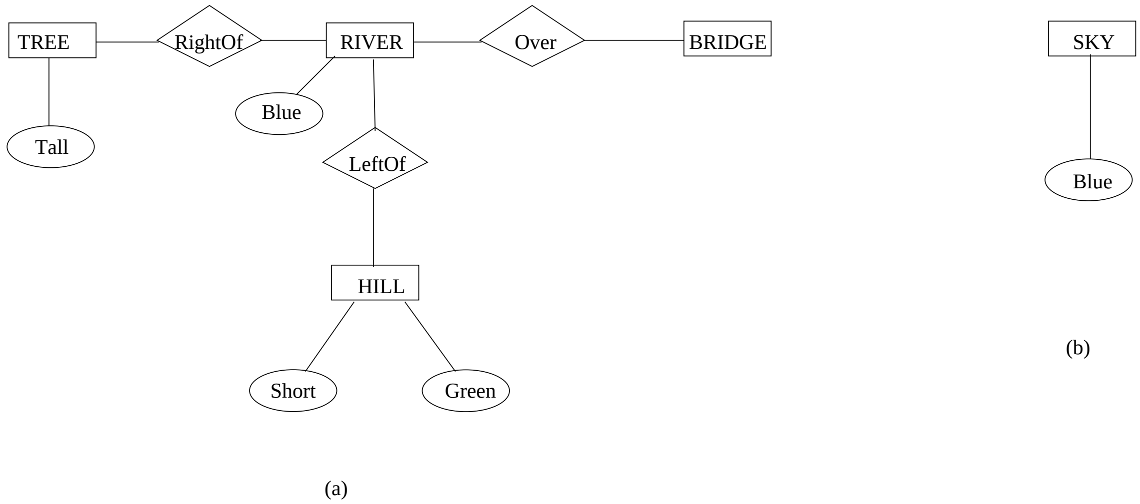
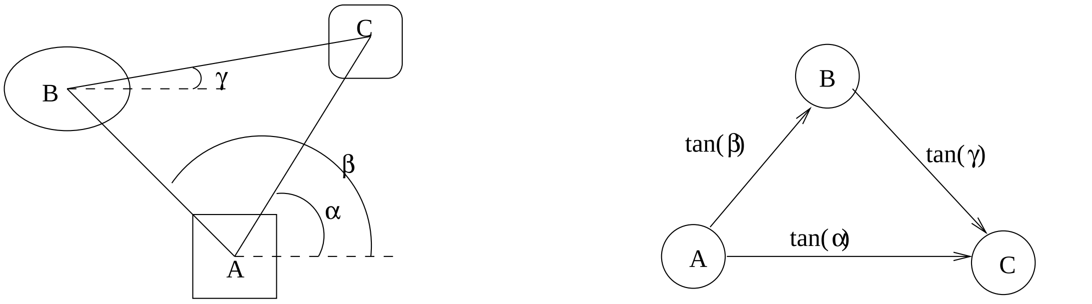


Figure 2.4: E-R digram of a picture (a) Foreground (b) Background.

Regarding spatial relationships, there is a set of rules for deducing spatial relationships between objects, given spatial relationships of other objects [103, 102]. Sistla [104] developed similarity measures for spatial relationships, focusing on the spatial relationships *leftOf, behind, above, below, inside, outside* and *overlaps.*

Gudivada [45] represents spatial relationships using angles among objects in an image. An image is represented as a graph, called spatial-orientation graph, where nodes are symbol of objects in the image and edges are weighted based on the angle between objects. In [44], the centroid coordinates of objects in an images are used to create the orientation angle between objects. Formally, an image $I = \{(o_i, x_i, y_i), 0 \leq i < n\}$, where x_i and y_i represent the centroid coordinates of object o_i on 2D-space. A spatial-orientation graph of an image I is an undirected weighted graph where the weight of an edge is the slope between object centroid coordinates. Figure 2.5 shows an example of an image and its corresponding spatial-orientation graph.

The similarity between images are measured based on angles differences among objects in images. The advancing of this technique is that it is invariant to rotation.



(a) An image (b) Spatial-orientation graph of the image (a)

Figure 2.5: An image and its corresponding spatial orientation graph

2.6.3 Video and moving-object retrieval

Currently, most video database systems use textual annotations for describing the content of video clips such as those described in [19, 48, 72, 91, 113]. Zhang et al. [105, 118] provide visual features such as colour and shapes of representative images for retrieval and browsing video clips in addition to textual and structural description of video clips.

In [88], Nagasaka and Tanaka used colour differences between frames to decide cuts (clips) in video. Once a cut has been decided, a representative frame from this cut is chosen as an index for the cut. Video clip retrieval is based on the colour of a specified object. A different approach to detecting clips was presented by Arman [5]. In this approach, DCT (Discrete Cosine Transform) coefficients are used.

Khokhar and Ghafoor [60] used a heterogeneous processing framework to retrieve video clips. The framework consists of image understanding and signal processing algorithms to assist content-based retrieval of video clips. There are three levels of multimedia abstraction: raw multimedia data level (the lowest level), domain representation level which includes facial, pictorial and audio domains, and knowledge-based level.

They introduced *concept* queries such as concept “slam-dunk” in basketball. A concept is represented by rules in a knowledge-base. A conceptual query is processed in two steps. Firstly, abstract data representation of video data is used to identify clips which contain intended faces, objects, and audio. Secondly, the selected video clips are analysed at frame level to determine the concept stated in the query.

Retrieving video clips using motion description and trajectories has been investigated in [24, 25, 26]. The MPEG encoding scheme is used to extract motion vectors of macroblock (a macroblock is 16×16 area). Based on motion vectors, macroblock trajectories are extracted and then used for object motion recovery. Motion in a video is retrieved using representative trajectories. The trajectory-based retrieval is conducted either exact-match based or similarity-based.

Lee [69] introduced the concept *track* with multiple class primitives. Video retrieval is conducted on a video-record basis. A video-record is a subsequence of a video sequence in which the starting frame is the frame where one or more objects appearing and the ending frame is the frame where those objects disappear.

Each object in a video sequence is described using motion primitives. There are 16 motion primitives which belong to four classes namely *translations at constant depth* (8 primitives): North(n), North-East(ne), East(e), South-East(se), South(s), South-West(sw), West(w), North-West(nw); *translation in depth* (2 primitives): Close-the-camera(close) and Away-the-camera(away); *rotation at constant depth* (2 primitives): Clockwise(cw), Counter-Clockwise(ccw); *rotation in depth* (4 primitives): Rotate-to-left(left), Rotate-to-Right(right), Rotate-upward(upward), Rotate-downward(downward). Motion primitives in a set are said to be in conflict if the set contains more than one motion primitive belonging to the same class. A motion track is a sequence of one or some combination of non-conflict motion primitives, for example ((se,close),(ne,away),s,n) is a track.

A query on video records can be done by a video identifier, by object(s) contained in video records, by track(s) or by their combinations. Lee [69] mentioned video records can be retrieved using track similarity, but proposed no similarity measure.

An approach to modelling moving objects in video sequences using an extended 2D-string called a 2D B-string was proposed by Shearer [100]. The difference between 2D B-string and the original 2D-string is that the 2D B-string uses an object extension represented by two symbols, one symbol representing the start point and the other representing the finish point along the axis [65]. Types of matching between 2D B-string were redefined from Chang's original 2D-string which involved *category* relationships (topological relationships of rectangular objects), orthogonal relationships (directional relationships in term of east-west, north-south) and exact spatial relationships

To encode moving objects in a video sequence, a sequence of 2D B-strings is proposed to account for the changes of 2D B-string relationships among objects over time. Video sequence matching is based on 2D B-string matching. Three types of limited similarity matching can be done.

A model of moving objects which enables queries that refer to future values of dynamic attributes was proposed by Sistla [101]. The model is called Moving Object Spatio-Temporal (MOST). In addition to ordinary (static) attributes, the model is supported by dynamic attributes which are represented by three sub-attribute: *value, updatetime* and *function*. The value of a dynamic attribute is determined by time (updatetime) and the function associated with it.

A database state is a mapping that associates a set of objects to its class. Each database state has an associated time stamp. A database history is an infinite sequence of database states.

A query is a predicate over the database history. There are 3 types of queries: instantaneous, continuous and persistent query. An instantaneous query is a query evaluated on history starting at the query entered. A continuous query is a sequence of instantaneous queries at certain times and is evaluated continuously until all conditions are satisfied. A persistent query is a sequence of instantaneous queries at a certain time on the infinite history.

MOST is equipped with a temporal logic based query language called Future Temporal Logic. The syntax of the language follows the syntax of SQL but is enriched by temporal and spatial predicates.

The most interesting feature of MOST is perhaps the ability to query future states. This type of query is possible since the temporal function is part of a dynamic attribute. The problem is that in many situations we do not know the temporal function.

Bimbo [10] uses spatiotemporal logic to describe and query image sequence. Spatiotemporal logic is an extension of Temporal Logic with Spatial Logic. In this logic, temporal properties are expressed as Temporal Logic assertions on spatial relationships that changes over time. Spatial relationships are expressed using Spatial Logic of symbolic projection. The following is the syntax of the logic taken from [10]:

$$\Theta := (\sigma, k) \models \theta$$

where Θ is a temporal assertion, σ is a scene sequence, k is the index of scene σ and θ is a temporal formula which is formed by combining spatial assertion Φ through the boolean connectives of negation, conjunction and the temporal *until* operator unt_t:

$$\theta := \Phi | \neg\theta | \theta_1 \wedge \theta_2 | \theta_1 \ unt_t \ \theta_2$$

A spatial assertion is a spatial relationship between centre point of objects projected on x, y and z axis.

Spatiotemporal logic is used to encode spatial and temporal relationships among objects in a scene. Scene retrieval is conducted by matching spatiotemporal logic representation of a query scene with spatiotemporal logic representation of database scenes.

Chapter 3

Modelling Images Using Spatial Information

3.1 Introduction

The work described in this chapter fit into the area of content-base image retrieval. This research aims to overcome the limitations inherent in using textual retrieval methods for image retrieval. Part of the "content" of an image can be described by its visual information such as colour, shape and texture and the spatial relationships among objects in the image. Current research on image retrieval based on spatial relationships mainly focuses on directional relationships. There are situations that the relationships between two objects in an image cannot be described using directional relationships since the nature of relationships are not directional, but topological. 2D Projection Interval Relationships (2D-PIR) were developed to fill this gap.

2D-PIR is a technique that we have developed to model an image or a picture as a network of objects and spatial relationships among them in the image. We consider both topological and directional relationships as equally important in expressing relationships among objects in an image.

To derive directional relationships, we use the projection concept from the 2D-string representation. It differs from the 2D-string [14] representation in using Allen's interval relationships [3] over the projections of the entire objects along the

x and y axes, rather than $\{<, :, =\}$. Using Allen's operators (interval relationships), directional relationships between intervals, which are the projections of objects in both the x and y axes (interval projection relationships), can be generated. Furthermore, they can also be used to capture topological relationships, especially for rectangular objects. The main advantage of this method is that it offers more information about spatial relationships between objects in a picture as compared to traditional methods [87]. In other words, it has more expressive power than traditional methods (that is it can express more spatial relationships between objects).

Using projection techniques and interval relationships, some *limited topological relationships* can be derived. We call them limited topological relationships since they are derived between bounding rectangles of objects. To represent correct and robust topological relationships we use Egenhofer's formalism [27]. We then integrate topological relationships with projection interval relationships forming 2D-PIR.

This chapter is written based on our earlier papers [82, 86, 87]. In the following sections we will discuss 2D-PIR as a pictorial data model, the interaction among topological relationships and interval relationships, picture similarity retrieval using 2D-PIR, and a pictorial query language for image retrieval.

3.2 2D-PIR

As we mention above, both directional and topological relationships can be derived by using projection techniques alone. However, topological relationships derived are not robust. Let χ and ψ be projection interval relations on x-axis and y-axis respectively and the domain of χ and ψ is the set $\{<, =, m, o, d, s, f, >, mi, oi, di, si, fi\}$ [1] which are symbols for interval relationships. Figure 3.1 illustrates this.

In Figure 3.1, the relationship between A and B according to the above scheme is $(<, oi)$, between A and C is $(<, m)$ and between B and C is (fi, si). Let us consider these relationships. What kind of information do we get when the 2D-PIR

[1] They are symbols for *before, equals, meets, overlaps, during, starts, finishes, after, meetsInverse, duringInverse, startsInverse* and *finishesInverse* respectively

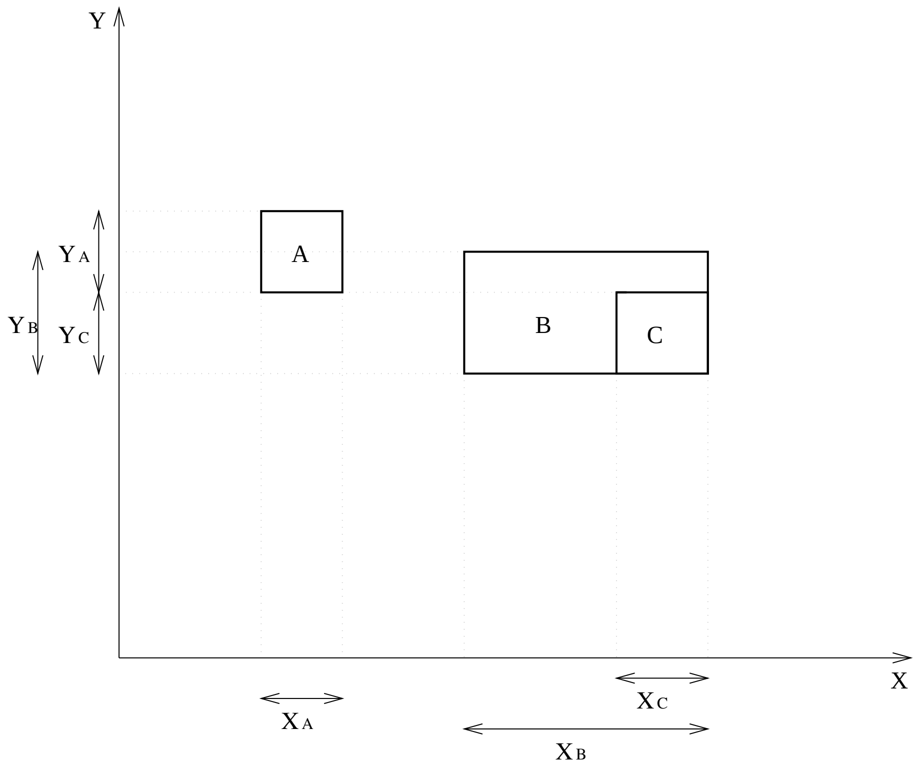


Figure 3.1: Interval relationships on rectangular objects

for A and B is $(<, oi)$? First, it tells us that A and B are *disjoint* and A is *leftOf* B since $x_A < x_B$. Secondly, it informs us that *part of* A is *above* B since $y_A \ oi \ y_B$ or $y_B \ o \ y_A$. Now, we turn to B and C. The 2D-PIR between B and C is (fi, si). The information we can get from this relationship is: 1) B *covers* C (or C *coveredBy* B), 2) C is located on the bottom-right of B. Thus, we can infer that C is covered by B on the bottom-right.

The above examples show only rectangular objects whose boundaries are parallel to the x and y axes. The shape of real world objects is rarely so simple or regular. Consider Figure 3.2. In this figure, the relationship between A and B is (di, di). The relationship that would be inferred is B *inside* A (A *contains* B) which is incorrect.

To solve this problem we introduce topological relationships. The new relationship is called 2D-PIR. 2D-PIR is defined as a triple (δ, χ, ψ) where δ is a topological relationship from the set $\{dt, to, ct, in, ov, co, eq, cb\}$[2]. The relationship between A

[2]These represent the topological relationships *disjoint, meets, contains, insides, overlaps, covers, equals, coveredBy*

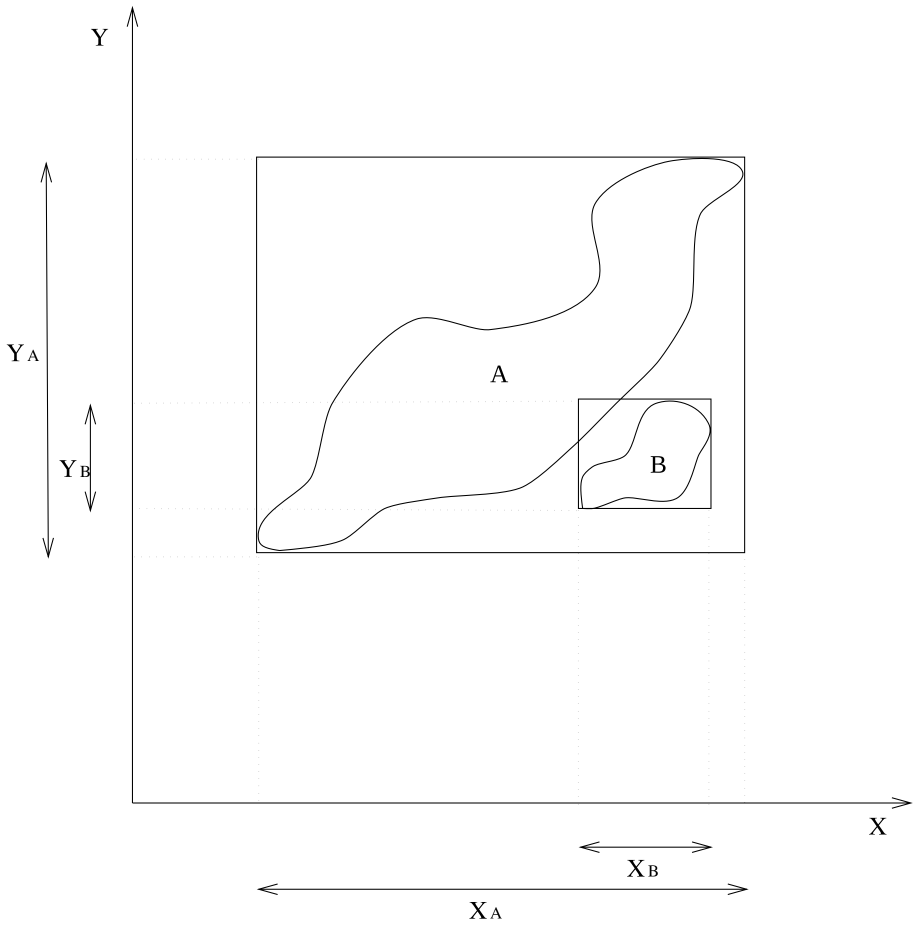


Figure 3.2: Interval relationships on non-rectangular objects

and B in Figure 3.2 will be (dt, di, di) which indicates that A and B are *disjoint* objects and B is *surroundedBy* [3]. The triple (dt, di, di) can be interpreted as the *surroundedBy* relationship, since $\chi = di$ and $\psi = di$ implies that there exists *parts of* A which are *right, left, above* as well as *below* B.

3.3 Interaction between 2D-PIR Components

The above section discusses interval relationships and shows how we arrive at 2D-PIR, a robust representation that integrates directional and topological relationships. In this section we will explore the interaction between the topological component (δ) and the interval components (χ and ψ).

[3]A surroundedBy B if the 2D-PIR between A and B in (x, di, di) where $x \in \{dt, to\}$

There are two interesting aspects of this interaction. First, we can represent 2D-PIR in more compact way since all redundancies can be removed. Second, a simpler algorithm for computing topological relationships and 2D-PIR on regions can be developed.

Suppose A and B are two spatial objects. If B is inside A then the bounding rectangle of B must be inside the bounding rectangle of A which means that the projection B on x-axis and y-axis must be *during* the projection of A on x-axis and y-axis respectively. However the reverse is not necessarily true.

Based on the interaction between topological and 2D interval relationships, we can extract five basic topological relationships in term of 2D-PIR namely *disjoint (dt), touches (to), inclusion (ic), equals (eq)* and *overlaps (ov).* Other topological relationships can be implied from the *ic* relationship and interval relationships (χ and ψ). The disjoint relationship (*dt*) is a basic topological relationship since this relationship cannot be implied from any topological relationship combined with interval relationship χ or ψ. Disjoint may exist for any combination of χ and ψ. Yet, the interval relationships *before* and *after* imply *disjoint.* The latter property is quite useful in deciding whether two regions (polygons) are disjoint. If the projection of polygons (on x-axis or y-axis) has relationship *before* ($<$) or *after* ($>$) then the two polygons are disjoint. However we can not conclude that the relationship between polygons is not *disjoint* if their projection (either on x-axis or y-axis) is not *before* or *after* as stated above. Figure 3.3 is an example where two polygons are disjoint but their χ and $\psi \notin \{<, >\}$.

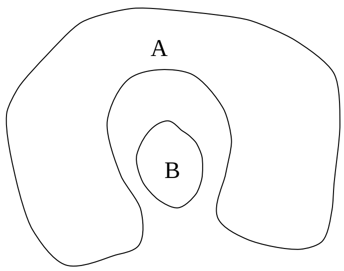


Figure 3.3: A and B are disjoint

Similarly, *touches* is a basic topological relationship in term of 2D-PIR. *Touches* relationship may exist for any combination of χ and ψ. The interval relationship *meet* (m) and *meetsInverse (mi)* does not imply *touches* relationships and *touches* does not imply *meet* and *meetsInverse* either.

The next basic topological relationship is *equals (eq)*. Even though both χ and ψ are *equals* does not necessarily imply topological relationships *equals*. However, topological relationship *equals* always imply *equals* relationship on χ and ψ.

The new basic relationship is *inclusion* (ic). Two spatial objects are *inclusion* if the interior of an object is subset of the other. We can deduce other topological relationships (inside, contain, covers, coveredBy) using ic and some combination of χ and ψ.

Another basic relationship is *overlaps*. The reason for including *overlaps* as a basic topological relationship in term of 2D-PIR is that this relationship cannot always be implied from the combination of ic, χ and ψ as depicted in Figure 3.4. In this figure, χ and ψ between A and B in (a) and (b) is the same. Topological relationship between A and B in (b) is overlap which clearly cannot be deduced from ic combine with χ and ψ. In conclusion, the domain of δ in 2D-PIR: (δ, χ, ψ) is $\{dt, to, ic, eq, ov\}$.

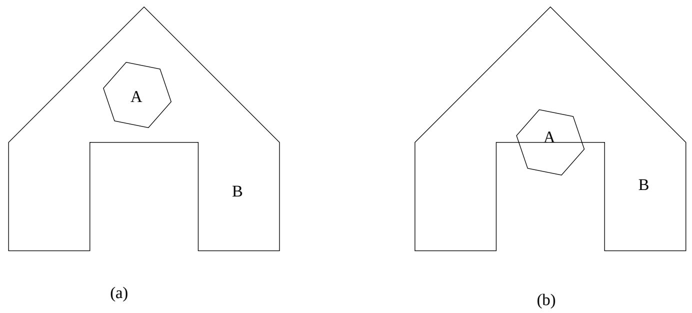


Figure 3.4: (a) A inside B (b) A overlap B

Our basic topological relationships above give a strong foundation for Clementini [21] to generalise topological relationships into five general relationships.

As we mentioned above other topological relationships can be implied from basic topological relationships combined with interval relationships (χ and ψ). The following rules deduce the rest of the topological relationships (*covers, coveredBy, inside* and *contains*) by combining the relationship ic with χ and ψ.

1. Rule #1: if $\delta = ic \ \wedge \ \chi \in \{s, f\} \wedge \psi \in \{s, f, d, =\}$ then the topological relationship is *coveredBy*.

2. Rule #2: if $\delta = ic \ \wedge \ \chi \in \{s, f, d, =\} \wedge \psi \in \{s, f\}$ then the topological relationships is *coveredBy*.

3. Rule #3: if $\delta = ic \ \wedge \ \chi \in \{si, fi\} \wedge \psi \in \{si, fi, di, =\}$ then the topological relationship is *covers*.

4. Rule #4: if $\delta = ic \ \wedge \ \chi \in \{si, fi, di, =\} \wedge \psi \in \{si, fi\}$ then the topological relationship is *covers*.

5. Rule #5: if $\delta = ic \ \wedge \ \chi = d \wedge \psi = d$ then the topological relationship is *inside*

6. Rule #6: if $\delta = ic \ \wedge \ \chi = di \wedge \psi = di$ then the topological relationship is *contains*

3.4 A Data Model for Pictorial Databases

2D-PIR provides the basis for our symbolic model for pictures. As mentioned previously we choose a directed graph to model a picture based on 2D-PIR relationships among objects in the picture. In other words, our symbolic model for a picture is a graph of objects and their 2D-PIR relationships.

Consider Figure 3.5 which depicts the scene "A car was hit by a truck in Rainbow St. just on the right side of a tree". In Figure 3.6 spatial relationships encoded in 2D-PIR are: (dt, o, d) between the tree and Rainbow St., (dt, o, oi) between the tree and the sedan, (dt, b, di) between the tree and the truck, (ov, di, di) between Rainbow St. and the sedan, (ct, di, di) between Rainbow St. and the truck and (to, m, o) between the sedan and the truck.

Note that 2D-PIR requires a directed graph to model a picture, as opposed to the strings used in the 2D-string representation, since some of interval and topological relationships are not transitive.

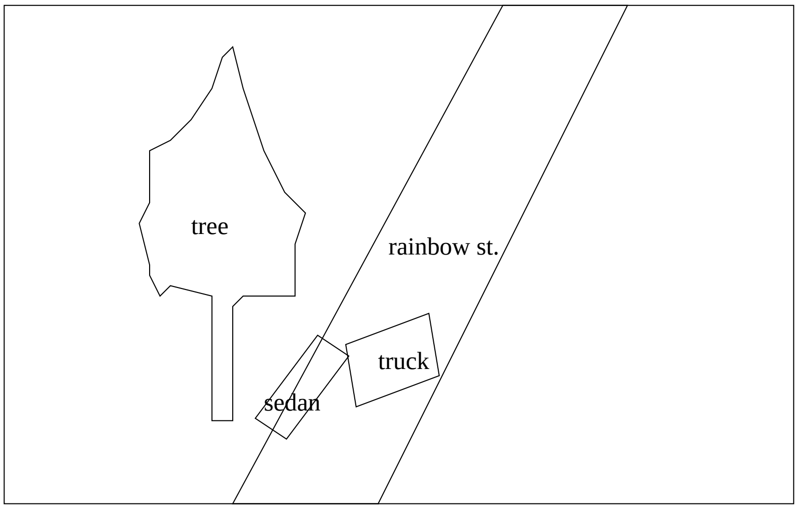


Figure 3.5: A car was hit by a truck in Rainbow St.

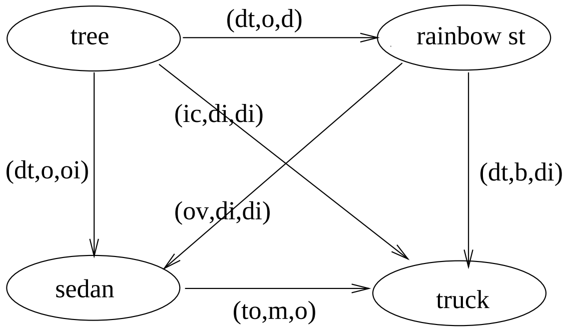


Figure 3.6: A 2D-PIR of Figure 3.5

Definition 7 *A 2D-PIR graph is a connected labelled 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 labelled by 2D-PIR relationships.*

Definition 8 *A symbolic picture model is a 2D-PIR graph.*

An example of 2D-PIR graph is shown in Figure 3.6.

The graph-based model above abstracts away many details from actual image and retains spatial relationships among objects in the image. We need to construct graph carefully, otherwise we may only be able to use comparison algorithms that are very inefficient.

3.5 Picture Retrieval Using 2D-PIR

3.5.1 Graph construction

Since a picture is represented symbolically by a graph then to decide whether or not two pictures are matched, it is necessary to compare the corresponding graph representations. The fundamental equality relationship between graphs is known as *graph isomorphism*. Two graphs G_1 and G_2 are *isomorphic* if there exists a 1-1 and onto function $f : V(G_1) \rightarrow V(G_2)$ such that $xy \in E(G_1)$ if, and only if $f(x)f(y) \in E(G_2)$. Detecting graph isomorphism in the general case is non polynomial (NP) [41]. In addition, detecting sub-graph isomorphism is known to be NP-Complete [110]. However, if we construct our 2D-PIR graphs carefully, we can devise efficient processing algorithms for them. The basic idea is to sort objects in the graph. Our strategy here is to sort objects based on their position in the picture plane.

The following simple algorithm chooses a fixed topology of 2D-PIR graph in order to find a polynomial algorithm for 2D-PIR graph matching.

Algorithm: Graph construction
input: A pre-processed picture (all objects have been recognised and all topological and projection relationships have been computed)
output: A directed graph (adjacency list) representing the picture
begin
(1) Arrange objects from left to right and top to bottom (that is objects in the picture are sorted based on their position left-right and then top-bottom priority.
(2) Let P be the ordered set of objects in a picture based on the arrangement in (1)
(3) **for** each object O_i in P
(4) create a linked-list of 2D-PIR relationships with all objects that follow O_i in the arrangement from (1).
(5) **endfor**
end.

The time complexity of this algorithm is easily analysed; it is $O(e)$ where e is the number of edges in the graph which is always $(n^2 - n)/2$ (n is the number of objects in the picture). Thus, the complexity of the algorithm in terms of objects is $O(n^2)$.

3.5.2 Picture matching

In this sub-section we will introduce a picture matching algorithm based on exact matching between their corresponding 2D-PIR graphs. In the subsequence sub-sections we will develop a similarity measure based on 2D-PIR and modified the algorithm for similarity matching.

Algorithm: Picture matching
input: two 2D-PIR graphs (G_1 and G_2) representing pictures P_1 and P_2.
output: integer 0 : no-match
integer 1: P_1 is a sub-picture of P_2
integer 2: P_1 is equivalent of P_2

begin
(1) if $V(G_1) = V(G_2)$ then
(2) for $v_i \in V(G_1)$
(3) if linked-list of $v_i \in V(G_1) =$ linked-list $v_i \in V(G_2)$
(4) continue
(5) else return 0 /*no match*/
(6) endif
(7) return 2 /* P_1 is equivalent with P_2 */
(8) endfor
(9) elseif $V(G_1) \subset V(G_2)$ then
(10) for $v_i \in V(G_1)$
(11) find $v_j \in V(G_2)$ such that $v_i = v_j$
(12) if each element of linked-list v_i is also element of linked-list v_j
(13) continue

```
(14)          else return 0 /*no match*/
(15)          endif
(16)          return 1 /*P1 is sub-picture of P2*/
(17)      endfor
(18)  else return 0 /*no match*/
(19)  endif
end.
```

A query picture in the above algorithm is a picture or a picture sketch which is used by a user to pose a query. The above algorithm assumes that a query picture contains at most the same number of objects as the number of objects in the intended picture stored in a database. We will relax this assumption when we deal with similarity matching.

This algorithm is dominated by lines 2 to 8 or line 10 to 17, which require $(n^2 - n)/2$ comparisons – that is, $O(n^2)$ in the worst case. Note that n is the number of objects in P_1. It is efficient compared to 2D-string matching algorithm which is $O(M) + O(N^2 * lp^3)$. Note that both M and N in the 2D-string matching algorithm are the number of entries in matching tables which are greater than the number of objects in a picture (since the 2D-string method uses partition) and lp is the maximum length of matching tables. Let us look at some examples and demonstrate how the algorithm works.

Example 1. shown in Figure 3.7 has two pictures: a query picture (QP) and a database picture (DP).

Figure 3.8 contains graphs (symbolic picture models), represented as adjacency-lists, for QP and DP constructed by the graph construction algorithm.

These graphs are inputs for the picture matching algorithm. The execution of the algorithm is as follows: since the nodes of QP are a subset of the nodes of DP ($V(QP) \subset V(DP)$), the algorithm executes lines 10 to 17. The linked-list comparison between the same element (node) of QP and DP (line 16) is always true, so the algorithm will return 1 (QP is a sub-picture of DP).

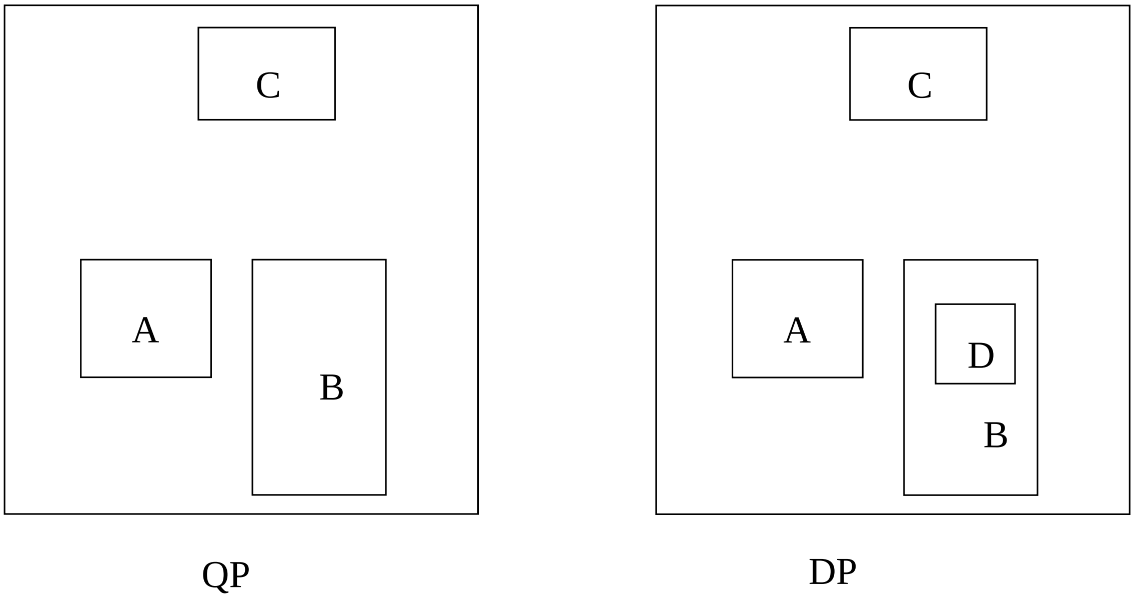


Figure 3.7: Example 1: picture matching

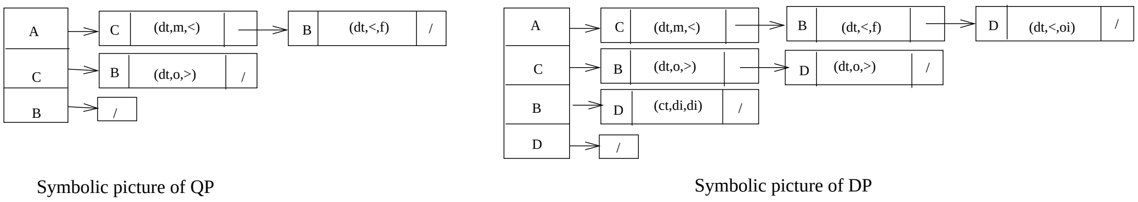


Figure 3.8: Symbolic pictures of Figure 3.7

Example 2. Replace the symbol C in QP of Example 1 with another symbol, e.g. M. The picture matching algorithm will quickly detect that QP does not match with DP (line 18).

Example 3. Figure 3.9 shows another example of matching between a query picture and a database picture.

Figure 3.10 contains graphs (symbolic pictures) represented as adjacency-list for QP and DP constructed by the graph construction algorithm. In this case the picture matching algorithm will return 0 (no match), since the result of line 12 for node A is false.

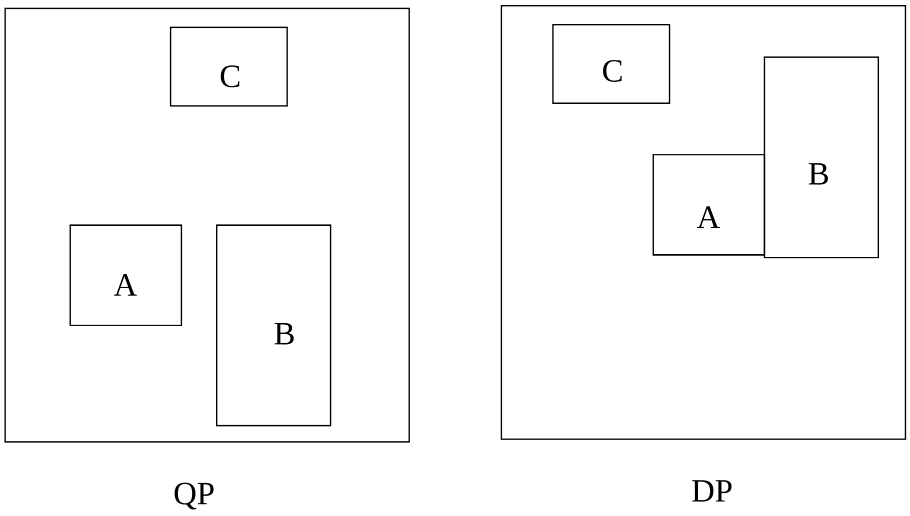


Figure 3.9: Example 3: picture matching

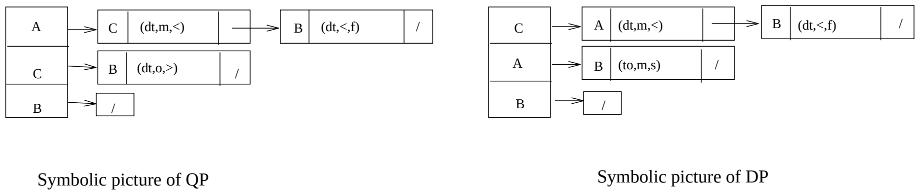


Figure 3.10: 2D-PIR graphs for QP and DP in Figure 3.9

3.5.3 Similarity-based matching

Exact matching is probably not very useful for image retrieval systems, similarity retrieval is preferred. The core problem in similarity retrieval is determining a similarity metric that is both efficient to compute and captures the essential aspects of similarity that humans use. By quantifying the concept "*degree of similarity*", we can then perform useful retrieval operations such as thresholding and ranking.

The degree of similarity between two pictures represented using the 2D-PIR graph is dependent on the degree of similarity between their corresponding 2D-PIR relationships. The degree of similarity between 2D-PIR relationships, in turn, is dependent on the degree of similarity between components of 2D-PIR. 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. As a result, it is necessary to construct similarity metrics both for topological relationships (δ) as well as for interval relationships (χ and ψ).

Topological relationships distance

The distance between δ_1 and δ_2 (that is $D(\delta_1, \delta_2)$) has been defined by Egenhofer and Al-Taha [29]. It is derived from the difference of elements of *9-intersection* topological relationships encoding scheme.

Let $R^0, \partial R$, and $\bar{R}$ denote *interior, boundary* and *exterior* of pointset R. Let A and B be two pointsets, topological relationships (δ) are encoded using a matrix 3×3, *R-3*, called the *9-intersection.*

$$\delta = \begin{pmatrix} A^0 \cap B^0 & A^0 \cap \partial B & A^0 \cap \bar{B} \\ \partial A \cap B^0 & \partial A \cap \partial B & \partial A \cap \bar{B} \\ \bar{A} \cap B^0 & \bar{A} \cap \partial B & \bar{A} \cap \bar{B} \end{pmatrix}$$

By assigning 1 if the intersection is not empty and 0 otherwise, encoding of topological relationships on *9-intersection* is established. The following examples are encoding of *touches (to)* and *overlap (ov)* on *9-intersection.*

$$to = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad ov = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

The distance between interval relationships is the sum of differences between corresponding entries (absolute value) on their *9-intersection* representations. For example, $D(to, ov) = 3$. The proof for this distance correctness can be found in [29]. Table 3.1 contains topological distances between the eight topological relationships.

D	dt	to	eq	ic	ov
dt	0	1	6	5	4
to	1	0	5	5	3
eq	6	5	0	4	6
ic	4	5	4	0	4
ov	4	3	6	4	0

Table 3.1: Topological distances among basic topological relationships

Note that the distance between *ic* with another relationship is the maximum distance between the corresponding relationships with in, ct, co and cb.

Interval relationships distance

The distance between two interval (projection) relationships χ_1 and χ_2 or ψ_1 and ψ_2 is derived from similar method that has been defined above.

Let $I^0, \partial I_s,\ \partial I_f$ and $\bar{I}$ denote *interior, start boundary, finish boundary* and *exterior* of interval I. Let A and B be two intervals, interval relationships are encoded using a matrix 4×4, R, called the *16-intersection.*

$$W = \begin{pmatrix} A^0 \cap B^0 & A^0 \cap \partial B_s & A^0 \cap \partial B_f & A^0 \cap \bar{B} \\ \partial A_s \cap B^0 & \partial A_s \cap \partial B_s & \partial A_s \cap \partial B_f & \partial A_s \cap \bar{B} \\ \partial A_f \cap B^0 & \partial A_f \cap \partial B_s & \partial A_f \cap \partial B_f & \partial A_f \cap \bar{B} \\ \bar{A} \cap B^0 & \bar{A} \cap \partial B_s & \bar{A} \cap \partial B_f & \bar{A} \cap \bar{B} \end{pmatrix}$$

By assigning 1 if the intersection is not empty and 0 otherwise, encoding of interval relationships on *16-intersection* is set up. The following examples are encoding of *meet (m)* and *overlap (o)* on *16-intersection.*

$$m = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \quad o = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

The distance between interval relationships is sum of differences between corresponding entries (absolute value) on their *16-intersection* representation. Formally, let $D(R_a, R_b)$ be distance between interval relationships R_a and R_b.

$$D(R_a, R_b) = \sum_{i=0}^{4} \sum_{j=0}^{4} |R_a[i,j] - R_b[i,j]|$$

For example, $D(m, o) = 4$. Note that *16-intersection* encoding above cannot differentiate the interval relationships *before* and *after.* However, by introducing a third interval called *delay interval*, denoted by Δ, we can derive *before* $(<)$ and *after* $(>)$ using *meets (m)* or *meet-inverse (mi)* .

$$A \ < \ B = A \ m \ \Delta \ m \ B$$

$$A \ > \ B = A \ mi \ \Delta \ mi \ B$$

The distance between $<$ and $>$ is $2 \times D(m, mi)$. Based on *16-intersection* encoding above, the distance between m or mi with $<$ or $>$ is 3. The distance between *before* with other interval relationships is approached by *before* with *meet* plus the distance *meet* with other interval relationships, or $D(<, r) = 3 + D(m, r)$ where r is interval relationships other than *before*. Similarly the distance between *after* with other interval relationships is approached by *after* with *meet-inverse* plus the distance *meet-inverse* with other interval relationships, or $D(>, r) = 3 + D(mi, r)$ where r is interval relationships other than *after*.

The above distance definition fulfils all requirements for a distance function. Proof of this claim is as follows.

We have to show that the distance defined above fulfils all distance theorems (always greater or equal to zero, symmetric and triangle inequality).

Theorem 1 *For two intervals, the interval distance is between 0 and 12. If The distance between two interval relationships is 0 if and only if they are the same interval relationship.*

Proof: Every 16-intersection has 16 entries. If all entries between two 16-intersection that represent interval relationships are the same than the distance between them must be 0. But if two 16-intersection has the same value on all entries implies they are the same interval relationship. On the other hand if the there are differences among entries in 16-intersection, the distance must be greater than 0. The maximum distance between two 16-intersection using entries differences above is achieved between *during* (d) and *during inverse* (see Table 3.2). However by definition $D(<, >) = 2 \times D(m, mi)$, $D(<, r) = 3 + D(m, r)$ and $D(>, r) = 3 + D(mi, r)$ (note that r is an interval relationship other that $<$ or $>$. Maximum $D(m, r)$ or $D(mi, r)$ is 8, so the maximum $D(<, r)$ or $D(>, r)$ is 11. But $D(<, >) = 2 \times D(m, mi)$, is 2×6, which is 12. Therefore the maximum distance between two interval relationships is 12. $\square$

Theorem 2 *The interval distance is symmetric, that is if R_a and R_b are interval relationships then $D(R_a, R_b) = D(R_b, R_a)$.*

Proof:

$$D(R_a, R_b) = \sum_{i=0}^{4}\sum_{j=0}^{4} |R_a[i,j] - R_b[i,j]|$$

and

$$D(R_b, R_a) = \sum_{i=0}^{4}\sum_{j=0}^{4} |R_b[i,j] - R_a[i,j]|$$

For each entries it is true that $|R_a[i,j] - R_b[i,j]| = | - R_a[i,j] - R_b[i,j]| = |R_b[i,j] - R_a[i,j]|$

consequently,

$$\sum_{i=0}^{4}\sum_{j=0}^{4} |R_a[i,j] - R_b[i,j]| = \sum_{i=0}^{4}\sum_{j=0}^{4} |R_b[i,j] - R_a[i,j]|$$

which implies $D(R_a, R_b) = D(R_b, R_a)$. $\square$

Theorem 3 *The sum of interval distance between interval relationships R_a, R_b and R_b, R_c is greater than or equal to the distance between interval relationships R_a and R_b. Formally, $D(R_a, R_b) + D(R_b, R_c) \geq D(R_a, R_c)$*

Proof: To prove this theorem, first we will show that every entry in the 16-intersection, that is every intersection, follows the the triangle inequality theorem above, and the we will show that the entire 16-intersection also holds this theorem.

The difference between two intersection is either 1 or 0. Let $|R_a[i,j] - R_c[i,j]| = 0$. Thus, $R_c[i,j]$ is either equal to or different from both $R_a[i,j]$ and $R_c[i,j]$. If it is equal then $|R_a[i,j] - R_b[i,j]| + |R_b[i,j] - R_c[i,j]| = 0$. Otherwise, $|R_a[i,j] - R_b[i,j]| + |R_b[i,j] - R_c[i,j]| = 2$. As a result, the triangle inequality holds for $|R_a[i,j] - R_c[i,j]| = 0$.

Now, Let $|R_a[i,j] - R_c[i,j]| = 1$. Thus, $R_b[i,j]$ is either equal to $R_a[i,j]$ and different both $R_c[i,j]$ or vice versa. Therefore, $|R_a[i,j] - R_b[i,j]| + |R_b[i,j] - R_c[i,j]| = 1$ which is equal to the difference between two intersections. Therefore, the triangle inequality also holds for $|R_a[i,j] - R_c[i,j]| = 1$. Consequently, the following triangle inequality is true.

$$|R_a[i,j] - R_b[i,j]| + |R_b[i,j] - R_c[i,j]| \geq |R_a[i,j] - R_c[i,j]|$$

therefore,

$$\sum_{i=0}^{4}\sum_{j=0}^{4} |R_a[i,j] - R_b[i,j]| + \sum_{i=0}^{4}\sum_{j=0}^{4} |R_b[i,j] - R_c[i,j]| \geq \sum_{i=0}^{4}\sum_{j=0}^{4} |R_a[i,j] - R_c[i,j]|$$

which implies $D(R_a, R_b) + D(R_b, R_c) \geq D(R_a, R_c)$. $\square$

Table 3.2 gives the distance between the 13 interval relationships, derived from the above method.

D	$<$	m	o	s	d	f	=	oi	si	di	fi	mi	$>$
$<$	0	3	7	9	10	11	11	11	11	10	9	9	12
m	3	0	4	6	7	8	8	8	8	8	7	6	6
o	7	4	0	4	5	8	8	8	8	5	4	8	11
s	9	6	4	0	3	6	4	8	6	9	8	8	11
d	10	7	5	3	0	3	7	5	9	10	9	7	10
f	11	8	8	6	3	0	4	4	8	9	6	6	9
=	11	8	8	4	7	4	0	8	4	7	4	8	11
oi	11	8	8	8	5	4	8	0	4	5	8	4	7
si	11	8	8	6	9	8	4	4	0	3	6	6	9
di	10	7	5	9	10	9	7	5	3	0	3	7	10
fi	9	6	4	8	9	6	4	8	6	3	0	8	11
mi	9	6	8	8	7	6	8	4	6	7	8	0	11
$>$	12	9	11	11	10	9	11	7	9	10	11	3	0

Table 3.2: Interval distances among 13 interval relationships

Similarity function

Based on the distance measure for topological relationships and the distance measure for interval relationships, we can now define the distance metric for 2D-PIR relationships. The distance between two 2D-PIR relationships, r_1 and r_2 is defined using the following distance formula:

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

This is used to measure the distance between two symbolic pictures. Our objective is to measure similarity between two pictures. First of all, we need to define a function that computes the similarity value between two 2D-PIR relationships. Our approach is to use a normalised similarity function derived from the distance function. The maximum similarity value between two 2D-PIR relationships is 100 when their distance is 0. On the other hand, the minimum similarity value between two 2D-PIR relationships is 0 when their distance is the maximum distance that can be reached. The maximum distance between two 2D-PIR can be achieved if the distance among their components is also maximum. The maximum distance between two topological relationships is 6 (see Table 3.1) and the maximum distance between two interval relationships is 12 (see Table 3.2). Thus, based on the distance formula between two 2D-PIR relationships above, the maximum distance between 2D-PIR relationships is 20.785. Let r_1 and r_2 be two 2D-PIR relationships. The similarity between r_1 and r_2, denoted by $\varsigma(r_1, r_2)$ is defined using the following formulae:

$$\varsigma(r_1, r_2) = 100.0 \times ((20.785 - D(r_1, r_2))/20.785)$$

Recall that a symbolic picture is a connected labelled digraph $G(V, R)$ where V is a finite non-empty set of symbols representing the objects in a picture and R is a set of edges labelled by 2D-PIR relationships between the objects. Let $G_a(V_a, R_a)$ and $G_b(V_b, R_b)$ be two symbolic pictures and $V_a \cap V_b$. The similarity between G_a and G_b is the sum of the similarity between corresponding 2D-PIR in R_a and R_b

divided by minimum number of relationships in R_a and R_b . More formally, if $R_a = \{r_{1a}, r_{2a}, \cdots, r_{na}\}$ and $R_b = \{r_{1b}, r_{2b}, \cdots, r_{nb}, \cdots\}$ then the similarity between G_a and G_b is defined by the following formula:

$$\varsigma(G_a, G_b) = \sum_{i=1}^{k} \varsigma(r_{ia}, r_{ib})/n$$

where k is the cardinality of $(V_a \cap V_b)$

Figure 3.11 shows four pictures. The problem is to find which picture among P2, P3, and P4 is most similar to P1. Visual inspection suggests that P1 is most similar to P2, and less similar to P3 and P4.

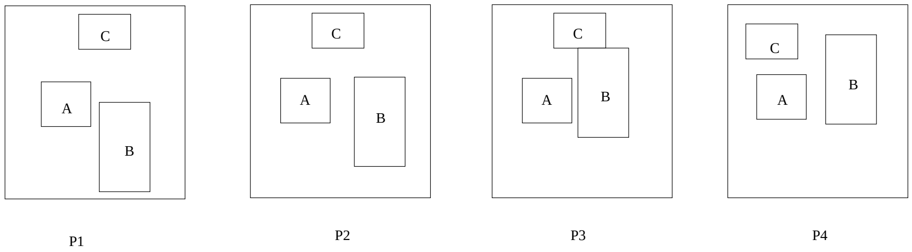


Figure 3.11: Some pictures

According to the formulae developed above, the similarity value between P1 and P2 is 92.59, between P1 and P3 is 84.88, P1 and P4 is 57.59, between P2 and P3 is 89.40, between P2 and P4 is 63.15 and between P3 and P4 is 71.99. Thus, indeed P1 is more similar to P2 than P1 to P3 or P1 to P4.

3.5.4 Similarity matching algorithm

We have described a distance function to measure the similarity of two pictures represented as 2D-PIR graphs. Two pictures are considered similar if their similarity value is greater than a threshold value [4]. We can say that the larger the similarity value between two pictures the more similar they are. If the similarity value between two pictures is 100 then the pictures are equivalent (their corresponding spatial relationships are equals)

[4] A threshold value between 0 to 100 that can be adjusted by a user. A default threshold value will be determined by experiments

The following algorithm computes the similarity value between two pictures represented by 2D-PIR graph. The 2D-PIR representation can retrieve pictures based on sub-picture match (exact and similar).

Algorithm: Picture similarity matching
input: 2D-PIR graphs ($G_1(V_1, R_1)$ and $G_2(V_2, R_2)$) representing pictures P_1 and P_2.
output: the distance between P_1 and P_2
begin
(1) set $S(similarity)$ to 0 /* set the similarity between two pictures to 0 */
(2) **if** $V_1 \cap V_2$ then
(3)   **for** each $v_i \in V(G_1)$
(4)   **for** each node n in linked-list v_i
(5)   find the corresponding node m in linked-list v_j of $V(G_2)$
                such that $v_i = v_j$ and $n = m$ or $v_i = m$ and $n = v_j$
(6)   compute the similarity value and add it to S
(7)   **endfor**
(8)   **endfor**
(9)   return (S/minimum cardinality(R_1, R_2))
(10) **else** return 0 /* different pictures */
end.

Basically, the above algorithm compares the corresponding edges in two symbolic pictures. In the worst case there are as many as $(n^2 - n)/2$ edge comparisons (note that n is the number of objects in the picture). So, the worst case complexity of the algorithm is $O(n^2)$ where n is the number of objects in G_1 or G_2 (the smaller one). For similarity matching based on 2D-strings some new proposed algorithms such as in [15] and [70] are NP in the worst case. So, the above algorithm is not only simpler than the 2D-string matching algorithm, but is also more efficient than any currently known 2D-string matching algorithm.

3.5.5 The effect of rotation and reflection

So far we have discussed similarity retrieval based on the 2D-PIR representation, but ignored the effect of rotation and reflection on similarity retrieval of pictures. Consequently, the picture similarity matching algorithm presented above is not invariant for matching a picture with another picture where the latter picture is, in fact, a rotated version or reflected version of the first picture.

There are two approaches to solve this problem: re-projection and rotation transformation. Re-projection means re-positioning a picture (from the database) that is a potential match based on a chosen reference object. For example, if the two pictures P_1 and P_2 contain the same objects, then choose any object in P_2 as a reference. The next step is to do a projection of other objects based on the the angle of the minimum bounding rectangle of the reference object both for P_2 and P_1 and then construct 2D-PIR relationships on P_1 and P_2. Note that we need only to compute projection relationships, since topological relationships are invariant under rotation. The rotation transformation approach takes care of the rotation effect of 2D-PIR relationships of a picture.

The re-projection approach is good for checking rotation which is not 90^0 or a multiple of it (such as 180^0 or 270^0). If re-projection is employed to rotations that are 90^0 apart, the result will be 2D-PIR relationships on the same coordinate configuration. On the other hand, the rotation transformation approach is good for these rotations. The rotation transformation is also good for checking a reflected version (both for vertical and horizontal reflection) of a picture. Thus the two approaches must be integrated to check for rotated or reflected versions of a picture.

The rotation transformation has an interesting theoretical aspect. As we mentioned above, this approach is effective for $90^0, 180^0$ and 270^0 rotations, since these rotations yield unique transformations of 2D-PIRs. We will also discuss how 2D-PIRs are affected by reflection.

Suppose A and B are two objects in a picture (P). Let (T, R_1, R_2) be a 2D-PIR relationship between A and B in P where T is the topological relationship component, R_1 and R_2 are interval relationships created by projecting A and B on the x-axis and y-axis respectively. Let ξ be a function such that

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

The following statement holds: If (T, R_1, R_2) is a 2D-PIR relationship between two spatial objects in a picture then $(T, \xi(R_2), R_1)$ is the 2D-PIR relationship between objects in the 90^0 counter-clockwise rotated version of the picture.

For any rotation of a picture which is a multiple of 90^0 counter-clockwise rotation, the interval relationships can be computed as a composition of 90^0 rotations. Suppose $\Re^n$ is a rotation function over 2D-PIR relationships where n is an integer such that $n \ MOD \ 90 \ = \ 0$. $\Re^0(T, R_1, R_2) = (T, R_1, R_2)$, $\Re^1(T, R_1, R_2) = (T, \xi(R_2), R_1)$, $\Re^2(T, R_1, R_2) = \Re^1(\Re^1(T, R_1, R_2)) = \Re^1(T, \xi(R_2), R_1) = (T, \xi(R_1), \xi(R_2))$.
$\Re^3(T, R_1, R_2)\Re^1(Re^2(T, R_1, R_2) = \Re^1(T, \xi(R_1), \xi(R_2)) = (T, R_2, \xi(R_1))$.
In general $\Re^n(T, R_1, R_2 \ = \ \Re^1(\Re^{n-1}(T, R_1, R_2))$ for $n \geq 1$.

Figure 3.12 shows a picture (P) and a version that has undergone a 270^0 counter-clockwise rotation. The 2D-PIR relationships between X Y, X Z and Y Z in P are $(dt, di, <)$, (ov, o, di) and $(dt, <, >)$ respectively. If we transform these relationships via a 270^0 rotation transformation (that is, $\Re^3(T, R_1, R_2)$) the 2D-PIR relationships are $(dt, <, di)$, (ov, di, oi) and $(dt, >, >)$ which , in fact, are 2D-PIR relationships between X Y, X Z and Y Z in Q respectively.

With respect to reflection, the following statement holds: if (T, R_1, R_2) is a 2D-PIR relationship between two spatial objects in a picture then $(T, (R_1), \xi(R_2))$ is the 2D-PIR relationship between the objects in a version of the picture that has been reflected in the x-axis. $(T, \xi(R_1), R_2)$ is the 2D-PIR relationship between objects in version of the same picture reflected in the y-axis.

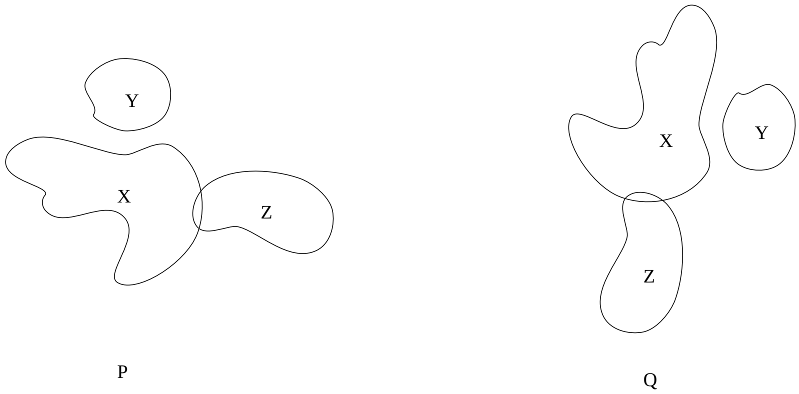


Figure 3.12: A picture (P) and its 270^0 rotated version (Q).

Figure 3.13 shows a picture (P) and its vertical (Pv) and horizontal (Ph) reflected versions. The 2D-PIR relationships between X and Y, X and Z and Y and Z in P are the same as the previous example. If we transform these relationships by vertical reflection, the 2D-PIR relationships are $(dt, di, <), (ov, oi, di)$ and $(dt, >, >)$, which are 2D-PIR relationships between X and Y, X and Z and Y and Z in Pv respectively. Similarly, we transform these relationships by horizontal reflection, the 2D-PIR relationships are$(dt, di, >), (ov, o, di)$ and $(dt, <, <)$ which are 2D-PIR relationships between X and Y, X and Z and Y and Z in Ph respectively.

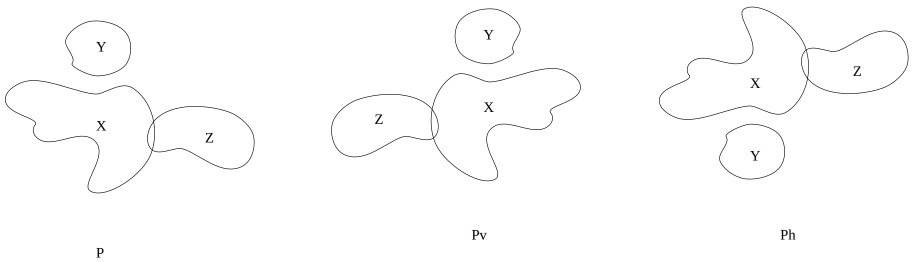


Figure 3.13: A picture (P) and its vertical reflection (Pv) and horizontal reflection (Ph).

3.6 Summary

2D-PIR is a unified representation of spatial relationships that integrates topological and directional relationships. 2D-PIR can be used to model an image as a "symbolic picture". A symbolic picture is a digraph where nodes are symbols represent objects and edges are labelled by 2D-PIR relationships.

To support similarity retrieval we defined a distance measure between 2D-PIR relationships which is the Euclidean distance among components of a 2D-PIR. The distance between two images is the summation of distance among corresponding 2D-PIRs.

Since the 2D-PIR graph is carefully constructed, the similarity algorithm, which is based on this graph, successfully avoids the NP problem, especially if symbols are unique. It is efficient when compared to the general graph isomorphism algorithm which needs $n^2n!$ comparisons [41] (note that in this case n is the number of nodes). It is also better than currently known 2D-string similarity matching algorithm.

As compare to the E-R picture [117], the 2D-PIR relationship is deduced automatically from a picture after the objects in the picture have been identified.

Chapter 4

A Prototype Image Retrieval System Using 2D-PIR

4.1 Introduction

A prototype of an image database system that implements the 2D-PIR structure and the similarity algorithm has been built using the Java language.[1] Our main object in building this prototype is to show that 2D-PIR is a good representation for retrieving similar images and is simple to implement.

Based on the prototype we conduct experiments on several collections of images. We compare the effectiveness of 2D-PIR similarity method with Gudivada's similarity method. We select Gudivada's similarity method for comparison since this method performs better than 2D-String similarity method. In addition, Gudivada uses a graph to represent a symbolic image which is similar to 2D-PIR representation. We also compare our model with other related models.

The result of our experiments show that 2D-PIR similarity matching is superior to exact keyword-based matching. Our experiments show that the 2D-PIR similarity algorithm also perform better than Gudivada's similarity algorithm since it has better filtering capability.

In the following sections we will discuss the implementation of the 2D-PIR data

[1]The prototype is available in http://www.cse.unsw.edu.au/ nabil/thesis/program/twodpir/

model using a *bi-array*, the system prototype, experiments on the prototype and comparison with similar systems.

4.2 Representation of the 2D-PIR Data Model

Recall that a picture or an image is modelled as a 2D-PIR graph, which is a connected labelled digraph $G(V, R)$ where V is a finite set of objects in a picture and R is a set of edges labelled by 2D-PIR relationships.

Generally a graph is represented using an adjacency matrix or linked list. However, a 2D-PIR graph is a complete directed graph with the exception that if A and B are nodes then an arc from B to A can be omitted if there exists an arc from A to B since the 2D-PIR relationship between B and A can be inferred from the relationships between A and B (2D-PIR relationships are symmetric). This property suggests that a simpler and more efficient representation of a 2D-PIR graph which we call the *bi-array* representation. Details of this array will be discussed in subsection 4.4.2.

4.3 The System for Image Retrieval

The architecture of the system is shown in Figure 4.1.

The system is composed of five major components: user interface, symbolic image transformer, insertion engine, retrieval engine and image database.

4.3.1 User interface

Unlike traditional databases, queries in an image or multimedia database are most likely posed in non-textual forms, such as sketches or existing image (iconic or "thumbnail" versions). Therefore a Graphical User Interface (GUI) must be supported so that non-textual queries can be posed easily. Our definition of symbolic picture models using 2D-PIR graphs directly supports graphical querying. Using

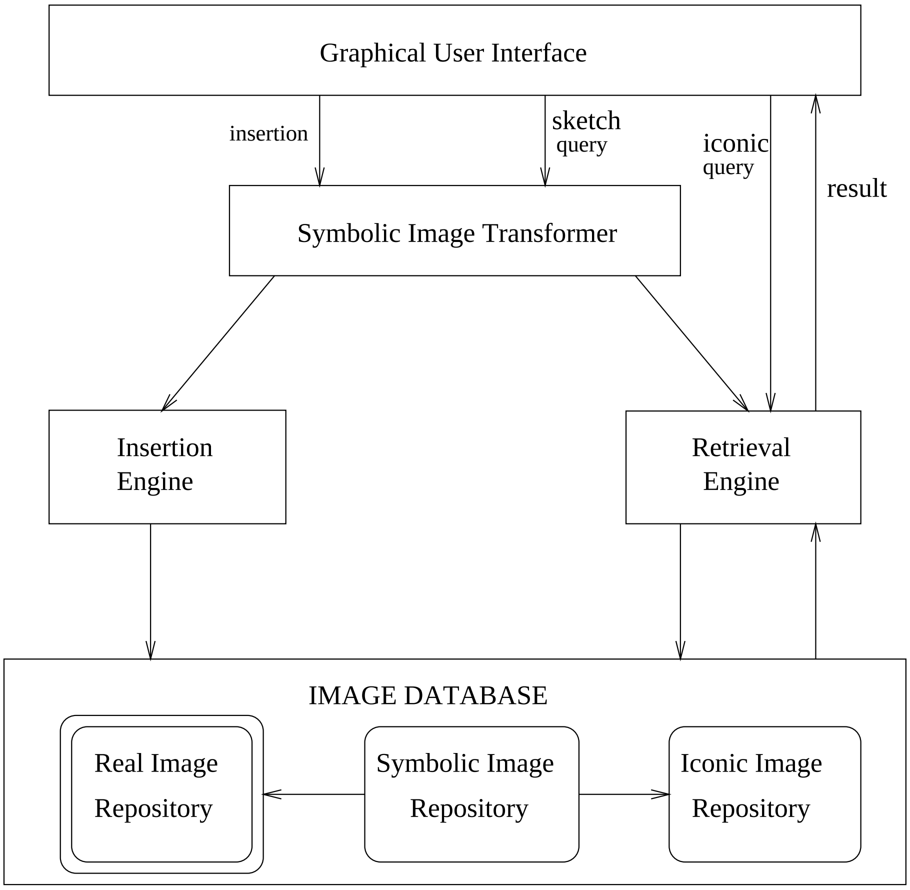


Figure 4.1: Architecture of a Prototype Image Retrieval System

this scheme, a sketch query is mapped to a symbolic picture which is ready to be compared with symbolic pictures stored in a database.

In the current prototype, all interaction with the system is through a GUI. This interface is designed for both image insertion and retrieval so that a user can use the system easily with very little training.

4.3.2 Symbolic image transformer

The symbolic image transformer converts an image to the corresponding 2D-PIR representation. An image is considered as a set of polygons representing the objects in the image along with symbols to identify each object. The symbolic image transformer accepts the set of polygons and their corresponding symbols and then computes 2D-PIR relationships among objects. The symbolic image transformer also accepts a user query sketch as a set of polygons and the corresponding symbols.

The output of this transformer is a *bi-array* that represents an image as a 2D-PIR graph.

4.3.3 Insertion engine

Once the user chooses an image to be inserted into the database, the system will display the image in the main window. The user is required to identify each object in the image by manually drawing approximate boundaries and assigning a text-based identifier for the object. If all of the objects have been identified, the user instructs the system to store the image. The system (through the symbolic image transformer) computes the 2D-PIR relationships among the objects and generates a 2D-PIR graph for the image and stores this representation along with pointers to the original image and its iconic version. Note that the user-supplied identifier is stored as symbol in a 2D-PIR graph and becomes the object-identifier for image retrieval.

4.3.4 Retrieval engine

Image retrieval is either *sketch-based* or *image-based*. Sketch-based queries begin by the user drawing a sketch containing rough object-outlines in some spatial arrangement and identifying each object. Image-based queries are commenced by clicking on an iconic image, either from a previous query or from a set of images chosen randomly from the entire image database. In both cases, the query is transformed into a 2D-PIR graph which is then used as the basis of the matching process.

In our prototype, matching is *approximate*. Approximate (similarity) matching uses the set of objects that are common to both the query and the image and uses the previously defined similarity measure on the spatial relationships between the common objects. In the case of similarity retrieval, the user may adjust the retrieval accuracy, which effectively sets a threshold value on the 2D-PIR graph similarity measure. An accuracy threshold of 0 means that any images having at least some of the objects in the query will be retrieved, regardless of the spatial arrangement of the objects. An accuracy of 100 means that only images for which the set of common objects has exactly the same spatial arrangement in both the image and the query will be retrieved.

4.3.5 Image database

Conceptually, the database is partitioned into three parts: a real image repository (where the original full-sized images are stored), an iconic image repository (where an iconic version ("thumbnail") of each image is kept) and the symbolic image repository (which holds the 2D-PIR graphs). Physically, we store only a pointer (URL) to the original image, so, in fact, the system can form the basis of a network-wide image database. Iconic images are stored locally since they are presented to the user in the first stage of the retrieval process.

The symbolic images are also stored locally, and are central to the image retrieval process. Each symbolic image contains:

- pointers to the original and iconic images
- a set of symbols representing objects in the image, using the object-identifiers supplied by the user
- a set of relationships among objects in the image, corresponding to the 2D-PIR graph

4.4 Design and Implementation Techniques

We adopt an object-oriented approach in building the prototype. We will not discuss the detailed design and implementation of the classes that make up the prototype, rather we will discuss the underlying concepts that we used to construct classes and objects in the prototype.

There are two important structures used in the prototype, *pointset* and *bi-array*. The *bi-array* structure is built on top of the pointset structure to represent a network of objects and their spatial relationships.

4.4.1 Pointset

A picture or an image is normally composed of some objects. Each object in a picture is considered as a set of points or a pointset. Each point is of the form $(x_1, x_2, \ldots x_n)$ where $\{x_i \mid i \in (1 \cdots n)\}$ are spatial coordinates. For a two-dimensional spatial object, its spatial points are normally represented as (x, y) points. A spatial object in a picture is considered as a *closed* set containing an *interior* and a *boundary*.

Representing a spatial object using both its *interior* and *boundary* is expensive since an interior normally contains very many points. Fortunately, we can represent a spatial object using its boundary. The boundary of a spatial object in an image can be captured automatically using available image processing algorithms or captured manually by dragging the pointer around the object. We use manual method to capture boundary of an object since in general it is not possible to do so automatically.

The boundary of an object is represented as a pointset. An pointset here generally describes a polygon. However, if the pointset is a singleton set then the object is a point and if the pointset containing two points then the object is a line. In Java, the pointsets can be represented as vectors of points [2].

Once every object in an image has been captured and identified, the image is represented by a set of pointset objects. *A pointset object* consists of the boundary pointset along with other attributes that describe the object. At present the only attribute used is the object's name; however visual attributes such as colour and shape could also be used.

The prime *method* for pointset objects is the computation of 2D-PIR relationships. This method computes the 2D-PIR relationships between pointsets.

Recall that a 2D-PIR between two spatial objects (pointsets) A and B is a triple (δ, χ, ψ) where δ represents topological relationships, χ and ψ represents relationships between object projected along the x-axis, and ψ respectively.

[2] a vector object in Java is a dynamic collection of objects of any type

There are two steps in computing the 2D-PIR relationship between two pointsets. The first step is to compute χ and ψ and the second step is to compute δ. It is important to compute χ and ψ first since in some cases we can avoid intensive computation of δ. For example, if χ or $\psi \in \{<, >\}$ then immediately we can decide that $\delta = dt$ (disjoint).

Let P_1 and P_2 be two pointsets. The following algorithm computes components of the 2D-PIR. To construct the 2D-PIR relationships between pointsets, we need only integrate the results of the first step and the second step.

ALGORITHM: Compute 2D-PIR relationships between two pointsets
INPUT: Two pointsets (P_1 and P_2)
OUTPUT: 2D-PIR relationship between P_1 and P_2
/*The first step: Compute interval relationship (IR) */.
begin /* first step */
    Let s_1, f_1, s_2, f_2 be the minimum and maximum values
    of intervals of P_1 and P_2. (Note that interval can be interval
    projection of a pointsets on x-axis or y-axis)
    if ($f_1 < s_2$) then IR = *before*
    elseif ($s_1 > f_2$) then IR = *after*
    elseif ($f_1 == s_2$) then IR = *meets*
    elseif ($s_1 == f_2$) then IR = *meetsInverse*
    elseif ($s_1 < s_2$)
        if ($f_1 < f_2$) then IR = *overlaps*
        elseif ($f_1 == f_2$) then IR = *finishesInverse*
        else then IR = *duringInverse*
        endif
    elseif ($s_1 == s_2$)
        if ($f_1 < f_2$) then IR = *starts*
        elseif ($f_1 == f_2$) then IR = *equals*
        else then IR = *startsInverse*
        endif

elseif ($s_1 > s_2$)

if ($f_1 < f_2$) then IR = *during*

else if ($f_1 == f_2$) then IR = *finishes*

else then IR = *overlapsInverse*

endif

endif

return then IR

end /* first step */

/* The second step: Compute topological relationship (TR) of P_1 and P_2.*/

begin /*second step*/

compute χ and ψ using the first step algorithm

if $\chi \in \{<, >\}$ or $\psi \in \{<, >\}$ then TR = *disjoint*

else

compute pointset intersection between P_1 and P_2

if the result is disjoint then TR = *disjoint*

elseif the result is overlap then TR = *overlaps*

elseif result is share boundaries but their interiors are disjoint then

TR = *touches*

else (${P_1}^0 \subset {P_2}^0$ or ${P_2}^0 \subset {P_1}^0$)

if their boundaries is not the same then TR = *inclusion*

else TR = *equals*

endif

endif

endif

return TR

end /*second step*/

The worst case complexity of the first step (compute interval relationships) is $O(n)$ where n is the cardinality of a pointset if we compute minimum and maximum value of intervals. However, if the minimum and the maximum value of intervals is stored during image insertion process (computation is done during insertion process), the complexity of the first step is $O(1)$.

The worst case complexity of the second step (compute basic topological relationships) is $O(mn)$ where m and n are the cardinality of pointsets P_1 and P_2 respectively.

4.4.2 Bi-array

As mentioned previously, we used a data structure called a *bi-array* to represent the 2D-PIR graph of a symbolic image. A *bi-array* consists of two dynamic arrays, called the *symbol array* and *relationship array* respectively. The *symbol array* contains names or concepts that identify each object in an image and the *relationship array* contains 2D-PIR relationships among objects in an image.

Consider Figure 4.2, which is a picture of a sedan hit by a truck in Rainbow St. beside a tree.

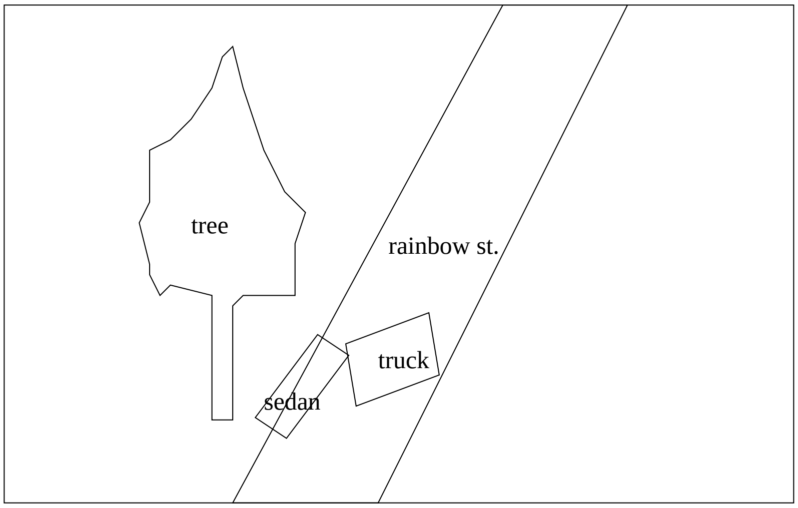


Figure 4.2: A car was hit by a truck in Rainbow St.

Figure 4.3 illustrates the bi-array representation of this images. Note that the image contains four objects of interest: tree, rainbow st, sedan and truck.

The purpose of the *bi-array* representation is to save storage space required for a 2D-PIR. It achieves this by storing object identifiers only once, and not using them

symbol array:	tree	rainbow st	sedan	truck		
relationship array:	(dt,o,d)	(dt,o,oi)	(dt,b,di)	(ov,di,di)	(ic,di,di)	(to,m,o)

Figure 4.3: Bi-array representation of figure 4.2

repeatedly in the *relationship array.* To achieve this, we need a mapping that tell us which pair of objects is involved in each relationship.

Given two objects represented by their names stored in *symbol array*, a mapping function is needed to find the corresponding 2D-PIR relationship between them, which is stored in *relationship array.* For example, the 2D-PIR relationship between "sedan" and "truck" in the Figure 4.3 is (to, m, o).

Given the location of symbols (names) in the *symbol array*, find the corresponding location of 2D-PIR relationship in *relationship array* and return the 2D-PIR relationship.

Let i and j be the locations of first and second symbols and n be the size of the *symbol array.* Here is the formula to find a location of 2D-PIR.

$$Loc = \frac{(2n - (i+1))i}{2} + j - i - 1$$

4.5 Experiments

4.5.1 Testbed

The database was populated with around 400 images of different types [3]. The images were grouped around a number of themes (e.g. Opera House, sunset and various animals). There are eight different collection of images used as testbed shown in Table 4.1.

[3] Most images are public domain images taken from Corel Corporation, Ottawa, Canada: http://leviathan.tamu.edu:70/1s/slides/Corel, excepts for a set of Sydney scenery images which were scanned from commercial postcards

Type of collection	Number of image	Description
Sydney water front city	52	Various images of Sydney such as Skylines and Harbours.
Antelopes	36	Various images of antelopes at different locations and actions.
Falcons	17	Various images of falcon.
Hawks	17	Various images of hawk.
Owls	22	Various images of owl.
Monkeys	15	Various image of apes, monkeys and mandrill.
Sunset	28	Various images of sunset scenery.
Mountains	14	Various images of mountains and hills
Others	199	Various uncategorised images

Table 4.1: Type of images used as a testbed

4.5.2 Performance measured

In this experimental work, we aimed to test the effectiveness of similarity retrieval on varieties of image collections, using the the concept *recall* and *precision* measures used in information retrieval.

Five queries were chosen for each type of collection. We asked 10 postgraduate students (users) to determine the similarity between the query image and the collection and assign five different values: *very similar*, *similar*, *bit similar*, *quite different* and *totally different*. If the majority of user decide that any image from the collection is toward similar (*bit similar*, *similar*, or *very similar*) with the query image then the image is considered as a relevant image.

The following standard definition is used for measuring recall and precision.

Definition 9 *Let p be number of all images that are relevant to the query image, q be the number of relevant images retrieved, and r be the total number of images retrieved. Then*

$$Recall = R = \frac{q}{p} \times 100 \qquad Precision = P = \frac{q}{r} \times 100$$

4.5.3 Results

We conducted five experiments on each collection by posing five query images taken from each collection (note that the same queries are also asked to the 10 users (postgraduate students)).

Each query was repeated for a number of different accuracy levels (from 0 to 100). We then calculated the (geometric)mean of recall and precision of retrieval for the five queries in each experiment for each accuracy level.

Table 4.2 gives the results from our experiments. Detail results are given in the Appendix A.

Collection	Accuracy/threshold value											
	0		20		40		50		60		80	
	R	P	R	P	R	P	R	P	R	P	R	P
Sydney	100	29	100	38	94	78	66	89	28	98	8	100
Antelopes	87	32	83	41	67	74	60	82	35	100	21	100
Falcons	90	52	90	72	79	78	79	78	69	82	50	100
Sunsets	89	25	77	31	53	74	48	77	29	94	20	100
Monkeys	100	20	100	29	50	100	50	100	25	100	25	100
Hawks	95	50	95	53	79	91	69	96	43	100	24	100
Owls	96	33	84	45	81	90	74	94	62	100	24	100
Mountains	100	14	100	26	74	56	51	76	30	100	23	100
Mean	95	32	91	41	72	80	62	87	40	97	24	100

Table 4.2: Recall/Precision Table (P: precision, R: Recall)

There are six different accuracies of 2D-PIR matching values or threshold values in Table 4.2: 0, 20, 40, 50, 60 and 80. Zero means matching is solely based on symbols (i.e. two symbols connected by conjunction); in other words, it is keyword-based matching. We use zero to compare similarity matching using 2D-PIR with keyword-based matching. The accuracy of 20, 40, 50, 60 and 80 thresholds are used to show the effect of 2D-PIR on recall and precision of similarity retrieval. Note that a user of the system may adjust the threshold value according to his/her preference.

From Table 4.2, it can be seen that on average the recall value is 95 percent for keyword matching. However, at the same time, the precision is low. Note that currently the prototype uses exact keywords (symbol) matching (two keywords). When the threshold value is increased, then the recall value is decreased, but the precision value is increased. For example, the precision value increases by a factor of 2.7 and 3.0 for threshold values of 50 and 60 percent respectively. If the threshold value is very high, for example 80 or above, then the system tends to do exact matching on 2D-PIR relationships.

4.5.4 Examples of retrieval

Figure 4.4 is the result of a query on a collection sunset pictures using similarity threshold 30. Note that the top left is the query. Note also they are ordered by decreasing similarity, the top left being 100% by definition. Figure 4.5 is the result of query on the same collection using similarity threshold 60. The result shows that the system retrieves similar images especially when the threshold value is quite high. Figure 4.6 and Figure 4.7 show a sketch query and the corresponding result.

Figure 4.4: Result of query using on a collection picture of sunset; threshold: 30

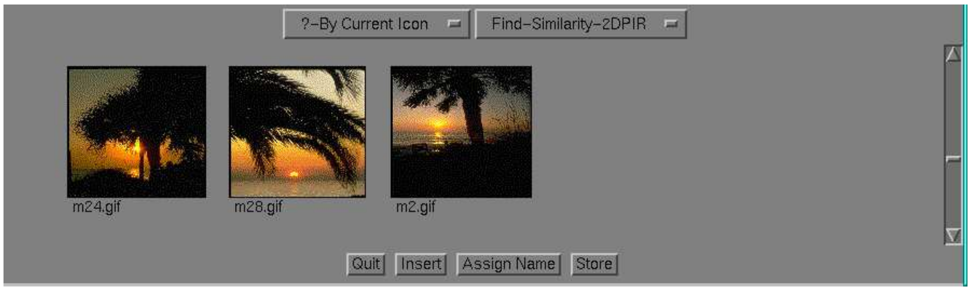


Figure 4.5: Result of query using the top left icon as query; threshold: 60

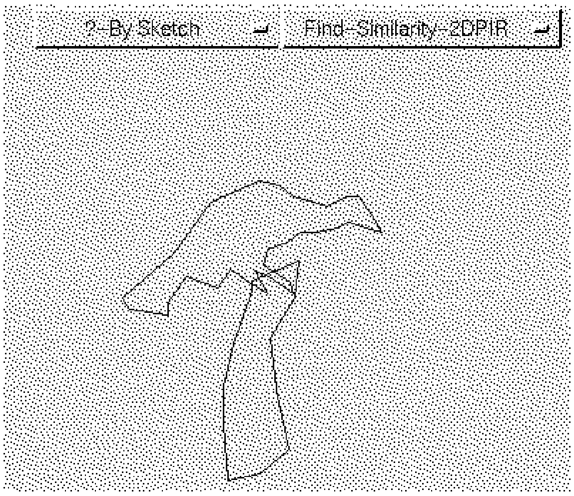


Figure 4.6: Sketch query: find a hawk perching on a stump

Figure 4.7: Result of the sketch query "find a hawk perching on a stump"

Figure 4.8 shows a result of a query (top left is the query) on a picture and its rotated versions (25, 90, 180 and 270) and its reflected versions (both on horizontal and vertical reflection).

Figure 4.8: Result of query on rotation and reflection collection

4.6 Comparison with Other Methods

Currently there is considerable amount of research on content-based (similarity) retrieval of images. However, most of these works uses visual features such as colour, texture and shapes. Some prototypes of these works are available in the Internet for demo including QBIC [49] and Virage [112]. Since these systems do not use spatial relationships in retrieving images, we will not compare our work with them.

There is already work that uses spatial relationships for image retrieval. Takahashi *at al.* [109] uses directional relationships among objects in an image for retrieval. Chang [13] uses directional relationship codes called 9-DLT to cluster similar images. Yu *et. al.* [117, 103, 104] use E-R diagrams to represent images and image matching is based on their E-R representation matching.

Gudivada [44, 45] also uses symbolic images. His similarity measure is based on angle differences among objects in the image. Gudivada's approach can handle rotation well. Lee [70] proposed similarity retrieval based on Chang's 2D-strings representation. Lee's approach cannot handle rotation invariance of images and is inferior to Gudivada's method.

Gudivada's method is quite close to ours in term of using graphs to represent spatial relationships. The main difference is that we use unified spatial relationships (directional and topological relationships) while he uses orientation angles, which can be viewed as low level directional relationships (one may argue that angle alone cannot be used to represent directional relationships, especially for image containing occluded objects).

Similar to Gudivada's method, our method can also handle rotational images as discussed in [86]. In fact we can also handle reflection [86], which is not discussed by Gudivada.

In this section we compare our work with Gudivada's work in details (we implemented Gudivada's algorithm and compare it with ours using the same testbed images). We also compare our work with Takahashi, Chang's 9-DLT, and Yu.

Before we compare the performance of retrieval between ours and Gudivada's, we would like to point out two errors in Gudivada's similarity function. In Gudivada's similarity algorithm each relationship (each edge in the graph) is a weighted value of $(1 + \cos(\theta))/2$. To get an idea what the θ is, consider Figure 4.9:

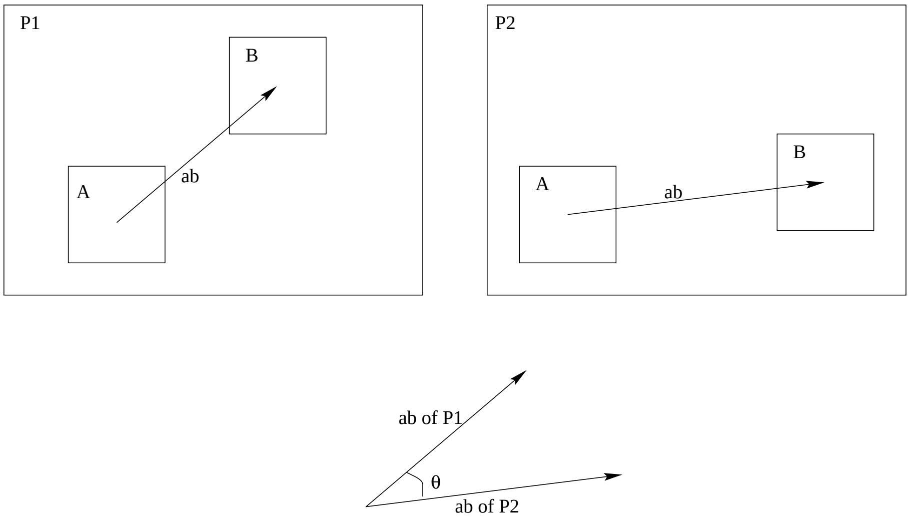


Figure 4.9: Orientation angle

The line segment *ab* of P1 is the line segment created between the central points (indicated by small letter) of objects A and B in P1. Similarly ab of P2 is the line segment created between central points of objects A and B in P2. The orientation angle θ is the angle between line segments ab in P1 and P2.

Now, consider Figure 4.10.

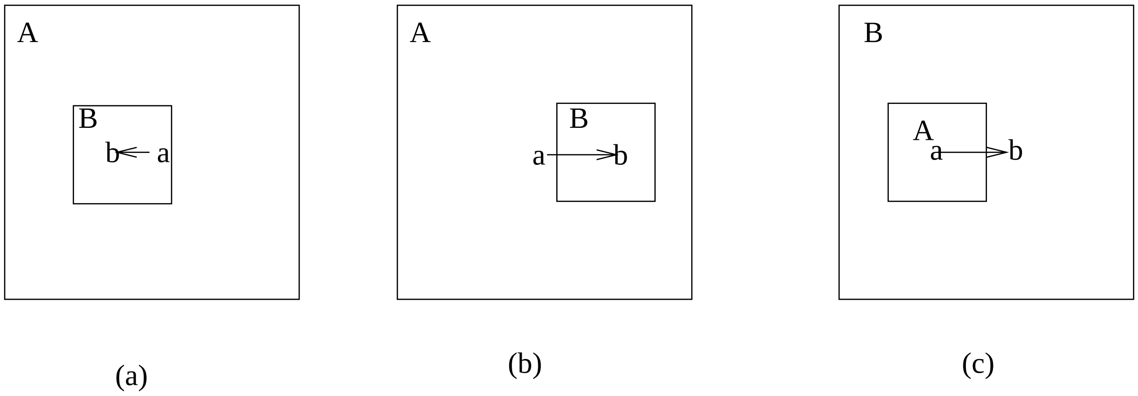


Figure 4.10: False reject and false gain of Gudivada's approach

The angle θ between A and B in Figure 4.10 (a) and (b) is 180^0, therefore the weight between edge AB in Figure 4.10 for a and b is 0. This means that in Figure 4.10, (a) and (b) are totally different, which is surely incorrect (*false reject*). On the other hand, consider Figure 4.10 (b) and (c). The angle θ between A and B in Figure 4.10 (b) and (c) is 0^0, therefore the weight between edge AB in Figure 4.10 for b and c is 1 this means Figure 4.10 (b) and (c) are the same image, which is incorrect (*false gain*). These problems can be removed if the reference point of an object is the left-bottom corner of it bounding rectangle (or other corners). However, the corner point of a bounding rectangle cannot remove the problems completely, it solves only on cases depicted in Figure 4.10. These problems appear since the topological relationships between objects are ignored.

In order to compare the performance between ours and Gudivada's similarity retrieval, we implemented Gudivada's similarity algorithm and tested it using the same image testbed that we use (note that our image testbed are real images, compare to Gudivada's testbed [45], which are artificial iconic images).

The following figures are recall-precision graphs that compare the retrieval performance of our similarity algorithm and Gudivada's similarity algorithm.

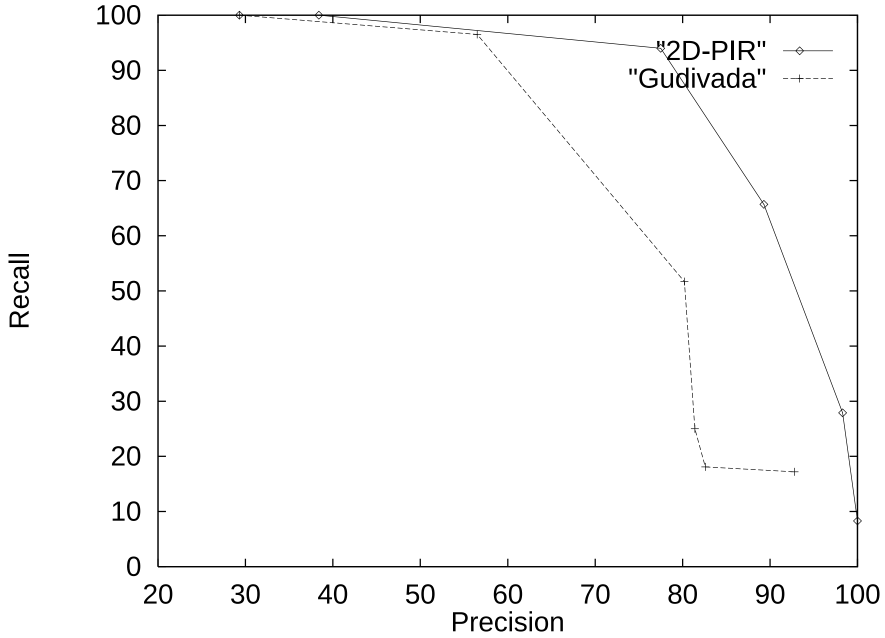


Figure 4.11: Recall-precision graph on images of Sydney

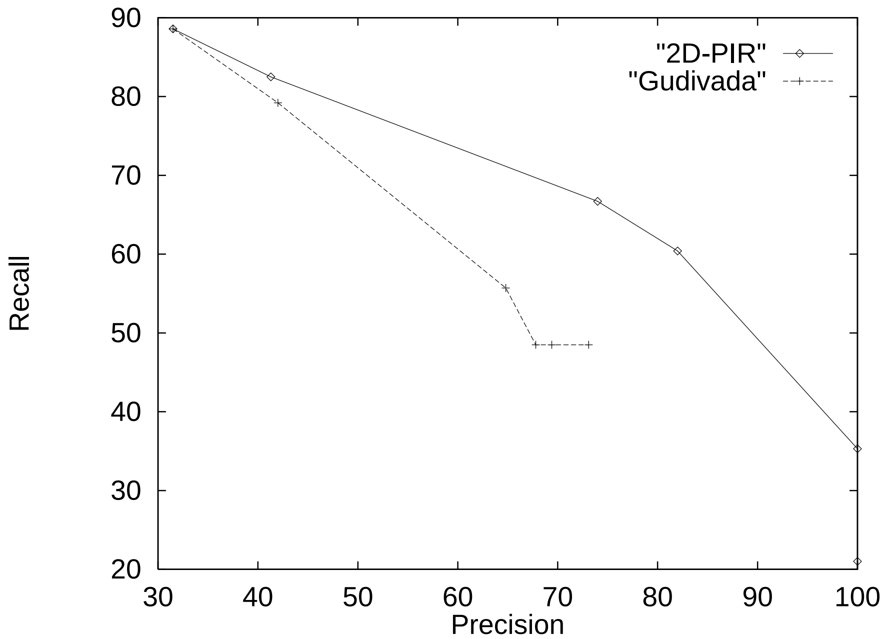


Figure 4.12: Recall-precision graph on images of antelopes

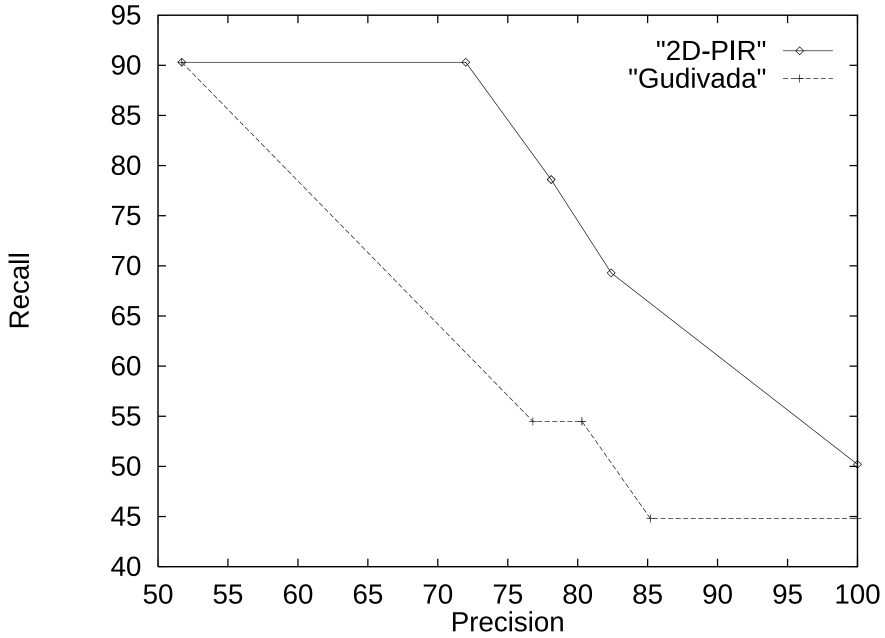


Figure 4.13: Recall-precision graph on images of falcons

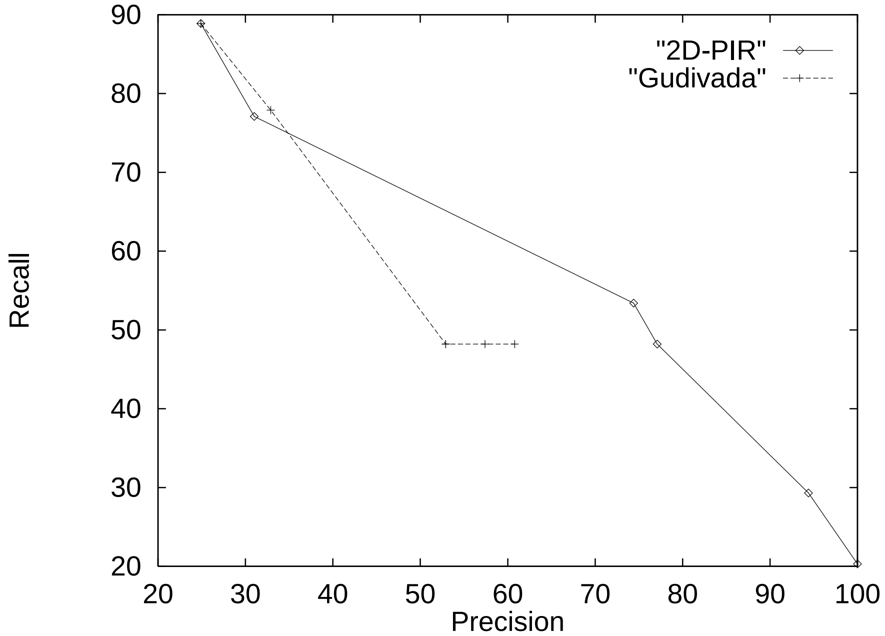


Figure 4.14: Recall-precision graph on images of sunsets

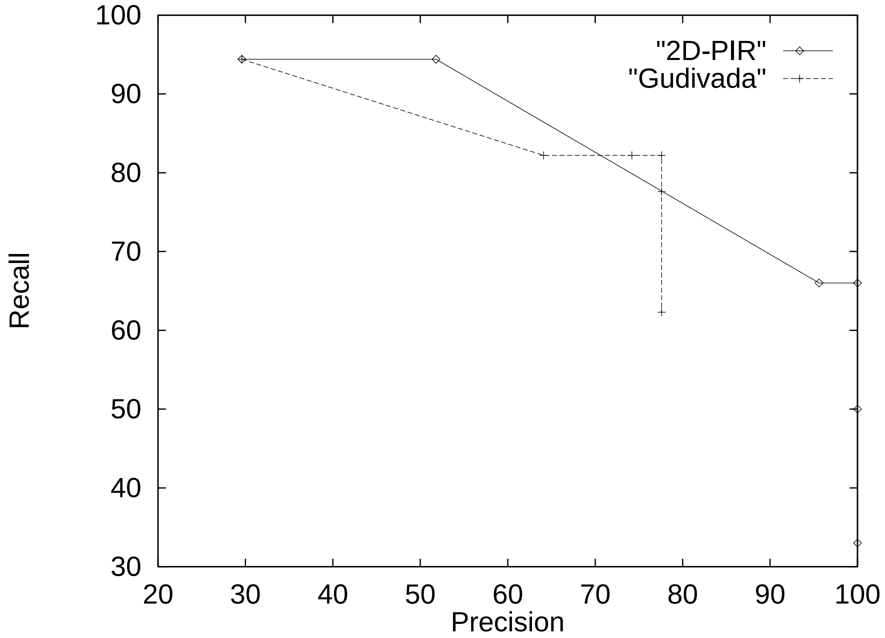


Figure 4.15: Recall-precision graph on images of monkeys

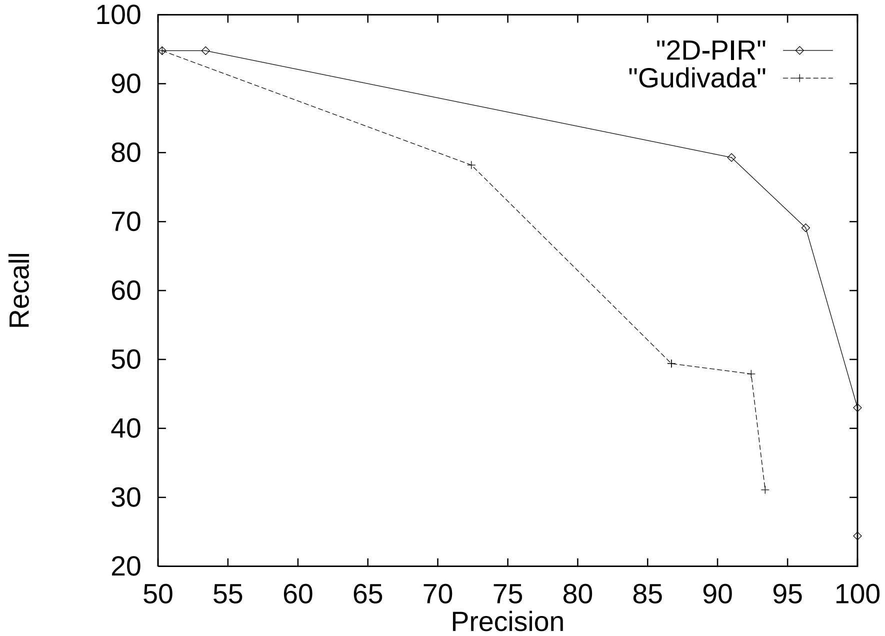


Figure 4.16: Recall-precision graph on images of hawks

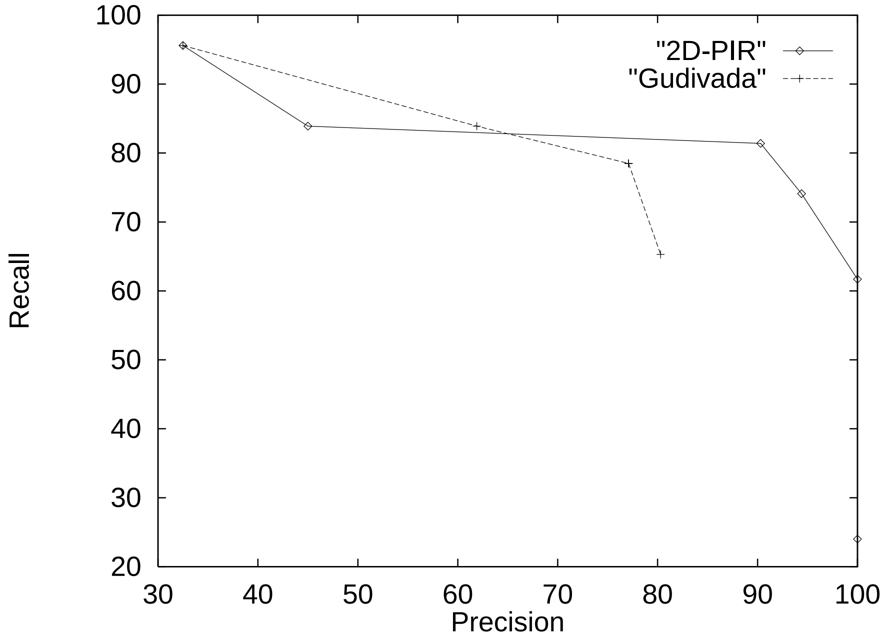


Figure 4.17: Recall-precision graph on images of owls

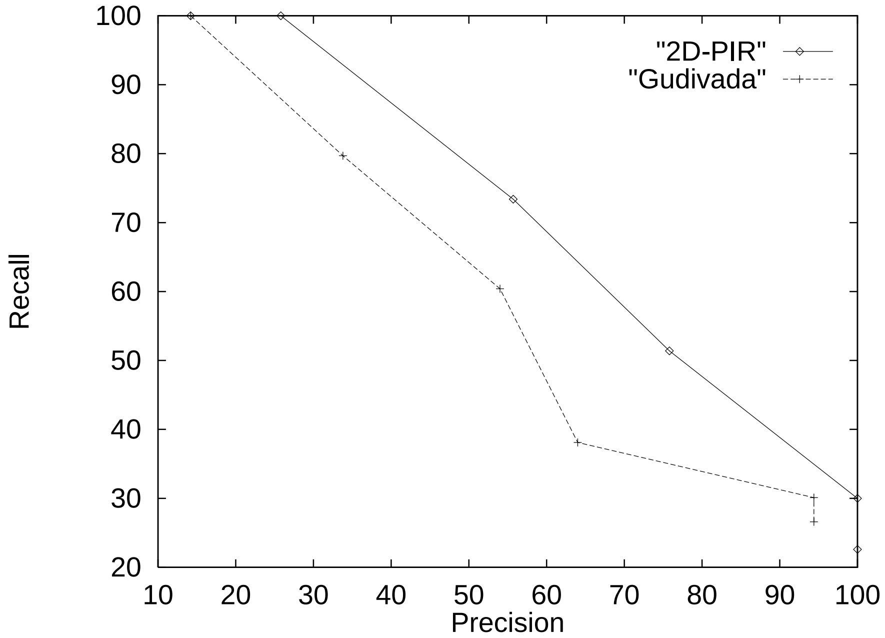


Figure 4.18: Recall-precision graph on images of mountains

Based on the recall-precision graphs on eight collections of real images above, our similarity algorithm, which is based on 2D-PIR, is superior compared to Gudivada's similarity algorithm. In general, our algorithm yields more precise results than Gudivada's at the same threshold value. At the same time our algorithm in most cases provides a larger recall value than Gudivada's at the same precision value. This means our algorithm has better filtering capability than Gudivada's.

So far we have discussed the application of 2D-PIR on similarity retrieval of images and compared it to Gudivada similarity retrieval method. 2D-PIR has wider applications than Gudivada's orientation angle.

For example, the orientation angle method cannot guarantee deriving correct directional relationships between objects in an image. Consider Figure 4.10 (a) and (b) again. Any directional relationship between A and B derived from orientation angle cannot be accepted since there is no such relationship. Hence, the answer of a query involving directional relationships cannot be guaranteed to be correct. This is unacceptable in many typical applications such as GIS. In contrast, 2D-PIR has many potential applications such as spatial databases (GIS, LIS). 2D-PIR relationships integrate directional and topological relationships among objects in an image, therefore any query involving topological and directional relationships can be answered. In fact, 2D-PIR provides the power to express partial directional relationships such as spatial relationship between Portugal and Spain in a map. Note that an extension of 2D-PIR can also handle moving objects, as will be discussed in the next chapter.

Now, we will compare briefly our work with Takahashi's fuzzy spatial relationships, Chang's 9-DLT and Yu's E-R picture. It is interesting that these works use a different approach in representing spatial relationships. In the following discussion we refer to Takahasi's work as *spade*, the prototype name that implements fuzzy spatial relationships for pictures retrieval, Yu's work as *E-R picture*, Chang's work as *9-DLT* and our work as 2D-PIR.

Spade represents an image using a table (a relation) and rules. The table stores attributes of images which are bibliographic indexes and rules represent spatial information including spatial relationships between objects in the image. *9-DLT* uses a set of 3-tuple (O_i, O_j, r_{ij}) to represent an image, where O_i and O_j are symbols representing objects in the image and r_{ij} is the spatial relationships between O_i and O_j. *E-R picture* represents an image using E-R diagram. Each object of interest in the image is an entity which can have attributes. The relationship between entities is either a spatial relationship or a non-spatial relationship (an action relationship). 2D-PIR uses a graph to represent an image where nodes are objects represented by their name and edges labelled by 2D-PIR relationships between objects.

2D-PIR and *E-R picture* representations are quite similar. Entities in an E-R diagram represent objects in an image which correspond to nodes in 2D-PIR. Each entity in the *E-R picture* can have several attributes. In an 2D-PIR graph each node currently has only one attribute, the name of an object. However 2D-PIR can be extended to include other attributes, especially attributes that can be extracted automatically such as shape, colour and texture. These attributes can be stored along with names by extending the *symbol array*.

In terms of spatial relationships, *spade* supports directional relationships. *Spade* uses the fuzzy concept to account for the fuzziness of directional relationships. To derive fuzzy directional relationships between objects in an image, a heuristic function of geometrical quantity is used. A set of if-then rules are also provided to combine spatial relationships. In addition, spade also stores positions of objects in an image and distance between objects. *9-DLT* supports eight directional relationships which are encoded as the integers 1..9. *E-R picture* supports directional and topological relationships, but these relationships are assigned by a user. In addition, a user can also assign an action relationship between entities in an image. 2D-PIR integrates topological and directional relationships and uses them in representing relationship between objects in an image.

Spade, 9-DLT and 2D-PIR derive spatial relationships automatically based on the positions of objects in an image. In *E-R picture* these relationships are assigned manually by a user. *E-R picture* provides a set of rules for deriving spatial relationships from the existing spatial relationships since the assigned spatial relationships may be incomplete. The incompleteness problem of spatial relationships is not an issue in 2D-PIR, since integrated directional and topological relationships are derived automatically and all 2D-PIR relationships among objects in an images are computed.

Systems which support only directional relationships such as *9-DLT* and *spade* can have problems similar to Gudivada's system. This is because not all spatial relationships are directional relationships; some are topological relationships. Compared to other systems, 2D-PIR provides the richest spatial relationships between objects. Hence it provides more information regarding how an object is spatially related with others. The *E-R picture* provides both directional and topological relationships, but these relationships are assigned by a user and are thus potentially incomplete, hence could reduce the effectiveness spatial relationships for similarity retrieval.

Only *E-R picture* and 2D-PIR use similarity measures for retrieval. *E-R picture* defines similarity measures both for spatial relationships and entities (objects with attributes) and 2D-PIR only defines similarity on spatial relationships. Both *9-DLT* and spade do not have similarity measures. *Spade* uses the fuzzy concept to retrieve "similar" images. *9-DLT* uses the concept of clustering (consecutive retrieval organisation where similar images are stored on contagious blocks) to find "similar" images. This approach is good for fast access, unfortunately, a cluster may contain different images since there is no well-defined method to measure image similarity.

A query in *spade* is specified using natural language (Japanese) which is processed two passes. In the first pass a query is transformed in SQL-like form and sent to a relational database to retrieve images (represented by keys) based on attributes. The set of images retrieved from the first pass are then processed in the second pass. In the second pass all images from the first pass are filtered using spatial relationships between attributes that represent objects in images. *9-DLT* has no query model. However, a sketch-based query can be developed. In *E-R picture* a query is posed in form of E-R diagram. A graphical user interface supports a user step-by-step to compose a query. This query is then transformed to E-R diagram. In 2D-PIR a query can be posed by a sketch and an icon. In the current prototype a user can draw a sketch in a window and then ask the system to find similar images. Results in the form of "thumb-nail" images will be displayed ordered based on similarity. A user can select an icon image to act as another query. 2D-PIR provides query based on random images. If a user chooses this option, a number of random images (10) will be displayed in the window. A user may start retrieval by choosing one of the images ("thumb-nail" images) to start more specific retrieval. This process can be repeated several times until the user finds the intended image(s). In 2D-PIR, a similarity threshold can be adjusted by a user. This adjustment is not provided in *E-R picture*.

One of the features of 2D-PIR which is not supported by other systems is that the model is extensible to moving objects. By extending 2D-PIR to temporal sequence of 2D-PIR (TS-2D-PIR) we can model moving objects. This extension will be discussed in the next chapter.

Table 4.3 summarises the comparison.

Features	Spade	9-DLT	Gudivada's	E-R picture	2D-PIR
Representation	rule-based	string	graph-based	E-R diagram	graph-based
How the representation is created	semi-automatic	n/a	semi-automatic	manual	semi-automatic
Type of relationships supported	directional	directional	Directional	non spatial and spatial (directional or topological)	directional and topological
How relationships captured	derive directional relationship from the position objects (automatic)	derive directional relationship from the position objects (automatic)	derive directional relationship from the position objects (automatic)	relationships are manually assigned by a user	derive topological and directional relationships based on objects boundaries, interior and their projections
Query model	natural language	no query model proposed	graphical query language (icon-based)	graphical (drawing e-r diagram) and keyword-based	sketch-based, icon-based and keyword-based
Support similarity matching	yes, to some extends using fuzzy directional relationships	yes	yes	yes	yes
Extensible	no	no	no	no	yes

Table 4.3: Comparison of 2D-PIR with other models

4.7 Summary

We have designed and implemented a prototype of image similarity retrieval based on the 2D-PIR method using the java language. we conducted several image similarity retrieval experiments with the prototype on several image collections.

If we compare keyword-based retrieval (0 threshold; two keywords connected with a conjunction) with 2D-PIR similarity retrieval, then the precision value in 2D-PIR is much higher. For example, the precision value increases by a factor of 2.7 and 3.0 for threshold values of 50 and 60 respectively. This is because 2D-PIR provides more information, such as directional relationships and topological relationships among objects (represented by keywords).

When we compare 2D-PIR similarity retrieval with Gudivada's similarity retrieval, 2D-PIR similarity retrieval performs better than Gudivada's on most collections that we used. Again, the 2D-PIR representation provides more information than Gudivada's. We briefly compare 2D-PIR with other systems that use spatial relationships. It seems that only E-R picture closes to 2D-PIR. However image insertion in E-R picture is manual while in 2D-PIR image insertion is semi-automatic.

Chapter 5

Modelling Moving Objects Using Spatiotemporal Information

5.1 Introduction

One of the most challenging tasks in multimedia data modelling is how to model a moving object or a set of moving objects in such a way that a query on moving objects is feasible. If such a model can be built, then content-based retrieval on motion pictures will benefit greatly from it.

The representation of moving objects requires both spatial and temporal data. There is already a substantial body of research in spatial databases [46, 97] and temporal databases [61]; however research on the integration of space and time in the database area is still in its infancy [2]. A current released survey on spatiotemporal databases [1] does not show a significant development in this area. As a consequence, research on the handling of large volumes of information about moving objects (e.g. for queries on video sequences) is also very limited [71].

Video retrieval systems are a very important application. Currently, most of video retrieval systems use textual description to retrieve video clips from a database [48, 91]. Video retrieval based on visual features has also been developed [118]. However, there has so far been no substantial work on content-based video retrieval using spatiotemporal features (in particular, spatiotemporal relationships). Future multimedia database systems will be required to answer queries such as: "find an object whose trajectory is similar to the trajectory drawn in a window" or "find a scene when object A enters another object B from the left". A combination query such as "find a scene when object A enters another object B from the left where the trajectory of A is similar to the trajectory drawn in a window" might also be posed. In order to develop such capabilities, we first require a model for moving objects. This chapter aims to take the first step towards the development of a data model for moving objects stored in image sequence databases such as video-clip databases (video server). Note that part of this chapter is taken from [83].

In subsequent sections we will describe our model of moving object(s) together with available operations. The next chapter will discuss a prototype system that implements this model.

5.2 A Moving Object Data Model

As we mentioned in Chapter 3, we model a spatial object as a pointset. A moving object is thus a pointset that continuously changes over time (i.e. a temporal sequence of pointsets). If we use a single point (e.g. centroid) to define the position of the object, this can be represented as a point in an n-dimensional space (we will consider primarily $n = 2$, but our model extends in a straightforward way to $n > 2$). For moving object this position changes over time. Note that we assume that the object is a rigid object and the motion is a translation.

The motion of such a moving object can be described by a sequence of spatiotemporal points, which we call a *track*. Each spatiotemporal point is of the form $(x_1, x_2, \ldots x_n, t)$, where the x_i's are spatial coordinates, and t is a temporal coordinate. The sequence $[(x_0, y_0, t_0), (x_1, y_1, t_1), \ldots, (x_n, y_n, t_n)]$, which gives the positions of a particular point in the object at times $t_0, t_1, t_2, \ldots, t_n$, describes the *track* of the object.

For retrieval of a moving object based on its *track* or trajectory, the above representation, which we call *point representation* is inefficient since we have to consider every point on the track. To overcome this problem we develop a higher-level representation of *track* based on *symbolic direction* and *distance*. This representation is much more efficient than the *point representation* both in terms of space required to store a *track* and time required for retrieving a *track*.

If we consider the position of a moving point at t_0 and then the position of the same point at some later time t_i, we can identify *distance* and *directional* relationships between the two positions. Let I denote the *interval* of time between t_0 and t_i (intervals are represented by their start and finish times (i.e. $I = (t_0, t_i)$). Let (x_0, y_0, t_0) and (x_i, y_i, t_i) be the positions of a moving point at t_0 and t_i respectively. The *track* of the point over the interval I can be represented as a triple (d, r, I) where d is the (Euclidean) *distance* travelled by the point during I and r is the fixed *direction* of the point's movement during I. The domain of d is the real numbers, while the domain of r is the set of possible directions depicted in Figure 5.1 (note that the domain of r can be expanded if a more accurate representation of direction is required). Under this scheme, the *track* of a moving object from time t_0 to t_n is represented as a sequence $[(d_1, r_1, I_1), (d_2, r_2, I_2), \ldots, (d_n, r_n, I_n)]$, where $I_1 = (t_0, t_1)$ and $I_n = (t_{n-1}, t_n)$. Note that since the direction of movement remains constant within each interval, an object moving in a complex manner will have a longer *track* representation than the one moving in a simple manner over a fixed period of time.

Given the above, a moving object is modelled as $(M, [(d, r, I)])$ where M is the name of the object and $[(d, r, I)]$ is its *track*.

Being able to model a single moving object is quite useful from a querying perspective (for example, we can ask queries about certain kinds of object trajectories) However, this model does not consider the relationships of a moving object with other objects (whether it moves or not) in a scene. A model that can describe how an object moves (its track) and how an object relates to other objects is clearly more expressive than the above model. Our objective is to create the latter model.

For this purpose, we extend our model of pictures (2D-PIR graphs) discussed in Chapter 3. There are two main reasons why we extend this picture model. First, we would like to build a unified model for static and dynamic multimedia data.

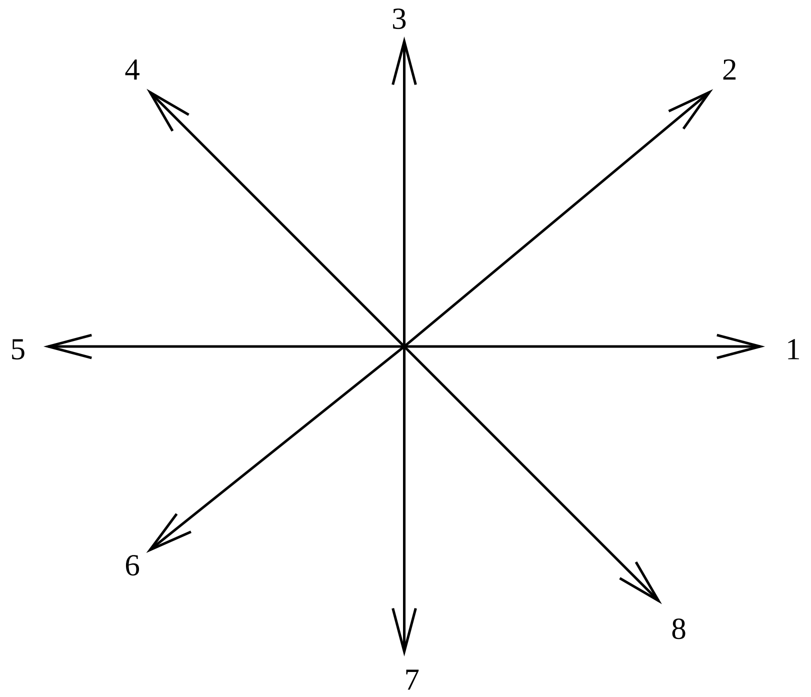


Figure 5.1: Directions

Second, a graph-based model (2D-PIR) allows us to integrate the idea of the *track* of a moving object with relationships between objects.

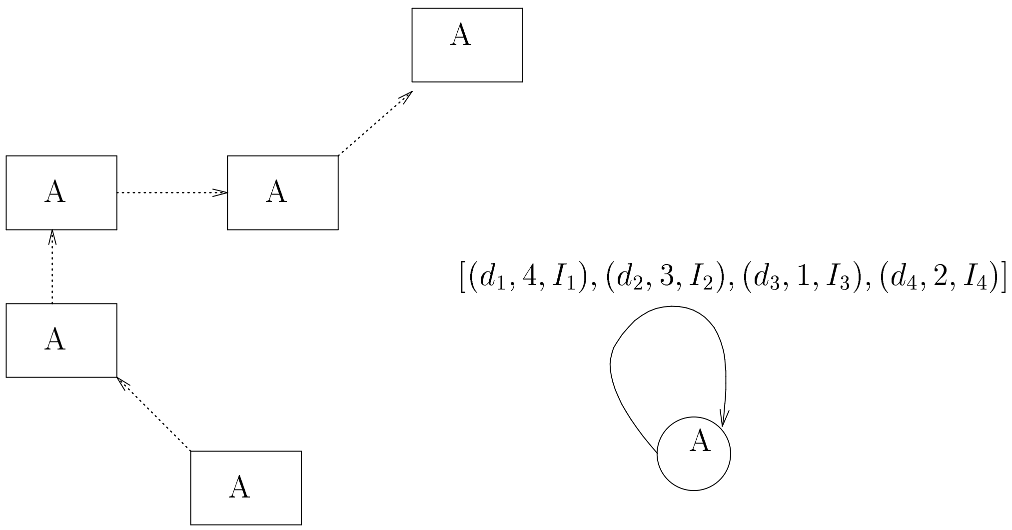
We begin with the concept of a *scene*: an abstraction that consists of a set of objects, where some of these objects may be moving. More formally:

Definition 10 *A* scene *is a connected labelled digraph* $S(M, R)$ *where* M *is finite non-empty set of symbols representing objects in the scene and* R *is a set of edges labelled by a sequence of spatiotemporal relationships (st-relationships) between objects.*

An *st*-relationship is a spatial relationship that holds between moving objects over a specified interval. Let r_{st} be a *st*-relationship, r_s be a spatial relationship and I be a time interval represented by (t_s, t_f) where t_s is the start time and t_f is the finish time of the interval, $r_{st} = (r_s, I)$. A *track* $[(d, r, I)]$ is a sequence of spatiotemporal relationships between a moving object with itself (self spatiotemporal relationship).

In this model, we extend the two-dimensional projection interval (2D-PIR) representation developed in Chapter 3 to include time intervals. This extension provides the capability to model spatiotemporal relationships between objects: that is, a sequence of $[(\delta_i, \chi_i, \psi_i, I_i)]$ for $i = 1..n$ where the δ are topological relationships, χ and ψ are interval (projection) relationships on the x-axis and y-axis respectively and I is a time interval. We call such relationships TS-2D-PIR, which stands for temporal sequence of 2D-PIR.

Figure 5.2(a) shows a scene of a moving object (A). The model for A's movement is: $(A, [(d_1, 4, I_1), (d_2, 3, I_2), (d_3, 1, I_3), (d_4, 2, I_4)])$, where d_i is the distance travelled in the corresponding direction over interval I_i (for $i \in \{1..4\}$). For example d_1 is the distance travelled in direction 4 over interval I_1. Figure 5.2 (b) is the graph corresponding to the scene of Figure 5.2 (a).



(a) (b)

Figure 5.2: A moving object (a) and the corresponding scene (b)

Figure 5.3 (a) shows a scene where objects A and B move towards each other until finally object A is inside object B. Figure 5.3 (b) is the corresponding scene (graph) of Figure 5.3 (a). The edge between A and B describes how objects A and B are related over time. The loop at A or B describes how each object moves.

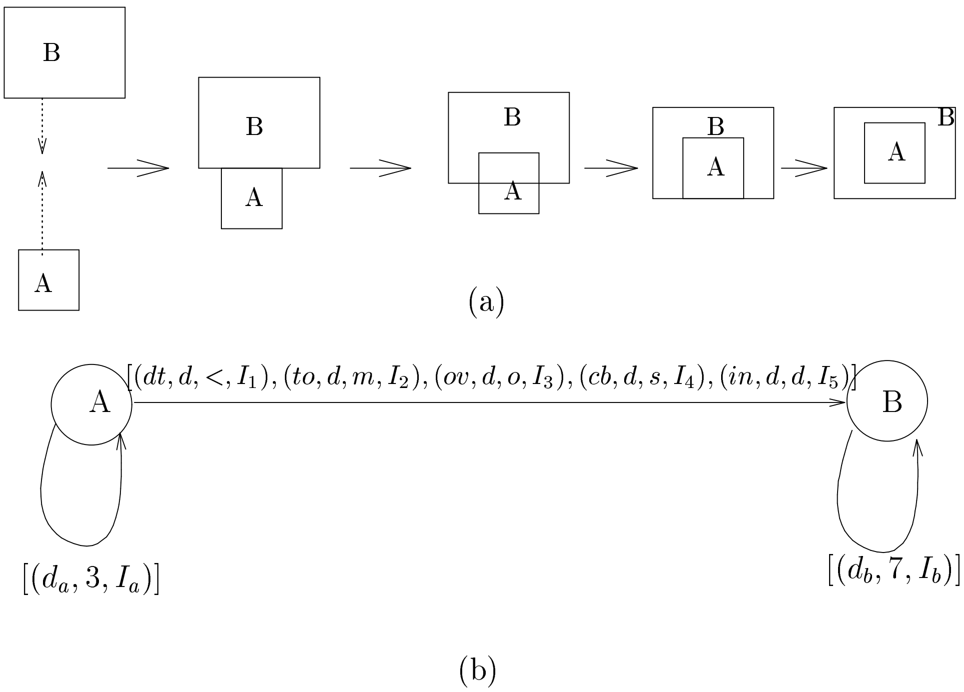


Figure 5.3: Two moving objects (a) and the corresponding scene (b)

5.3 An Algebra for Moving Objects

The following semantics of operations are defined for the model. We use the object-oriented *message passing* notation, *Obj op*(*args*), where *Obj* is an object, *op* and *args* is an operation's name (message) and the corresponding argument(s) of *op* respectively. The next section provides the syntax for a query language base on these operations.

5.3.1 Projection operation

The projection operation on a scene is an operation that creates a new scene based on an existing scene. The new scene is a subset of the original scene.

Let S be a scene; $m_1, m_2, \ldots, m_m$ be elements of M, a set of symbols representing objects in a scene; R be a set of spatiotemporal relationships among objects in a scene. Note that the elements of R are the sequences $[(d,r,I)]$ and $[(\delta,\chi,\psi,I)]$.

Definition 11 *The object projection of objects $m_1, m_2, \ldots, m_n$ of scene $S(M,R)$*

(i.e. $S\ \Pi(m_1, m_2, \ldots, m_n)$) is another scene $T(U, V)$ such that T is a graph of S containing objects $m_1, m_2, \ldots, m_n$ and V is a set of edges constructed according to the order of m_i $(i = 1 \ldots n)$.

Figure 5.4 illustrates this operation. Projection of Scene S on objects b and C creates the scene T containing objects objects b and C.

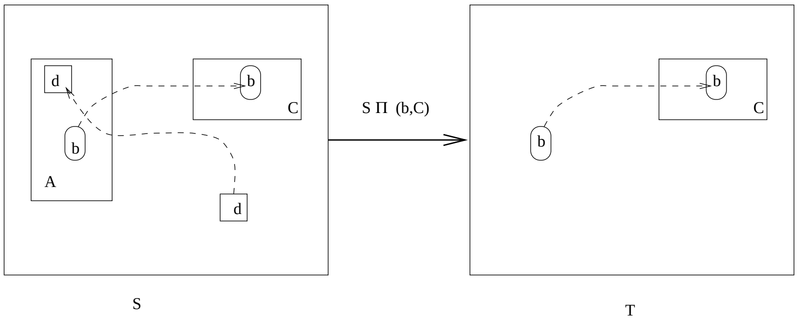


Figure 5.4: Projection of Scene S on objects b and C

This operation is needed if a user wants to isolate an object or some objects for a specific purpose, such as to observe the trajectory of an object or observe how objects are related in a scene. Sometimes a user needs to isolate or observe objects at a certain time interval. To do this, the following projection definition, called *spatiotemporal projection*, is provided.

Definition 12 *Spatiotemporal projection of a scene $S(M, R)$ is an object projection of a scene S over objects $m_1, m_2, ...m_n$ at interval I. The result of this projection is a new scene containing objects $m_1, m_2, ...m_n$ and their relationships at interval I. Formally, ($S\ \Pi(m_1, m_2, \ldots, m_n, I)$) is is a connected labelled digraph $T(U, V)$ where $U = \{m_1, m_2, m_3..., m_n\}$ and V is set of spatiotemporal relationships among object in U over interval I.*

Figure 5.5 illustrates spatiotamporal projection on scene S.

5.3.2 Selection operation

The *selection* operation (denoted by σ) finds an appropriate scene(s) in a set of scenes in order to answer a user query. If S_s and T_s are sets of scenes, *selection* is

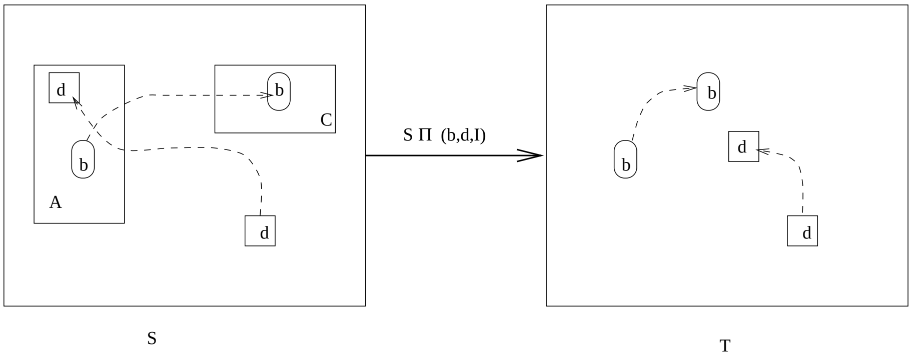


Figure 5.5: Spatiotemporal projection of Scene S on objects b and d at an interval I

defined as:

Definition 13 *$S_s : S\ \sigma_P$ is T_s such that $T_s \subset S_s$, P is a predicate on S_s and S is an iteration variable of S_s.*

The predicate has the form of containment of an object(s) in a scene, comparison of scenes, spatiotemporal relationships (both the spatial part and temporal part) and trajectories.

The containment predicate is the simplest one. Let $O_1, O_2, ...O_n$ be object's name and S be a scene. Then *S contains O* is true if S contains O, S contains $\{O_1, O_2, ...O_n\} \subset S$ is true if the set $\{O_1, O_2, ...O_n\}$ is contained in S.

The most interesting predicate for video data is most likely a similarity comparison. For example, the query: "find a scene(s) in a scene database that contains an object whose trajectory is similar to the trajectory drawn in the window".

Similarity comparison can be illustrated as follows: consider two objects O_1 and O_2, a distance operator (D) and a threshold value (ϑ). O_1 is considered similar to O_2 if $D(O_1, O_2) < \vartheta$. More formally, let ς be a similarity comparison operator. $\varsigma(O_1, O_2) \Longleftrightarrow D(O_1, O_2) < \vartheta$. We will discuss in detail how to measure distances for trajectories and scenes.

5.3.3 Spatial extraction operation

The *spatial extraction* operation (denoted by γ) extracts the spatial component of a spatiotemporal relationship between objects. Note that when this operation is applied to a single object the result is the *trajectory* of the object. Note also the difference between this operation and the *projection* operation. The *projection* operation will return a scene while the *extraction* will return a relationship. Let A and B be symbols for objects in scene or set of scenes S. The following is the definition of spatial extraction.

Definition 14 *$S\ \gamma(A, B)$ is R such that R is the spatial component of a spatiotemporal relationship between A and B in scene S if S is a scene or a singleton set of scene. If S is a set of scenes then γ will iterate through S applying spatial extraction on each scene resulting in a set of spatial components of a spatiotemporal relationship between A and B.*

Note that if $A = B$ the operation can be written as $S\ \gamma(A)$ which will retrieve the *trajectory* of A if A is a moving object.

5.3.4 Temporal extraction operation

The *temporal extraction* operation (denoted by ω) extracts the interval component of a spatiotemporal relationship between objects.

Let A and B be symbols for objects in scene S and R be a spatial relationship. The following is the definition of temporal extraction.

Definition 15 *$S\ \omega(A, B, R)$ is I such that I is the temporal interval between A and B on a 2D-PIR sequence R in scene or a singleton set of scene S. If S is a set of scenes then ω will iterate through S, applying temporal extraction on each scene resulting in a set of intervals between A and B on the sequence of spatial relationships R.*

Note that a sequence of spatial relationships can be specified using a shorthand notation that will be discussed in the next operation. If R is not specified (ie. $S\ \omega(A, B)$), the operation will return the whole interval, that is from starting time to the end. If $A = B$ the operation can be written as $S\ \omega(A, R)$ or $S\ \omega(A)$

5.3.5 Spatial matching operation

The spatial matching operation (denoted by η) is useful when retrieving a scene by comparing the spatial part of spatiotemporal relationships between objects. Let L_1 and L_2 be two sequences of spatial relationships of the form $[(\delta_1, \chi_1, \psi_1), (\delta_2, \chi_2, \psi_2), ...,$ $(\delta_n, \chi_n, \psi_n)]$, the following is a simple definition of this operation.

Definition 16 *L_1 $\eta(L_2)$ is true if L_1 matches with L_2 or false otherwise.*

Matches between L_1 and L_2 are defined as follows:

1. $L_1 = L_2$ is called a *strong match,*
2. If one or more 2D-PIR component of L_1 or L_2 contains "don't care" relationship and $L_1 = L_2$ with "don't care" component, is called a *weak match.*

We use "?" for the "don't care" symbol in spatial relationships. For example, we can represent "object A enters object B" using the following 2D-PIR sequence, $[(dt, ?, ?), (to, m, d),$ $(ov, o, d), (ic, d, d)]$, if we "don't care" about the direction from which A enters B. Another example is a 2D-PIR sequence representing "object A enters B from the left side of B": $[(dt, <, ?), (to, m, d), (ov, o, d), (ic, d, d)]$. Here we "don't care" about the above-below direction (ψ). That is, as long as A enters B from the left the specification "A enters B from the left" is satisfied).

For query specification purposes we use the first, intermediate and the last elements of a sequence of 2D-PIR for matching. For example instead of using the long sequence $[(dt, ?, ?), (to, m, d),$ $(ov, o, d), (ic, d, d)]$ we can use $[(dt, ?, ?)..(ic, d, d)]$ to specify enters. A 2D-PIR sequence can be specified as [2D-PIRf .. 2D-PIRm .. 2D-PIRl] where 2D-PIRf and 2D-PIRl is the first and the last elements of a 2D-PIR sequence and 2D-PIRm is an element between the first and the last element of 2D-PIR sequence. If 2D-PIRm is empty (that is, [2D-PIRf .. 2D-PIRl]) then it specifies the first and the last element of a 2D-PIR sequence. We can use this shorthand since we can reason the whole sequence using the moving topological and interval neighbourhood graph which will be discussed later.

5.3.6 Temporal operations

We use Allen's interval-based relationships [3] for temporal operations. These operations are necessary when we need to test temporal relationships between events. For example, in the query "find a scene where A enters B during C passes B", a temporal operation *during* is needed to test temporal relationships between "enters" and "passes". In fact these operations are very useful to answer queries that involving many time relationships among events created by the moving objects in a scene.

Another useful temporal operation is *dur* that computes the duration of a time interval. Given a time interval I, I *dur* will return the duration of interval I(note that I can be the interval of the whole scene).

5.3.7 Overlay operation

The overlay operation (denoted by Ω) overlays two different scenes to form a new scene. Let $S_1(M_1, R_1)$ and and $S_2(M_2, R_2)$ be two scenes and τ be time delay after S_1 starts and before S_1 finishes. $S_1\ \Omega(S_2, \tau) = S(M, R)$ where $M = M_1 \Omega M_2$ and R is a set of relationships among objects in M.

Note that τ is a synchronisation tool between S_1 and S_2. If $\tau = 0$ then S_1 and S_2 start at the same time. If $\tau = t$ then S_2 starts t unit time after S_1 starts.

Overlay is a powerful operation that can create a more complex scene based on existing scenes. Using this operation, or combining it with the projection operation, a user can create new scenes based on existing scenes. This is particularly useful in animation or video systems. Figure 5.6 illustrates this operation.

5.4 Scene Retrieval

Scene retrieval allows a user to find a scene from a video-clip database by comparing it with a known scene. A simple example of scene retrieval is to find a moving object in a scene whose trajectory is the same as or similar to the user trajectory sketched in a window. We indicated how this query might be answered in the previous section.

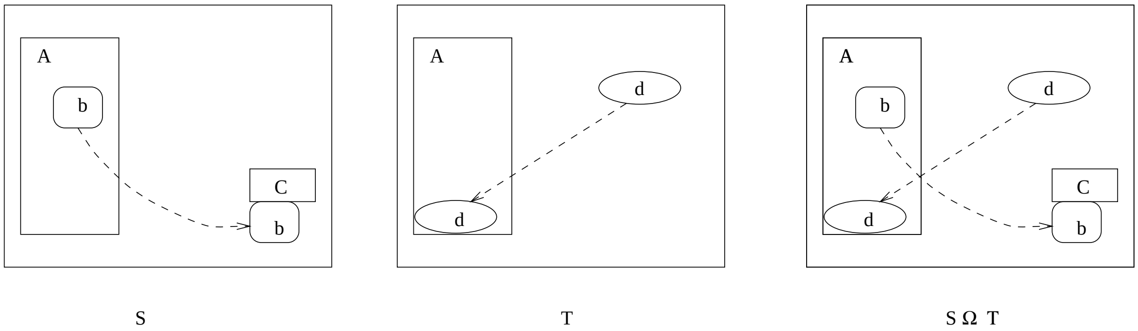


Figure 5.6: Scene S overlays Scene T

5.4.1 Scene construction

A scene can be considered as a sequence of symbolic pictures (this is analogous to the real world where a moving picture is a sequence of frames). Note that a symbolic picture is a connected labelled 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 labelled by 2D-PIR relationships. To construct a scene, we choose the first symbolic picture in the scene and construct a graph based on left-right, top-bottom arrangement on objects in the scene and then iteratively construct the scene graph based on the first graph. The following algorithm constructs a scene graph.

Algorithm: scene graph construction

input: A preprocessed moving picture (all objects are recognised and all
necessary information such as the current position of an object, topological
and projection relationships at each key frame have been computed)

output: A directed graph (adjacency list) represents the scene

begin

(1) find the first element (E): the first frame of a scene.
Arrange objects in E from left to right and top-bottom

(2) let P be an ordered set of objects in E based on the arrangement in 1

(3) for each object O_i in P

(4) attach the object with a sequence of st-relationship: $[(d, r, I)]$
where $d = 0, r = 0$ and $I = (t_s, t_s)$, note that t_s is the start time

(5) create a list containing the sequence of TS-2D-PIR relationships: $[(\delta, \chi, \psi, I)]$

where $I = (t_s, t_s)$ in such a way that all objects follows O_i in the arrangement from (1)

(6) endfor

(7) for each of the rest elements (key frames) in the scene (X)

(8) Update graph edges, that is $[(d, r, I)]$ and $[(\delta, \chi, \psi, I)]$ accordingly

(9) if a new object appears then

(10) add this object to the list objects in the scene

(11) repeat step 4

(12) for each existing objects (other than the new object) X

(13) add the new object containing TS-2D-PIR relationships: $[(\delta, \chi, \psi, I)]$ onto the tail of the corresponding list

(14) endfor

(15) endif

(16) endfor

end.

The preprocessor will decide several key frames from a scene. The first frame of a scene will be a key frame. The next key frame is chosen by scanning every k frames (where k is a small constant, for example k = 30 or jump to the most significant frame (directed by the scene change)). The preprocessor should be "smart" enough to detect disappearing objects during scanning. If, for example, an object A is detected at frame i and this object disappears at frame i+k, then the frame i + k/2 must be chosen as a key frame. The scanning processes will start from frame i + k/2 if the object A is detected in this frame. Otherwise, another key frame based on the above mechanism must be chosen (frame i + k/4). This will give us an approximation of the track for each moving object and we can generate the original TS-2D-PIR using both a *moving topological* and *moving interval* neighbourhood graph which will be discussed in the next subsection.

Figure 5.7 is an example of a scene containing four objects; two of which are moving objects. The other two "objects" C and D are regions. At a certain time (say t_s) object A starts moving. Object B is not moving until it is touched by object

A. Object A touches object B and brings it into region D and releases it. This scene could represent a person moving from a region (C) and touching a handcart outside region C and then pulling this cart to region D. While the person is inside region D he/she throws the cart somewhere but the cart is still inside region D.

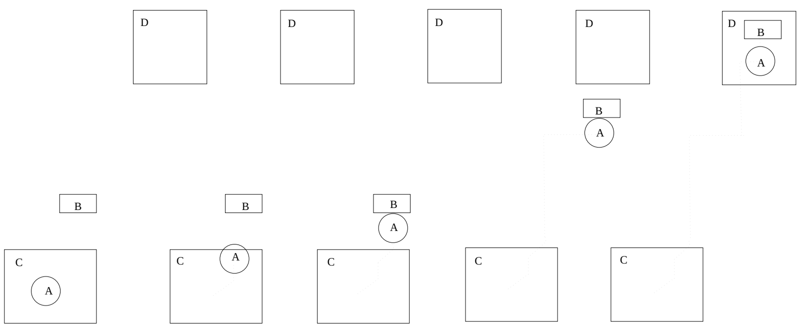


Figure 5.7: A scene of moving objects

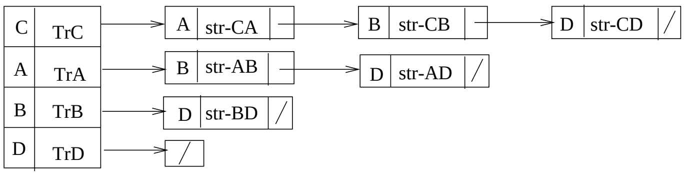
Figure 5.8 is the corresponding symbolic scene (scene graph) of Figure 5.7 constructed by the algorithm described above.

Note that TrC and TrD are still objects, therefore they do not have trajectories, $([(0, 0, I)])$. Consequently, str-CD is a singleton sequence, indicating that the spatial relationship between C and D is constant over time.

5.4.2 Moving topological and interval neighbourhood graphs

As discussed previously, we can use moving topological and interval relationships to generate the original TS-2D-PIR of a scene. In addition, these graphs is also useful for expanding a shorthand 2D-PIR sequence used in a specification. Therefore we will discuss moving topological and interval neighbourhood graphs and their constructions.

Both moving topological and interval neighbourhood graphs are useful in reasoning about changes of topological and interval relationships between moving objects during motion. Given two topological/interval relationships between objects, the



TrC $= [(0, 0, I)]$
TrA $= [(d_{a1}, 2, I_{a1}), (d_{a2}, 3, I_{a2}), (d_{a3}, 2, I_{a3}), (d_{a4}, 3, I_{a4}), (d_{a5}, 1, I_{a5}),$
$(d_{a6}, 3, I_{a6}), (d_{a7}, 1, I_{a7})]$
TrB $= [(d_{b1}, 3, I_{b1}), (d_{b2}, 1, I_{b2}), (d_{b3}, 3, I_{b3}), (d_{b4}, 1, I_{b4}), (d_{b5}, 3, I_{b5})]$
TrD $= [(0, 0, I)]$
str-CA $= [(ic, di, di, I_{ca1}), (ic, di, fi, I_{ca2}), (ov, di, oi, I_{ca3}), (to, di, m, I_{ca4}), (dt, di, <, I_{ca5}),$
$(dt, fi, <, I_{ca6}), (dt, o, <, I_{ca7}), (dt, m, <, I_{ca8}), (dt, <, <, I_{ca9})]$
str-CB $= [(dt, fi, <, I_{cb1}), (dt, o, <, I_{cb2}), (dt, m, <, I_{cb3}), (dt, <, <, I_{cb4})]$
str-CD $= [(dt, <, <, I_{cd})]$
str-AB $= [(dt, <, <, I_{ab1}), (dt, m, <, I_{ab2}), (dt, o, <, I_{ab3}), (dt, s, <, I_{ab4}), (dt, d, <, I_{ab5}),$
$(to, d, m, I_{ab6}), (dt, d, <, I_{ab7})]$
str-AD $= [(dt, <, <, I_{ad1}), (dt, m, <, I_{ad2}), (dt, o, <, I_{ad3}), (dt, s, <, I_{ad4}), (dt, d, <, I_{ad5}),$
$(to, d, m, I_{ad6}), (ov, d, o, I_{ad7}), (ic, d, s, I_{ad8}), (ic, d, d, I_{ad9})]$
str-BD $= [(dt, <, <, I_{ad1}), (dt, m, <, I_{ad2}), (dt, o, <, I_{ad3}), (dt, s, <, I_{ad4}), (dt, d, <, I_{ad5}),$
$(to, d, m, I_{ad6}), (ov, d, o, I_{ad7}), (ic, d, s, I_{ad8}), (ic, d, d, I_{ad9})]$

Figure 5.8: Scene graph of Figure 5.7

moving topological/interval neighbourhood graph has the ability to describe intermediate topological/interval relationship(s) between the moving objects.

This feature is useful in speeding up scene construction from a sequence of frames (images). That is, it is not necessary to check every frame to construct a scene since if we compute topological and interval relationships between two frames, then we can reason about topological and interval relationships in other frames between the two frames.

Moving topological neighbourhood graph

Definition 17 *Two topological relationships between two spatial objects are moving-neighbours if they can be directly transformed into one another by continuously moving the objects.*

For example *disjoint* and *touch* are moving-neighbour since they can be transformed to the other by continuously moving an object towards the other or by both objects approaching each other until they touch.

In order to construct moving-topological neighbourhood graph, we consider the following three cases.

1. Case 1: There are two objects A and B. A is larger than B. Initially A and B are disjoint and A moves according to Figure 5.9

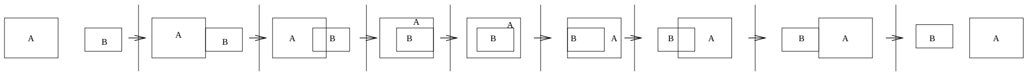


Figure 5.9: Case 1

2. Case 2: There are two objects A and B. A is the same size as B. Initially A and B are disjoint and A moves according to Figure 5.10

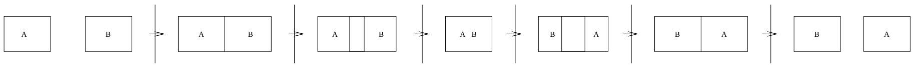


Figure 5.10: Case 2

3. Case 3: There are two objects A and B. A is smaller than B. Initially A and B are disjoint and A moves according to Figure 5.11

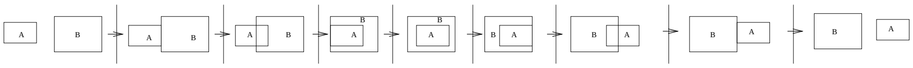


Figure 5.11: Case 3

In case 1, the sequence of topological relationships is *disjoint, touches, overlaps, covers, contains, covers, overlaps, touches, disjoint.* In case 2 the sequence of topological relationships is *disjoint, touches, overlaps, equals, overlaps, touches, disjoint.* In case 3, the sequence of topological relationships is *disjoint, touches, overlaps, covers, inside, coveredBy, overlaps, touches, disjoint.*

Based on the above cases the moving-topological neighbourhood graph shown in Figure 5.12 is constructed.

Since we can collapse the topological relationships *covers, coveredBy, contains* and *inside* to one topological relationship *inclusion* in terms of 2D-PIR, the above graph can be simplified as shown in Figure 5.13.

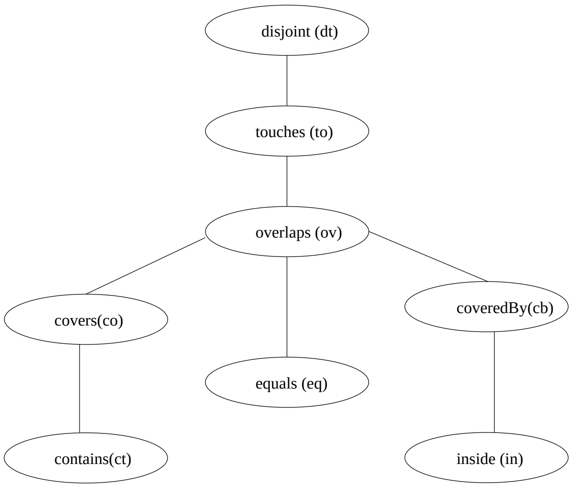


Figure 5.12: Moving topological neighbourhood graph

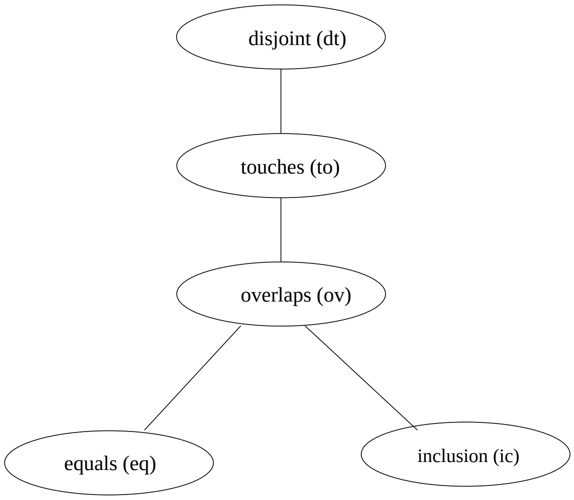


Figure 5.13: Moving basic topological neighbourhood graph

Moving interval neighbourhood graph

Definition 18 *Two interval relations are moving-neighbours if they can be transformed into one another by continuously moving the intervals.*

The method for constructing the interval neighbourhood graph is similar to the method for constructing the moving-topological neighbourhood graph. The interval neighbourhood graph is derived in the same way as *B-neighbours* in Freksa [35]. Figure 5.14 shows the moving-interval neighbourhood graph.

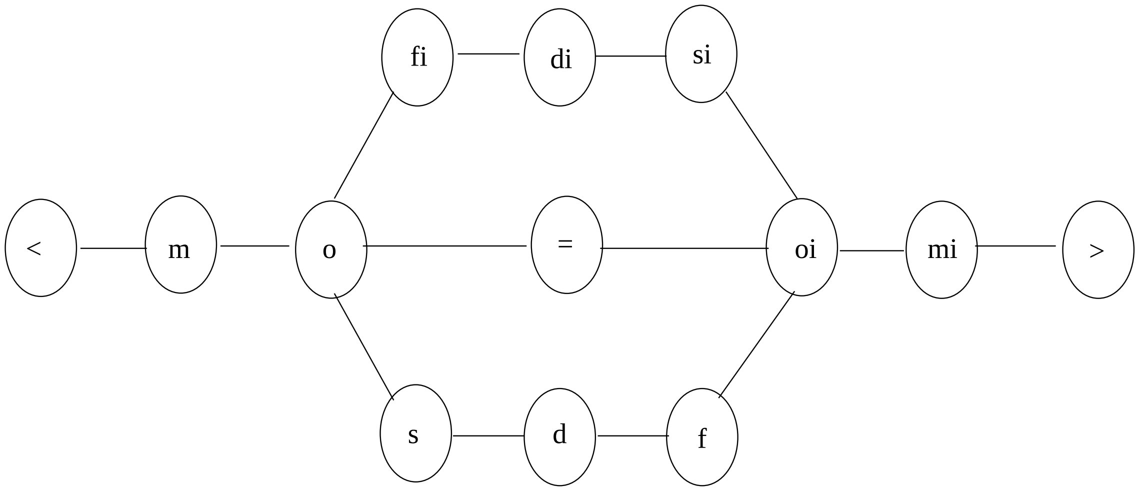


Figure 5.14: Moving-interval neighbourhood graph

5.4.3 Scene matching

Unlike picture matching, scene matching has not been widely discussed in the literature, primarily because no data models for scenes have been developed. Since we extend the symbolic picture to model a symbolic scene, then our method for picture matching can be modified in such a way that it also works for scene matching. In fact, our algorithm for scene matching is quite similar to our algorithm for picture matching, the only difference is that the similarity computation is more complex in scene matching, since each list contains a sequence of st-relationships. So, we will not discuss this algorithm further, instead we will discuss in detail how to compute similarity between scenes.

5.4.4 Similarity functions for scenes

The degree of similarity between two scenes represented using scene graphs labelled by a sequence of st-relationships is dependent on the degree of similarity between their corresponding st-relationships. There are two major types of st-relationships, and each type needs different treatment when measuring similarity. The first type of st-relationships are the st-relationships created when an object moves (which is in the form of $[(d, r, I)]$ or *track*). This relationship exists only for moving objects. The second type of st-relationship is TS-2D-PIR created among objects in a scene, that is

$[(\delta, \chi, \psi, I)]$. In this work we are interested in the spatial aspects of *st*-relationships, since this is used in measuring the similarity of moving objects.

Trajectory similarity function

The key idea in measuring "movement similarity" is to measure the *distance* between two trajectories. Since a trajectory is modelled as a sequence of distance-direction pair (i.e. $[(d_1, r_1), (d_2, r_2), \ldots, (d_n, r_n)]$, the distance between two trajectories is affected by the distances among the elements of the trajectories.

Let r_a and r_b be two directions in {1,2,3,4,5,6,7,8}. The distance between r_a and r_b, $D(r_a, r_b)$ is defined by a minimum arc between r_a and r_b. For example D(1,2) = 45 and D(1,8) = 45 and D(1,4) = 135.

Now, we derive the similarity between r_a and r_b. The relationship between distance and similarity is that, if the distance between r_a and r_b is maximum then the similarity between r_a and r_b is 0 (totally different) and if the distance between r_a and r_b is 0 then the similarity between r_a and r_b is 100 (r_a same as r_b). The following formulae defines the similarity between r_a and r_b. Note ς is a symbol for similarity.

$$\varsigma(r_a, r_b) = 100 \times \frac{Max\ D(r_a, r_b) - D(r_a, r_b)}{Max\ D(r_a, r_b)}$$

Let A and B be two distance-direction pairs (d_a, r_a) and (d_b, r_b). The similarity between A and B, $\varsigma(A, B)$, is defined by:

$$\varsigma(A, B) = Min(d_a, d_b) \times \varsigma(r_a, r_b)$$

Let $T_1 = [(d_{p1}, r_{p1}), (d_{p2}, r_{p2}), \ldots, (d_{pn}, r_{pn})]$ and $T_2 = [(d_{t1}, r_{t1}), (d_{t2}, r_{t2}), \ldots, (d_{tm}, r_{tm})]$ be two trajectories. Assume the sequence T_2 is longer than the sequence T_1 such that T_2 contains k number of substring $l_1, l_2, \ldots, l_k$ whose lengths are the same or less than T_1. The head of a trajectory is its first element. The tail of a trajectory is its last element. Other elements of a trajectory are its body. Imagine

T_1 and T_2 are flattened in a plane. T_1 *overlaps* T_2 if the head of T_1 is located on the left of head of T_2 and the tail of T_1 is located on the right of head of T_2. T_1 *in* T_2 if the head of T_1 is located on the same position or right of head of T_2 and the tail of T_1 is located on the same position or left of tail of T_2. T_1 *overlapped by* T_2 if the head of T_1 is located on the right of head of T_2 and the tail of T_1 is located on the right of tail of T_2.

The similarity between T_1 and T_2 is defined by the following formulae.

For T_1 overlapping T_2, $S(T_1, T_2)$ is computed as follows

$$S_1 = Max\left(\frac{\sum_{(i,j)=(1,k)}^{(k-1,1)} \varsigma((d_{pi}, r_{pi}), (d_{tj}, r_{tj}))}{\sum_{i=1}^{n} d_{pi}}, \quad \forall\, k \mid k \;=\; 1 \cdots n\right)$$

For T_1 *in* T_2,$S(T_1, T_2)$ is computed as follows

$$S_2 = Max\left(\frac{\sum_{(i,j)=(k,k)}^{(n+k,n+k)} \varsigma((d_{pi}, r_{pi}), (d_{tj}, r_{tj}))}{\sum_{i=1}^{n} d_{pi}}, \quad \forall\, k \mid k \;=\; 1 \cdots m-n+1\right)$$

For T_1 overlapped by T_2, $S(T_1, T_2)$ is computed as follows

$$S_3 = Max\left(\frac{\sum_{(i,j)=(m-n+2,1)}^{(m,k)} \varsigma((d_{pi}, r_{pi}), (d_{tj}, r_{tj}))}{\sum_{i=1}^{n} d_{pi}}, \quad \forall\, k \mid k \;=\; n-1 \cdots 1\right)$$

$$\varsigma(T_1, T_2) = Max(S_1, S_2, S_3)$$

One exception is that if both T_1 and T_2 are straight lines then the similarity between T_1 and T_2 is equal to $\varsigma(r_a, r_b)$.

These formulae have important properties, which are described by the following theorems.

Theorem 4 *$\varsigma(T_1, T_2)$ is scale invariant.*

Proof: Let T_1 and T_2 be two trajectories where T_1 is taken by scaling T_2. Let us assume T_1 is a smaller scale of T_2, then the length of the sequences T_1 and

T_2 are the same, since they contain the same sequence of directions. Let T_1 be $[(d_{p1}, r_{x1}), (d_{p2}, r_{x2}), \ldots, (d_{pn}, r_{xn})]$ and T_2 be $[(d_{t1}, r_{x1}), (d_{t2}, r_{x2}), \ldots, (d_{tn}, r_{xn})]$. $\varsigma(T_1, T_2)$ will be maximum if T_1 *in* T_2.

$$\varsigma(T_1, T_2) = \frac{\sum_{(i,j)=(1,1)}^{(n,n)} S((d_{pi}, r_{xi}), (d_{tj}, r_{xj}))}{\sum_{i=1}^{n} d_{pi}}$$

since T_1 and T_2 has the same length.

$$\varsigma(T_1, T_2) = \frac{\sum_{i=1}^{n} min(d_{pi}, d_{ti}) \times 100}{\sum_{i=1}^{n} d_{pi}}$$

since $S(xi, xj) = 100 \; \forall$ i = j

$$\varsigma(T_1, T_2) = \frac{\sum_{i=1}^{n} d_p i \times 100}{\sum_{i=1}^{n} d_{pi}}$$

since $d_p i < d_t i$ (T_1 is small scale of T_2)

$$\varsigma(T_1, T_2) = 100$$

which means $T_1 = T_2$. □

An average direction of a trajectory T = $[(d_1, r_1), (d_2, r_2)...(d_n, r_n)]$ is computed using the following formulae.

$$\overline{T} = \frac{\sum_{i=1}^{n} d_i \times r_i}{\sum_{i=1}^{n} d_i}$$

Theorem 5 *If* $T_1 = T_2$ *then* $\overline{T_1} = \overline{T_2}$.

Since $T_1 = T_2$ then both T_1 and T_2 have exactly the same sequence of distance-direction. Thus, their average direction must be equal.

However, if $\overline{T_1} = \overline{T_2}$ then it is not necessary that $T_1 = T_2$. Let T_1 be [(12,2)] and T_2 be [(4,3),(4,2),(4,1)]. $\overline{T_1} = \overline{T_2}$ but $T_1 \neq T_2$.

Using an analogy, we can use Theorem 5 to approximate the similarity between two trajectories as follows: if two trajectories are similar then the distance between their average trajectories is less or equal to a threshold. Note that if the distance between two average trajectories is less or equal to the threshold, then the trajectories are not necessarily similar. We will show that this analogy is a good approximation in a set of experiments, which will be discussed in the next chapter.

The application of this analogy is to aid filtering a query on trajectory. For example, if the average difference between two trajectories is greater than the threshold then we simply report that the trajectories are considered not similar. However, if the average difference between two trajectories is less or equal to the threshold we need to check further whether they are similar or not using the above similarity formulae.

Average trajectories can also be used as a method for clustering trajectories. For example we can cluster trajectories whose average are between 1 and 2, between 2 and 3 and so on. Finding similar trajectories is directed to its cluster.

Multiple moving objects similarity function

To measure the similarity of two collections of moving objects, we need to measure the similarity between their corresponding TS-2D-PIR. However, we need only to compare spatial part of TS-2D-PIR, a sequence of the form $[(\delta, \chi, \psi)]$, which is nothing more than a sequence of 2D-PIR. We first of all need to measure the similarity between two $(\delta_1, \chi_1, \psi_1)$ and $(\delta_2, \chi_2, \psi_2)$, which was defined in Chapter 3.
Let

$$ST_1 = [(\delta_{p1}, \chi_{p1}, \psi_{p1}), (\delta_{p2}, \chi_{p2}, \psi_{p2}), \ldots, (\delta_{pn}, \chi_{pn}, \psi_{pn})]$$

and

$$ST_2 = [(\delta_{t1}, \chi_{t1}, \psi_{t1}), (\delta_{t2}, \chi_{t2}, \psi_{t2}), \ldots, (\delta_{tn}, \chi_{tn}, \psi_{tn})]$$

be the spatial components of *st*-relationships.

Recall (from Chapter 3) that the similarity of two 2D-PIR relationships r_1 and r_2 is defined using the following formulae:

$$\varsigma(r_1, r_2) = 100.0 \times ((20.785 - D(r_1, r_2))/20.785)$$

The similarity between ST_1 and any ST_2 is defined by:

$$\varsigma(ST_1, ST_2) = \frac{\sum_{i=1}^{n} \varsigma(r_{pi}, r_{ti})}{n}$$

where i = 1..n.

Scene Similarity

Having defined the similarity between trajectories of moving objects and the similarity between spatiotemporal relationships of multiple moving objects, we can now define the similarity between scenes.

Recall that a scene is a connected labelled digraph $S(M, R)$ where M is finite non-empty set of symbols in the scene and R is a set of edges labelled by a sequence of spatiotemporal relationships (st-relationship) between objects. Let $S_1(M_1, R_1)$ and $S_2(M_2, R_2)$ be two scenes. Assume $M_1 \subseteq M_2$. The distance between S_1 and S_2 is the sum of the corresponding *st*-relationships in R_1 and R_2. More formally, let $R_1 = AT_1 \cup AST_1$ and $R_2 = AT_2 \cup AST_2$ where $AT_1 = \{T_{O_1.S_1}, T_{O_2.S_1}, \cdots, T_{O_m.S_1}\}$ and $AT_2 = \{T_{O_1.S_1}, T_{O_2.S_2}, \cdots, T_{O_n,S_2}\}$ are sets of all trajectories of all moving objects in S_1 and S_2 respectively, and $AST_1 = \{ST_{S_{1.1}}, ST_{S_{1.2}}, \cdots, ST_{S_{1.n}}\}$ and $AST_2 = \{ST_{S_{2.1}}, ST_{S_{2.2}}, \cdots, ST_{S2.n}\}$ are sets of all spatial components of the sequence of *st*-relationships (TS-2D-PIR) in S_1 and S_2 respectively such that $S_{1.i}$ and $S_{2.i}$ are the corresponding *st*-relationships. The distance between S_1 and S_2 is defined as:

$$D(S_1, S_2) \ = \ w_1 \sum_{i=1}^{m} D(T_{S1.Oi}, T_{S2.Oi}) + w_2 \sum_{i=1}^{m} D(ST_{S1.i}, ST_{S2.i})$$

where w_1 and w_2 are weights corresponding to the importance of trajectory and spatial component of TS-2D-PIR.

5.5 Moving Object Query Language

We propose a query language called MOQL (Moving Object Query Language) as a tool for specifying queries on moving objects. The language follows object-oriented message passing style.

RECEIVER MESSAGE-NAME (ARGUMENTS)

The sematic are as follows. A receiver can be an object such as a scene database, a scene, or an expression which can produce an object. Message-name is the name of an operation, which is defined in the receiver of the message. Arguments are set of arguments which are required by operations on moving object.

The specification of the language is described using the following BNF-like notation. Note that any symbol between semicolons is a terminal symbol.

THE QUERY LANGUAGE

Query = ProjectionQuery | OverlayQuery | ExtractionQuery | SelectionQuery
NameOfAScene = Identifier "is" Query
projectionQuery = SceneName "project" "(" ProjArgs ")"
ProjArgs = ObjectName | ObjectName Interval
OverlayQuery = OverlayOperand "overlay" "(" OverlayOperand "," TimeDelay ")"
OverlayOperand = SceneName | NameOfAScene | ProjectionQuery
TimeDelay = IntegerConst | OverlayOperand "dur"
ExtractionQuery = SpatialExtraction | TemporalExtraction |
    TemporalExtraction "dur"
SeletionQuery = TargetObject ":" Itvar "where" "(" Predicate ")"
TargetObject = SceneDatabase | SelectionQuery
Predicate = SimilarityPredicate | ExactPredicate
SimilarityPredicate = SceneSpec "similarTo" "(" SceneSpec ")"
ExactPredicate = ContainmentPredicate | SpatialPredicate | TemporalPredicate
ContainmentPredicate = SceneSpec "contains" ObjectName |
    SceneSpec "contains" "{" ObjectName { ", " ObjectName} "}"
SpatialPredicate = SpatialRelSpec "match" "(" SpatialRelSpec ")"
SpatialRelSpec = StpirRel | StpirRelShort | SpatialExtraction
StpirRel = "[" TwoDpir {"," TwoDpir} "]"

StpirRelShort = “[” TwoDpir {“..” TwoDpir} “]”

TwoDpir = “(” TopoRel “,” IntRel “,” IntRel “)”

TopoRel = “dt” | “to” | “ov” | “in” | “co” | “cb” | “ct” | “eq” | “?”

IntRel = “b” | “a” | “m” | “mi” | “o” | “oi” | “d” | “di” | “s” | “si” | “f” | “fi” |
“eq” | “?”

SpatialExtraction = SceneSpec “extractS” “(” SpArgs “)”

SpArgs = ObjectName | ObjectName “,” ObjectName

TemporalPredicate = IntervalObj IntRel IntervalObj | Duration RelOp Duration

IntervalObj = Interval | TemporalExtraction

Duration = IntegerConst | IntervalObj “dur”

TemporalExtraction = SceneSpec “extractT” “(” TArgs “)”

TArgs = ObjectName ”, ” ObjectName | ObjectName “,” ObjectName “,” SqPir

SqPir = StpirRel | StpirRelShort

RelOp = “=” | “<” | “>” | “<=” | “>=” | “!=”

SceneSpec = SceneName | Itvar | ProjectionQuery

SceneDatabase = Identifier

Itvar = Identifier

SceneName = Identifier

ObjectName = Identifier

Interval = IntegerConst IntegerConst

Identifier = Letter { NextSysbols }

NextSymbols = Letter | Digit

IntegerConst = Digit { Digit }

Digit = “0” | “1” | “2” | “3” | “4” | “5” | “6” | “7” | “8” | “9”

Letter = any English alphabet characters

5.6 Summary

A moving object is modelled as a pair $(symbol, track)$. The symbol uniquely identifies the object and the *track* describes how the object moves. We then consider modelling several moving objects in a scene. A scene is a graph where the nodes represent the objects, the loops in the graph are labelled by tracks, and edges are labelled by spatiotemporal relationships represented using TS-2D-PIR.

We have presented several operations on moving objects and scenes which are useful for retrieving scenes of moving objects in a scene database.

A scene graph is constructed from a sequence of pictures using a simple, efficient method. The scene similarity matching algorithm is quite similar to the picture similarity matching algorithm since picture and scene use the same model. The difference between picture similarity matching and scene similarity matching is in their similarity measures. Picture similarity matching uses the distance between 2D-PIR while scene similarity matching uses the distance between trajectories as well as between TS-2D-PIR.

Chapter 6

A Prototype Moving-Object Retrieval System

6.1 Introduction

To test the moving objects data model that we developed in the previous chapter, we have built a simple Java-based prototype of a moving-object retrieval system that implements the model and its operations. We have also implemented the scene similarity algorithm and conducted some retrieval experiments, especially on trajectory similarity. Since the model for moving objects based on extension of the model for images, the architecture of the prototype for moving objects retrieval system was developed by extending the architecture of the prototype of image retrieval system.

The results of experiments show that our trajectory similarity matching function works well on all types of trajectories.

It is hard to find comparable prototypes because it is a relatively new field of research. However, we will compare our model with the models of Dimitrova [26], Lee [69], Shearer [100], Sistla [101] and Bimbo [10].

In the following sections we will discuss a representation of the moving-object data model (track and TS-2D-PIR) using an extended *bi-array*, the system prototype, implementation techniques, experiments on trajectory similarity retrieval and compare our with with other models.

6.2 Representation of Moving-Object Data Model

Moving objects are composed in a *scene*. A *scene* is a connected labelled digraph $S(M, R)$ where M is a finite non-empty set of symbols of objects in the scene and R is a set of edges labelled by a sequence of spatiotemporal relationships (*st*-relationships). Each object is represented by its symbol (M) and can be a moving object or a still object. If all objects in a scene are still objects then the scene is a picture. Thus, a scene can be viewed as a generalisation of a picture and the structure that we use to represent an image can be extended to represent a scene.

We use a *bi-array* structure to represent a 2D-PIR graph. We will extend this structure to accommodate dynamic relationships among objects in a scene. Recall that a *bi-array* composes of a *symbol array* that contains a sequence of symbols and a *relationship array* that contains spatial relationships among object represented by the symbols in the *symbol array*. The *bi-array* structure needs to be extended to incorporate time and changing relationships since a moving object changes its position over time and possibly changes its spatial relationships with other objects in a scene. The extended *bi-array* structure will be discussed in detail in the subsequent section.

6.3 The System for Moving-Object Retrieval

The system's architecture is shown in Figure 6.1. The system consists of seven major components: user interface, scene analysis, query interpreter, symbolic scene transformer, insertion engine, retrieval engine and scene database.

6.3.1 User interface

The user interface is used to formulate queries and to generate scene representations either via development of an animation or through manual mark-up of objects on a real video, followed by automatic TS-2D-PIR generation.

In creating an animation, a user draws objects and their trajectories (if the

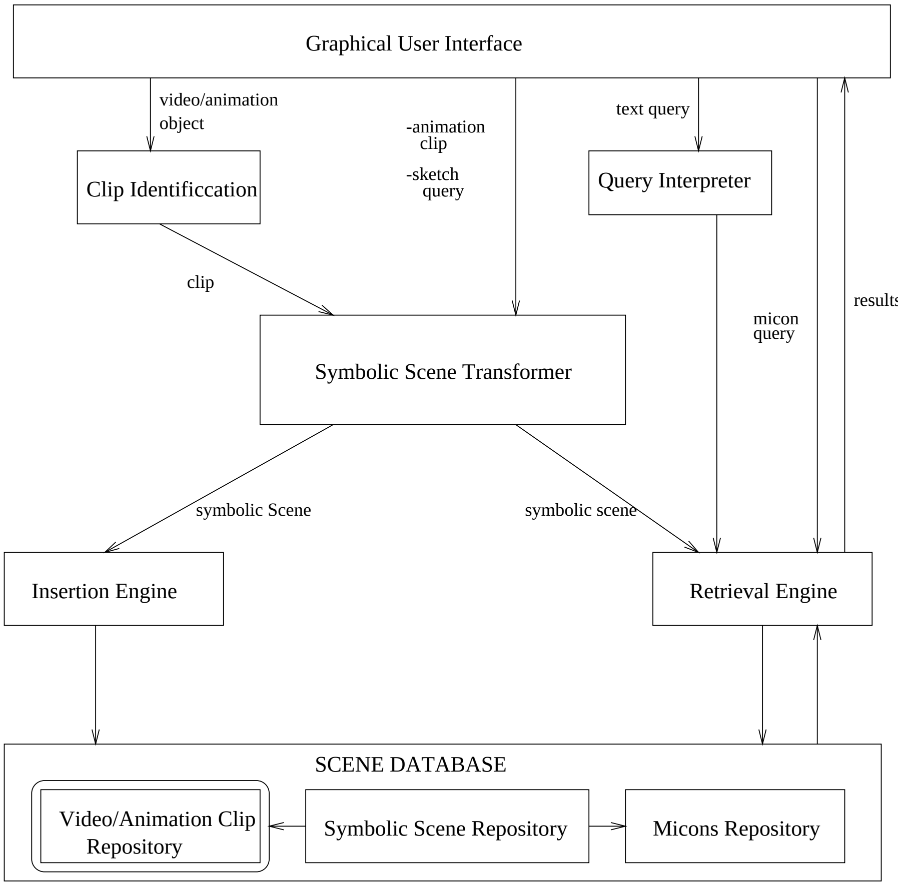


Figure 6.1: Architecture of The Moving Object Retrieval System

objects are moving objects) and plays this animation and then stores it in a database. The current prototype treats the speed of every moving object as the same constant value. This type of object movement is sufficient for our experiments.

In formulating queries, a user has two options: posing a query using a traditional query language or posing a query using a graphical query language. Using a graphical query language, a user may pose a query in the form of trajectory-based query (a user creates trajectory and ask the system to find scenes in a database that contains an object whose trajectory is similar to the user sketched trajectory) or clip-based query (a user sends the name of a scene or creates an animation and asks the system to find similar scenes).

In creating a symbolic scene from a real scene, the system interacts with a user to identify objects in the scene. This feature is not implemented in the current prototype; however, we will discuss the functionality of this feature in detail shortly.

The query interpreter interacts with a user if the user poses a query in a traditional text-based form. The interpreter validates the grammar of the query, interprets and sends the valid query to the retrieval engine.

6.3.2 Clip identification

"Real world" video objects such as movies or animations consist of a sequence of clips (A clip is a meaningful scenario which is composed of objects, some of them moving). A clip consists of a sequence of frames. A clip can be represented as a scene in our model.

Our intention is to store movies or animations based on their clips so that we can find a specific clip for a specific purpose. A clip is identified using its first and last frame number. A name associated with each clip can be assigned for high level identification.

The problem is that the majority of available video objects are *intact* video objects; That is the clips that compose them are not separated technically. Therefore we need a method to separate clips so that we can store the video object based on clips.

In general two methods can be applied to separate clips from video objects. The first and obvious method is a manual method. In a manual method a user is responsible for identifying all clips in the video objects and separating them as a set of clips. This method is tedious and consumes abundant amounts of time. A second method is an automatic clip identification method. One of these methods uses adjacent frame colour similarity as explained in [99, 80, 88]. The idea is that the colour histogram of a frame with its neighbour (previous and next frame) in a clip tend to be similar. If the colour histogram of a frame with its neighbour is not similar then the frame belongs to another clip. This method is quite successful and is used by [88] in identifying clips. There are other methods discussed in the literature. A good comparison of these methods can be found in [11].

The current prototype does not implement clip identification since it is beyond the scope of this thesis. This area of research is done in image processing.

6.3.3 Symbolic scene transformer

The major task of the *symbolic scene transformer* is to transform a video or animation clip to a symbolic scene. The transformer implements the scene graph construction algorithm. It accepts a video or animation clip and transforms it into a scene graph.

In transforming a real video clip, the transformer is aided by an image processing module and a user in identifying objects in the clip. This feature is not implemented in the current prototype, however, we will provide a brief explanation how this can be done.

The image processing module will identify objects contained in the scene starting from the first frame. This identification process is a non trivial problem. A fully automatic approach is not feasible using current technology, except for specific domains such as in video surveillance. Our approach requires manual intervention in segmenting and naming objects in frames. For the subsequent frames (the algorithm can jump to the next k frame as explained in the previous chapter), the processing of identification is repeated by comparing the similarity of objects in the current frame with objects in the previous frame. Once objects in the current frame are identified, the *track* of every object and spatiotemporal relationships among objects are incrementally computed.

In the current prototype, a user can create an animation clip and then send this clip to the transformer to create its corresponding scene graph. Since the objects are generated by the animation software, they can be automatically identified and TS-2D-PIR representation for the animation can be generated automatically.

6.3.4 Insertion engine

Once every object in a scene has been identified and their spatiotemporal relationships have been computed, the scene is transformed into a symbolic scene (in the form of a graph). The scene is stored in a scene database.

6.3.5 Retrieval engine

Retrieval of scenes from a scene database can be done through *trajectory-based*, *scene-based* or traditional query (using a query language) techniques.

Trajectory-based queries are performed by drawing a trajectory in a window. Next, a user may adjust a similarity threshold (if a similarity threshold is not adjusted, the system will use a default threshold, which is a "good" threshold based on experiments on trajectory similarity retrieval). The system will find scenes from the scene database that contain an object whose trajectory is similar to the user-sketched trajectory.

A scene-based query can be posed by a scene name or by an animated scene. If a user wants to retrieve scene similar to an existing scene, he/she can specify the name of an existing scene. The system also provides scene similarity retrieval based on a user animated scene as follows. Firstly, a user creates objects and specifies their trajectories by free-line drawing (if a trajectory is not given for an object, then the system will assume that the object is a static object). Next, the user plays the animation created to set up spatiotemporal relationships among objects. Finally, the user sends this animation as a query to the system. The system will create a corresponding scene graph of the animation and then find similar scenes from the scene database.

The system also supports a simple text-based query language to retrieve scenes. This type of query specification is quite similar to the query specification on traditional database systems.

The user is not only able to retrieve scenes from the database, but he/she can also create new scenes based on existing scene using *projection* and *union* operations. The easiest way to create a new scene is using the query language. However, a user may also use the graphical facility to create a new scene. For example, a user wants to overlay some objects of a certain scene stored in a scene database with his/her on-the-fly animated objects. For this purpose he/she may use projection operation on the intended scene and then performs an overlay with his/her animation.

6.3.6 Scene database

The scene database is partitioned into three parts: video/animation clips repository, where the original video or animation clips are stored, a miconic repository, where a motion icon (micon) version of the corresponding clip is kept, and the symbolic scene repository (which holds the scene graphs). Physically, we store only a pointer (URL) to the original clips. The micons are stored locally since they may be presented to the user in the first stage of the retrieval process. The symbolic scenes (scene graphs) are also stored locally, since they are central to the image retrieval process. Each symbolic scene contains:

- pointers to the original and micon versions of video/animation clips
- a set of symbols representing objects in the scene
- tracks of each object in the scene
- a set of spatiotemporal relationships among objects in the scene.

6.4 Design and Implementation Techniques

6.4.1 Structure

As in the previous prototype, we have built this prototype using the object-oriented approach, taking advantage of code reuse from the previous prototype. In this prototype we also use the concept of a pointset that has been successfully implemented in the previous prototype. We extended the *bi-array* structure to capture moving objects and spatiotemporal relationships among them.

Recall that the *bi-array* is a representation of a 2D-PIR graph. A *bi-array* consists of two dynamic arrays namely the *symbol array* and the *relationship array*. The *symbol array* contains names or concepts that identify each object in an image and the *relationship array* contains 2D-PIR relationships among the objects in an image.

To capture moving objects we extend the *symbol array* to a *moving bi-array* which contains not only the identifier (symbol) of the object but also the track of the object. The structure of the track is another *bi-array* which is composed of a *trajectory array* and a *time array*. Each element of the *trajectory array* is an integer where the last digit represents a direction of movement and the rest represent the length (distance) of the direction. The elements of the *time array* are integers that represent snapshots of time (in the prototype the time granularity is in milliseconds; however, time granularity is application dependent).

Let $d_1, d_2, ..., d_n$ be elements of the *trajectory array* and $t_1, t_2, ..., t_n$ be elements of the *time array*. The time interval that corresponds to the trajectory element d_i is $[t_{i-1}, t_i]$. Note that $t_0 = 0$. For example, the time interval corresponding to d_1 is $[0, t_1]$ and the interval corresponding to d_7 is $[t_6, t_7]$.

The *relationship bi-array* contains spatiotemporal relationships which are represented by another *bi-array*. This *bi-array* consists of a *sequence of 2D-PIR array* and a *time array*. The elements of the *sequence 2D-PIR array* are sequences of 2D-PIR that reflect changes in the 2D-PIR relationships between two objects over time.

Let $r_1, r_2, ..., r_n$ be elements of the *sequence 2D-PIR array* and $t_1, t_2, ..., t_n$ be element of the *time array*. The time interval corresponding to r_i is $[t_{i-1}, t_i]$. The pair $(r_i, [t_{i-1}, t_i])$ is a spatiotemporal relationship between objects.

Consider Figure 6.2. This is a scenario of three objects, A, B and C. A and C are moving objects and B is a still object. The movement of A and C are represented by dotted lines (trajectories).

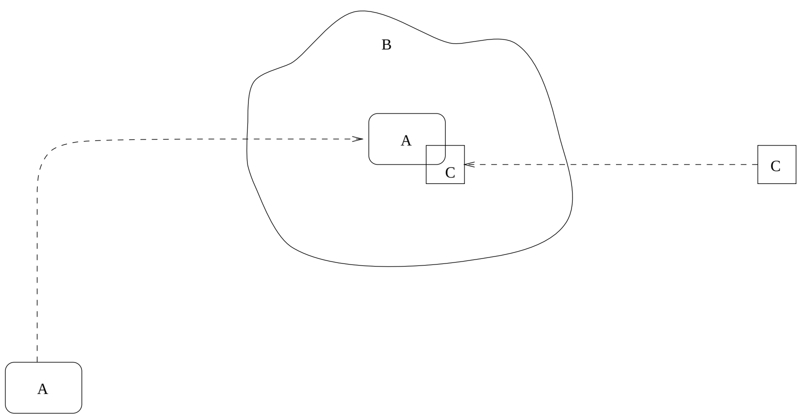


Figure 6.2: A and C enters B.

Figure 6.3 depicts the scene graph and Figure 6.4 shows *extended bi-array* representation of Figure 6.2

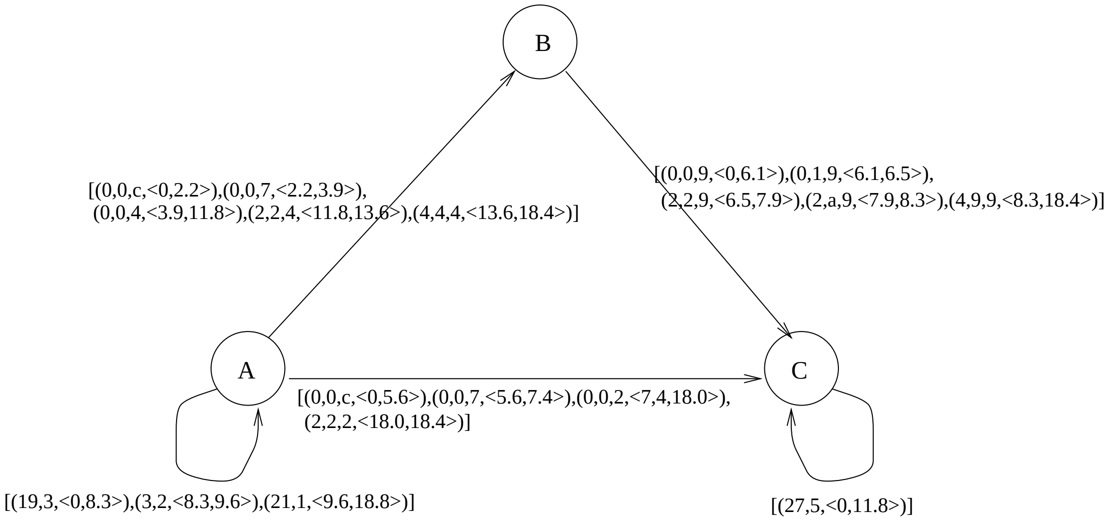


Figure 6.3: Scene graph of Figure 6.2

In Figure 6.3, node B has no loop since object B is a still object. Both node A and C have loops since A and B are moving objects. Each label in a loop is a sequence that represents the track of the corresponding object in the node. Each element of the sequence is composed of distance travelled, direction, and time interval. For example, the first element of the track A is $(19, 3, < 0, 8.3 >)$ which means A moves along direction 3 for 19 units during interval $< 0, 8.3 >$. Here the interval is represented by start time (0) and finish time (8.3). Note that the time granularity here is in seconds. Each edge between two nodes is labelled by a sequence that represents a spatiotemporal relationship between objects represented by the nodes. Every element of this sequence is composed of basic topological relationships.

MA:

A	B	C
[193,32,211]	[ ]	[275]
[8275, 9587, 18812]	[ ]	[11788]

RA:

[(0,0,c),(0,0,7),(0,0,4),(2,2,4),(4,4,4)]	[(0,0,c),(0,0,7),(0,0,2),(2,2,2)]	[(0,0,9),(0,1,9),(2,2,9),(2,a,9),(4,9,9)]
[2163, 3897, 11794, 13553, 18429]	[5621, 7416, 17967, 18429]	[6072, 6507, 7851, 8275, 18429]

Figure 6.4: Extended bi-array representation of figure 6.2; MA is the *moving bi-array*, RA is the *relationship bi-array*

The contents of the extended bi-array on Figure 6.4 are created based on the animation of Figure 6.2 using the prototype. If one looks carefully at the *sequence 2D-PIR array* of the *relationship array*, sometimes the sequence does not includes a 2D-PIR relationships containing *touches* or *meets* relationships when it logically should. For example, the *sequence 2D-PIR array* representing relationships between A and B : [(0,0,c),(0,0,7),(0,0,4), (2,2,4),(4,4,4)]. The third and the fourth element of this sequence are (0,0,4) and (2,2,4) respectively. During motion, initially A is logically disjoint, then touches and overlaps with B. However, (0,0,4) and (2,2,4) represent disjoint and overlap. The touches relationship is skipped, because in this case the touch relationship holds only an instant, while the prototype creates the animation on 30 frames/second. During scene matching, this problem will be corrected using the *topological* and the *interval neighbourhood* graphs discussed in the previous chapter.

6.4.2 Query facilities

There are two options for querying: by animation sketch (that is scene-based and trajectory-based) or by traditional text-based language. We provide these options since some queries are easier to express by sketching and others are easier to express using MOQL (Moving Object Query Language). Query by sketching is particularly good for similarity-matching retrieval and query based on a particular language is good for exact-matching and for creating new scenes based on existing scenes.

6.5 Experiments on Trajectory Matching

Experiments were conducted on trajectory similarity retrieval. To test our similarity measure on trajectory we create 88 different trajectories.

We measured the performance of retrieval based on *recall* and *precision* measures as used in testing the performance of image retrieval in Chapter 4.

From the collection of trajectories that we created we choose 10 trajectories as queries. We posed these queries to the system. Each query was repeated for a number of different accuracy levels (from 50 to 90). We then calculated the (geometric) mean of recall and precision of retrieval for the five queries in each experiment for each accuracy level. The same queries are asked to 7 users (postgraduate students and academics).

Table 6.1 gives the results from our experiments.

Query	Accuracy/threshold value									
	50		60		70		80		90	
	R	P	R	P	R	P	R	P	R	P
T_4	100	47	89	57	67	67	56	83	44	100
T_{18}	100	39	86	43	71	63	57	67	57	100
T_{20}	100	32	71	42	43	60	14	100	14	100
T_{21}	100	42	100	67	70	100	20	100	20	100
T_{26}	88	36	67	80	67	100	67	100	33	100
T_{29}	100	13	100	40	100	100	100	100	100	100
T_{32}	67	44	67	67	50	75	17	100	17	100
T_{49}	67	22	67	67	67	67	33	100	33	100
T_{59}	89	53	67	75	67	87	44	100	44	100
T_{82}	100	43	90	50	50	56	20	67	20	100
Mean	91	24	79	57	64	76	35	91	32	100

Table 6.1: Recall/Precision Table on Trajectories Similarity Retrieval(P: precision, R: Recall)

We choose the accuracy of 50, 60, 80 and 90 to show the filtering effect of the similarity trajectory algorithm that we developed in the previous chapter. We do not use the the threshold value below 50 since according to our measure if two trajectories are less than 50% similar then the two trajectory tends to have opposite directions. For example, if the first trajectory is fully in the right direction (direction 1) and the second trajectory in the top-left direction (direction 4) then they are 25 % similar.

Table 6.1 above shows that the trajectory similarity algorithm does a good filtering job in retrieving similar trajectories. For a threshold value of 50, the recall is very high but the precision is very low. On the threshold value 80 and beyond the precision value is very high and recall value is reduced significantly. On the threshold value 70, both the recall value and precision value are quite high. Therefore we choose the threshold value 70 as the default threshold value for trajectory similarity retrieval in the prototype.

6.6 Comparison With Other Models

Spatiotemporal information, especially information that deals with moving objects is a new topic in the database area. Our work is one of the pioneers in the area of spatiotemporal information and moving objects. To complete our discussion in this topic, we compare our model with some related models including Lee [69], Dimitrova [26], Bimbo [10], Shearer [100], and Sistla [101] models.

In Lee's model, the basic unit of data is video records, which are comparable to our basic unit of data scene. Retrieval is based on video records. In Lee's model, a query can be posed based on video record identifiers, objects in a video record, tracks or some combination of these. Our model does not use video identifiers for querying; however knowing the scene identifier (a name associated with a scene) is beneficial for constructing a new scene.

Lee's definition of a track is different from ours. Our tracks are true spatiotemporal data that contain sequences of distance-direction pair with temporal intervals associated with them. Our definition of trajectory, which is the spatial projection of track, is comparable to Lee's track. Lee's track is richer than our trajectory, since it considers four classes of motion taking account objects translation, rotation as well as camera motions. Our trajectory is very simple since it considers only object translation. As a consequence a query based on track in Lee's model is difficult to specify. In contrast queries on trajectories in our model is very simple, a user need only draw a trajectory (which is free line drawing) in a window and send it to the system as a query. In addition, we define similarity measure for retrieval on trajectories, which is lacking in Lee's model.

As we mentioned earlier our track is real spatiotemporal data, hence it is possible to answer a query "how long is an object moving in a certain direction" . Lee's track cannot be used to answer such a query since there is no time associated with a track.

Our model addresses spatiotemporal relationships among objects in a scene.

Therefore, queries regarding behaviour (how objects relate) objects during motion are supported. In addition it is possible to construct a new scene from existing scenes using the *projection* and *overlay* operations in our model. Lee's model does not support these features.

Dimitrova's model on moving objects combines object attributes such as name, type and shape of objects with trajectories and moving description (user supplied). Even though our model currently does not include object attributes, the model can be extended to include these attributes easily. A query language, called EVA, and its visual counterpart, VEVA, are provided for video clip retrieval. This model is only comparable to ours in the ability to query based on an object trajectory. A trajectory in Dimitrova's model is represented by a chain code. This representation is quite similar to our trajectory representation, except in our representation we take account the length of each direction in trajectory. Dimitrova's model does not consider time interval associated with each segment in the chain code.

In posing a query using trajectory, a user can choose an exact match or a similarity match, which is quite similar to ours. However, we define the similarity measure on trajectory matching while this measure is not defined in Dimitrova's model.

There is no comparable feature in Dimitrova's model with our spatiotemporal relationships. Thus one cannot ask queries regarding the relations among objects in a clip. Also, Dimitrova's model cannot create new clips from existing clips by means of projection of some objects in a clip or integration of some objects of a clip with other objects of another clip.

Bimbo's provides an interesting model in representing moving objects using spatiotemporal logic. The logic is an extension of (point-based) temporal logic, with the operator *until* as the main operator. Bimbo's model ignores tracks or trajectories, while tracks or trajectories are our main features. As a consequence Bimbo's spatiotemporal logic cannot answer queries on trajectory.

Bimbo's spatiotemporal logic is quite comparable to our spatiotemporal relationship model. The big difference is that Bimbo uses a logic approach and we use an algebraic approach. In our model, a query relates to relationships among objects in a scene can be answered. This can be done in Bimbo's model too. However, since Bimbo's model for temporal relationships is point-based, a query asking duration (for example "how long object A is inside region B"), cannot be answered. In modelling spatial data, Bimbo's model is point-based, that is it represents a spatial object as points while ours is region-based.

In Bimbo's model, a scene is represented by spatiotemporal logic assertions, while our model represents scene as a symbolic graph. In retrieving similar scenes Bimbo compares the spatiotemporal logic representation of a query scene with the spatiotemporal logic representation of scenes in a database. The scene matching is exact matching. Our model provides both exact and similarity matching. The spatiotemporal logic representation of a scene is very complex, and it is virtually impossible to use the spatiotemporal logic directly in specifying scene queries. Therefore, the visual query editor is provided which will translate the visual query to a corresponding spatiotemporal logic query. It is not clear how a query like "find a scene where two objects A and B enter region C in which A enters C before B enters C and finally A and B collides inside C" can be supported in Bimbo's model; while such complex queries are easily specified using either a graphical language or a non-graphical language in our model.

There is no counterpart operation or expression which is comparable to our *projection* and *overlay* operations. Therefore it is not possible to create a new scene from existing scenes.

Shearer [100] proposed a model that extends 2D B-strings for moving objects. 2D B-strings can represent both directional and topological relationships. However, topological relationships are not robust, they can only model topological relationships on rectangular objects (bounding rectangle of objects). This model can only be used for scene matching. It supports limited similarity matching (only three types of matching); while our model supports flexible similarity matching for scenes. Since Shearer's model does not support spatiotemporal relationships, a query regarding spatiotemporal relationships cannot be posed. For example, Shearer's model cannot answer the query "find a clip where object A enters region B" or the query "how long object A inside region B in a clip" or the query "find a clip where object A enters region B during object C passes region B".

Shearer's model ignores trajectories of moving objects, and thus cannot answer queries on trajectory. Similar to other models, this model does not support operation comparable to our operations *projection* and *overlay*.

A new model for moving objects called MOST has been proposed by Sistla [101]. The most interesting aspect of this model is the ability to ask about future states. In this model each dynamic object such as moving object has a temporal function associated with it. This function is the heart of the model, so that future states can be asked about. We do not have a comparable feature. Our closest feature with temporal function is tracks. But a track is unable to answer queries on future states, unless an interpolation function is developed. However, we consider that the temporal functions (motion function in this case for moving object) have a severe limitation: it is very difficult to determine the motion functions in advance. In fact to set up a temporal function, a track is needed; however, there is no concept of track in MOST.

MOST has a comparable concept to our spatiotemporal relationships called database histories. However, the representation of database history is not defined, hence we are not sure whether MOST can answer queries as we pointed out above (when we compare with other models). There is no definition of scene retrieval including similarity retrieval. A query in MOST is like other traditional database query (MOST uses SQL-like syntax) enriched with spatial operations and temporal logic operators such as *until* operator.

MOST does not have a comparable operation like our *projection* and *overlay*. So MOST also does not have an ability to reuse individual objects in a scene for creating a new scene.

Having compared our model with some other related models, we claim that our model has some unique features including integrating tracks/trajectory with spatiotemporal relationships and the ability to create a new scene based on existing scenes using projection and overlay operations, and supporting exact as well as similarity retrieval equipped with graphical language (note that our graphical language is never translated to a query language, it is direct representation of a scene graph (including trajectory) as well as traditional query language. Table 6.6 summaries the above comparison.

6.7 Summary

A prototype of a moving-object retrieval system is presented. The architecture in the prototype has been implemented using the Java language. We extend the *bi-array* representation of 2D-PIR graph to capture moving objects. Queries can be specified by a graphical facility or by a query language.

We conducted several experiments on trajectory similarity retrieval. The result of our experiments on trajectory similarity retrieval indicates that our metric for trajectory similarity is effective. So far no known experiments on trajectory similarity retrieval have been published, to the best of our knowledge, our experiment is the first to address similarity retrieval on trajectories.

We compare our model with other models. Our model provides some important features which are lacking in other models.

Features	Lee	Dimitrova	Bimbo	Shearer	Sistla	Ours
Data model	**Video record** (a record contains clip, text annotation and track record)	**Object-oriented** (A video is a composite object of triplet: object representation, motion representation and video ID)	Spatiotemporal logic	2D B-string	Moving object spatiotemporal (MOST)	Graph-based called symbolic scene
Motion representation	A notion of track is defined. A track is a combination of translation and rotation motions of a object. No time associated with a track.	Each moving object has a trajectory and a corresponding motion description. No time associated with a trajectory.	No	No	No	Each moving object has a track which is represented as triplet: distance travelled, direction and time interval.
Spatiotemporal relationships	No	No	Yes, in the form of spatiotemporal logic	No	Yes	Yes, spatiotemporal relationships are defined both for individual and multiple moving objects.

Table 6.2: Comparison of our model on moving objects with related models

Features	Lee	Dimitrova	Bimbo	Shearer	Sistla	Ours
Query on trajectory	Query by a track is specified manually. No definition of similarity on tracks.	Query based on trajectory is supported. Similarity matching on trajectory is allowed but the similarity measure is not defined.	No	No	No	Yes
Query based on spatiotemporal relationships	No	No	Yes, expressed by spatiotemporal logic.	No	Yes	Yes
Scene similarity retrieval	No	No	No	Yes, partial similarity matching	No	Yes
New scenes creation	No	No	No	No	No	Yes

Table 6.2: Cont.

Chapter 7

Examples of Applications

7.1 Introduction

In this Chapter we will demonstrate the power of query facilities (both the graphical query facility as well as the query language) using several applications. When using the query language, the syntax defined in Chapter 5 is used.

7.2 Animal Movement Querying System

The Animal Movement Querying System tracks the movement of animals using Global Positioning System (GPS). A GPS receiver device is attached to a sample animal to track its movement in a region.

The system can be used to study the movement of cattle in a large farming area or to study the migration pattern of a certain animal group (such as elephants, migratory birds, etc.).

The system has a map depicting regions that animals can move into. The GPS will sends time series data about the position of the sample animal. For this purpose, the position is sent hourly. When the system receive the data, it stores the data and creates a *track* and store spatiotemporal relationships among the animal and regions on the map.

Some possible queries posed to the system are:

1. Retrieve animal migration patterns that are similar to this trajectory.
2. Give me the trajectory of a named animal migration.
3. How long is a named animal inside a region?
4. Find animals that enter a certain region from the west side and get out of the region from the north side.
5. Find animals that stay in any region longer than a specified time (an indication that the animals may breed in this region).
6. Retrieve scenes that contains animals pass through a certain region.

Before we present how the above queries are specified, we would like to make some notes about the system and how this system can be implemented. The identity of animal is known, since we attached a GPS receiver to it. In general the model is region-based. GPS sends the position of animal as a point. The model can handle point objects (and also line objects), in fact the computation of spatiotemporal relationships between the animal (point) with every region becomes easier and faster. Since regions in the map are static, the computation of spatial relationships among regions is done only once. Note that the map is arranged is such a way such that the x-interval relationship "$<$" and "$>$" corresponds to west and east and y-interval relationship "$<$" and "$>$" corresponds to north and south respectively.

Assume that the animal migration is recorded in a database called MigrationDatabase. We illustrate how the above queries are specified using sketch-based query facility in this prototype.

Q1. Retrieve animal migration pattern that is similar to this trajectory.

This query can be answered by drawing a trajectory-sketched in a window. To enhance the display, the map can be displayed as a background and the user can draw a trajectory on this map. This sketch is then posed to the system for similarity retrieval (the user may adjust the threshold value; based on our experiment, the default threshold is 70). The system will find similar trajectories and return them to user. Using a GUI a user may request the system to show these trajectories in a window with the map as a background. An animal icon will be shown on the window, and the user may ask the system to provide animation of the migration of the animal.

Note that it is impractical to asks this query using traditional query language.

Q2. Give me the trajectory of a named animal migration.

The answer of this query is a trajectory which will be displayed in a window. If we know the scene name of the migration animal (say ElephantMigration), the specification of this query is as follows:

$ElephantMigration\ ExtractS(Elephant)$

However, if we do not know the name of the scene, we have to find the scene first and then extract the trajectory of Elephant. Here is the the query.

$(MigrationDatabase : M\ where(M\ has\ Elephant))\ ExtractS(Elephant)$

For queries Q3 to Q6, let us identify the animal as A and the region as R.

Q3. How long is a named animal stays a region?

First we create a name AInsideR which holds a scene which contains animal A inside region R. Next, we compute how long animal A is inside R. Here is the query specification.

$AInsideR\ is$

$MigrationDatabase : M\ where\ (M\ extractT(A, R, [(ic, d, d)])\ dur\ >\ 0)$

$AInsideR\ extractT(A, R, [(ic, d, d)])\ dur$

The expression $M\ extractT(A, R, [(ic, d, d)])\ dur\ >\ 0$ checks whether the spatial relationship (ic,d,d), which specifies A is inside R, exists. If yes, then the scene held by M is bound to name AInsideR. Next we extract the time interval of relationship (ic,d,d) between A and R in AInsideR using the expression $AInsideR\ extractT(A, R, [(ic, d, d)])$. Finally, we compute the duration of this interval, which is the answer.

Q4. Find a scene consists of animals that enter a certain region from the west side and leave the region from the north side.

The query specification is straightforward as follows:

$$MigrationDatabase : M$$
$$where\ (M\ extractS(A, R)\ match\ ([(dt, <, d)..(ic, d, d)..(dt, d, >)])$$

First we extract the spatial component of spatial relationships between A and R, which is a sequence of 2D-PIR. Next, the match message (corresponding to the match operation) with argument $[(dt, <, d)..(ic, d, d)..(dt, d, >)]$ which specifies west–inside–north is sent. If the match is affirmative, the corresponding scene (M) is retrieved.

Q5. Find animals that stay in any region longer than a specified time.

Let us use t for representing the specified time. The query specification is as follows:

$$MigrationDatabse : M\ where\ (M\ extractT(A, R, [(ic, d, d)])\ dur > t)$$

The expression $S\ extractT(A, R, [(ic, d, d)])$ extracts time interval of the corresponding spatial relationship (ic,d,d) between the animal A and any region R in the scene (which means A inside R). Let us call this interval I. Next the *dur* operation is sent to this interval which will return the duration of the interval. Finally this duration is compared to user supplied time t. if all the conditions are satisfied the corresponding scene(s) will be retrieved.

Q6. Retrieve scenes that contain animals pass through a certain region.

This query is similar to Q4, except the direction when A enters R and the direction A exits R are ignored.

$MigrationDatabase : M$

$where\ (M\ extractS(A, R)\ match\ ([(dt, ?, ?)..(ic, d, d)..(dt, ?, ?)])$

7.3 Traffic Monitoring System for Intersection

The system performs monitoring for an intersection not controlled by a traffic light, to monitor drivers behaviour on "give way" rules[1] in an intersection. A dedicated camera is installed in the intersection.

The following are some "give way" rules for intersection. If the driver of car A is turning at an intersection, he/she must give way to car B (see Figure 7.1).

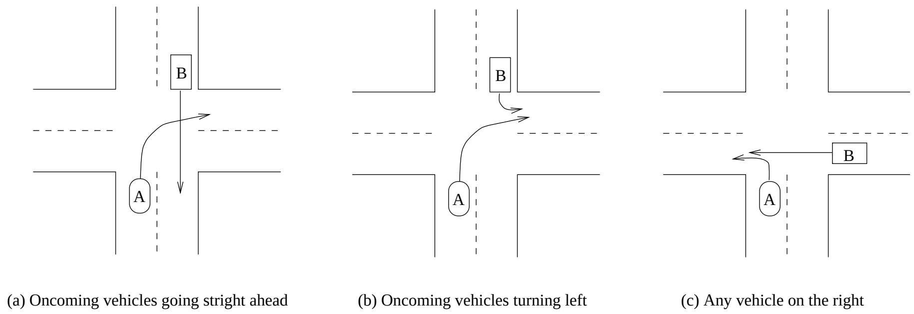


Figure 7.1: Give away rules

For retrieval purposes, an intersection is divided into several regions as depicted in Figure 7.2.

Vehicles coming from R1, R2, R3 and R3 are identified by V1, V2, V3 an V4 respectively.

The camera is activated when a vehicle enter the areas above. Here are some queries asking for scenes that contains drives which do not obey the "give way" rules.

[1]The rules are taken from *Road Users' Handbook, RTA New South Wales Australia, 1993*

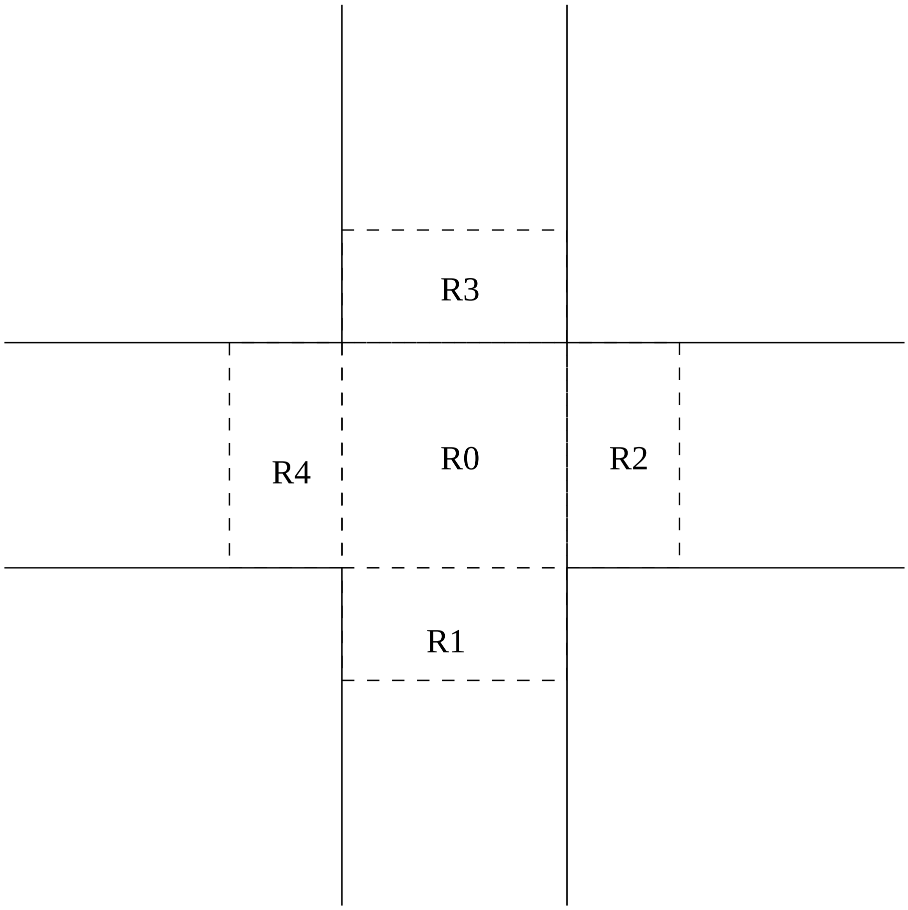


Figure 7.2: Intersection areas

1. Find scenes that possibly contains vehicles disobeying the give way rules given in Figure 7.1 (a) and (b).

2. Find scenes that contain vehicles from R3 and R4 crashed in R0.

3. Find scenes that contain vehicles disobeying the give way rule depicted in Figure 7.1 (c) such that the vehicle from R2 hits the vehicle from R1.

Let as assume that SDB is a database contains a collection of scenes recorded by the system. Q1 can be answered using the following MOQL expression:

$SDB: \ S \ where \ ([(ic, d, d)..(dt, >, >)] \ match(S \ extractS(V1, R1)) \ and$

$[(dt, d, <)..(ic, d, d)..(dt, >, d)] \ match(S \ extractS(V1, R0)) \ and$

$[(ic, d, d)] \ match \ (S \ extractS(V3, R3)))$

The expression $[(ic, d, d)..(dt, >, >)]$ *match* $(S\ extractS(V1, R1))$ checks the existence the sequence 2D-PIR between V1 and R1 which initially inside R1 $((ic, d, d))$ and finally outside R1 when V1 enters R2 $((dt, <, <))$ in TS-2D-PIR between V1 and R1. The expression $[(dt, d, <)..(ic, d, d)..(dt, >, >)]$ *match* $(S\ extractS(V1, R0))$ checks whether V1 pass R0 which starting from R1 and ending in R2 and the expression $[(ic, d, d)]$ *match* $(S\ extractS(V3, R3))$ checks whether V3 is inside R3. If these three expressions are satisfied then S will be retrieved.

The above MOQL expression is quite complex. The user needs to understand very well how V1 relates to R0, R1 and R2 very well during motion. The query above can be expressed with a simpler graphical query as shown in Figure 7.3

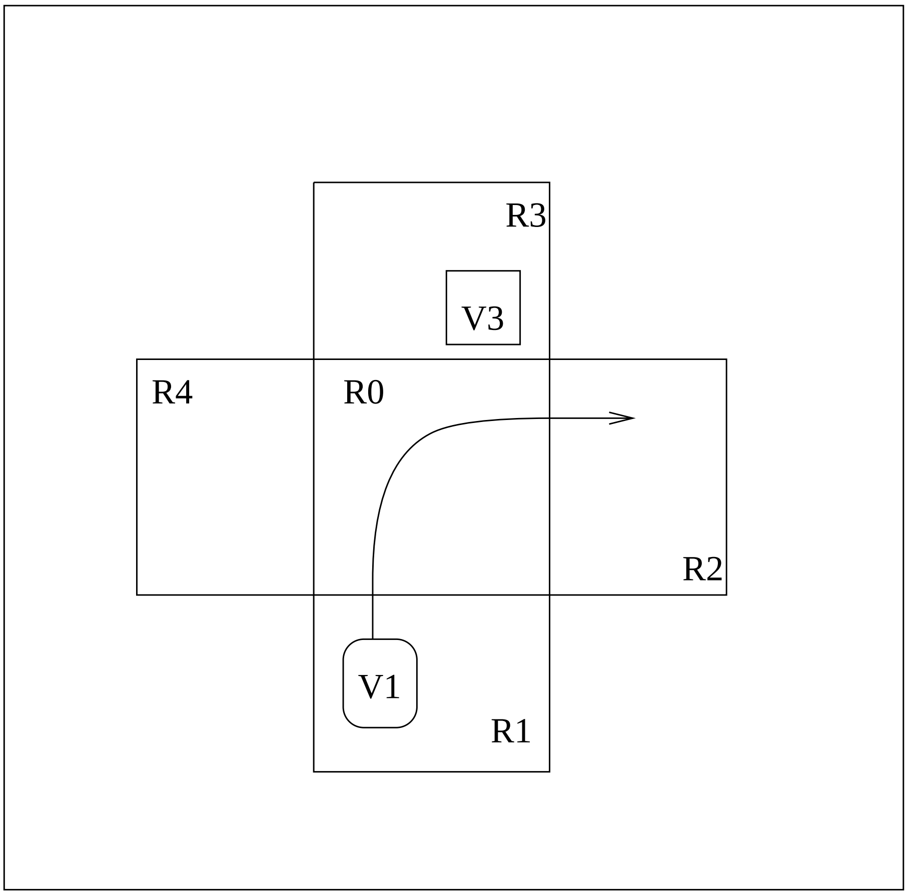


Figure 7.3: Graphical query of Q1

In Figure 7.3, the trajectory of V1 is specified. This will create an animation of V1 disobeying give way rule. This animation will be transformed into a scene graph. The next step is to retrieve a very similar scenes from SDB.

The following MOQL expression specifies Q2.

$SDB: \ S\ where([(dt,?,?)..(ic,?,?)]\ match\ (S\ extractS(V3,V4))\ and$

$[(dt,?,?)..(ic,d,d)]match\ (S\ extractS(V3,R0))\ and$

$[(dt,?,?)..(ic,d,d)]match\ (S\ extractS(V4,R0))$

The expression $([(dt,?,?)..(ic,?,?)]\ match\ (S\ extractS(V3,V4))$ specifies the existence of V3 crash with V4 (V3 initially disjoint with V4 and then "overlaps" with V4. Assume that V3 still "overlaps" V4 after crashing. The expressions $[(dt,?,?)..(ic,d,d)]\ match\ (S\ extractS(V3,R0))$ and $[(dt,?,?)..(ic,d,d)]match$ $S\ extractS(V4,R0)$ restrict that V3 cashes with V4 inside R0.

The following MOQL expression specifies Q3.

$SDB: \ S\ where([(dt,>,?),(to,?,?),(dt,>,?)]\ match\ (S\ extractS(V1,V2)))$

This expression assumes that after V1 hits V2, V1 disjoint with V2 (that is initially disjoint, then touch and finally disjoint).

7.4 A Dedicated Video Surveillance System

A Dedicated Video Surveillance System performs monitoring of an object or some objects from imposture (theft, abuse, deception, etc.). The object(s) which is to be protected is laid in a place surrounded by a security zone. A security camera watches the region. The camera will be activated if an object (in this case a person) enters the security zone. When the camera is active, a scene is created and stored in a database (called SecurityDatabase).

One of the indicators of imposture is that the suspected person stays inside the security zone longer than necessary (he is doing something on the protected object).

When the protected object is stolen or abused, the following query will retrieve possible scene that contains a suspected person.

$SecurityDatabase: S$

$$where((S\ extractT(SecurityZone, Person, [(ic, di, di)]))\ dur > t)$$

The where clause corresponds to the *selection* operation which scans the SecurityDatabase and applied its predicate on every scene (clip).

The expression $S\ extractT(SecurityZone, Person, [(ic, di, di)])$ extracts time interval of the corresponding spatial relationship (ic,di,di) between the security zone and any person in the scene (which means Person inside the security zone). Let us call this interval I. Next the *dur* operation is sent to this interval which will return the duration of the interval. Finally this duration is compared to the user supplied time t. If the duration is greater than t then the person is suspected to be an impostor. Note that t is supplied based on security officer judgement concerning the maximum time allowed for a person to be inside the security zone.

This is a very simple application of the moving-object database. In this application a fully automatic object identification is highly possible since the domains of involving objects are very limited, the security zone and person.

7.5 Video Retrieval System

Suppose we have a collection of clips stored in a database called SceneDB. The following are possible queries posed to the system that implement the moving object data model.

Q1. Find scenes in SceneDB that contain an object whose movement is similar to the movement of object X in this scene (Scene5).

This query can be answered by comparing the similarity of the trajectory of object X in Scene5 with the trajectory of any object from other scenes stored in SceneDB. The following expression specifies the query:

$$SceneDB : S\ where(S\ similarTo(Scene5\ project(X)))$$

The expression $Scene5\ project(X)$ projects the object X from Scene5. We need this projection since the query is explicitly asking for the object X's movement. We then compare the similarity between object X (in this case its trajectory) of any scene in SceneDB. Note that the similarity function will compare the similarity between object X to any object in each scene S. If one of the of object in scene S is considered similar to X then the scene S will be retrieved.

Q2. Find scenes in SceneDB that contain a walking person whose trajectory is similar to the trajectory drawn in this window.

A trajectory can be represented as a sequence of distance-direction pairs. So a trajectory drawn in a window is represented as a sequence $[d, r]$. To answer the query above, we need to compare this trajectory with the spatial projection of the person object (assume P is a symbol that identifies the intended person) contained in a scene. Let T represent the trajectory, then the query can be specified as follows:

$$SceneDB : S\ where((S\ project(Person)\ similarTo(T)))$$

The expression $S\ project(Person)$ projects a person(s) from scene S. The expression $S\ project(Person) similarTo(T)$ will compare the trajectory T with the trajectory of every person in S. The scene S will be retrieved if it contains a person whose trajectory is similar to T.

This query is much easier if we specify it using graphical query

Q3. Find a scene in SceneDB containing a person entering a region whose movement is sketched in a window as shown in Figure 7.4.

The sketch is transformed into a scene graph. Scene similarity matching is then applied to retrieve scenes in the database which are similar to the sketch.

It is worth noting that this kind of query will be common for retrieving moving objects especially those involving trajectories. For example, It is very hard to express a query that involves many objects using a traditional textual query language.

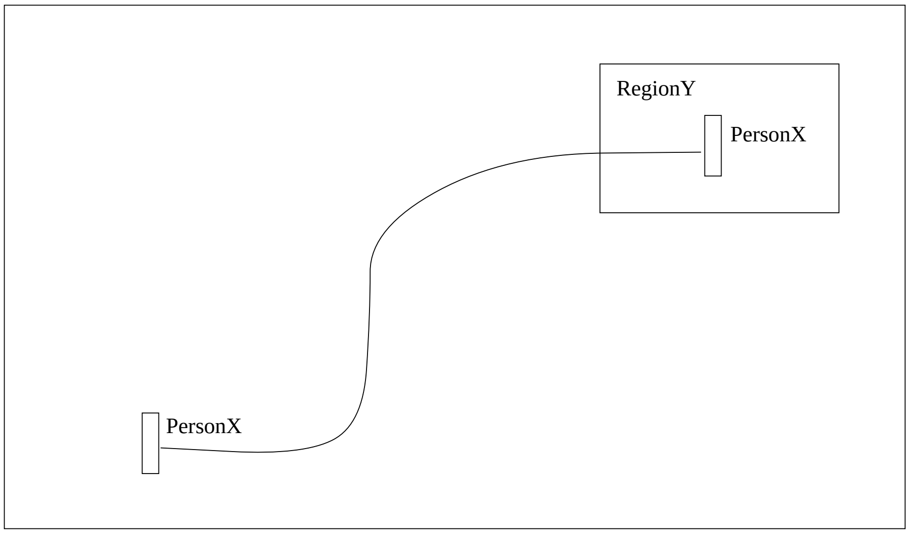


Figure 7.4: Sketch of query

Consider the query: "find an object X that passes between several objects (still objects)" as depicted in Figure 7.5.

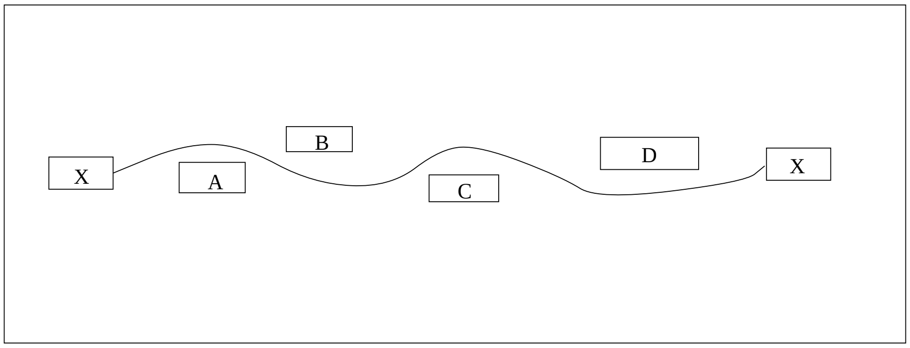


Figure 7.5: X pass through A, B, C and D

Imagine also if object X is a racing car passing through others (A, B, C, and D) where A, B, C, and D have their own movements. This is extremely hard to express in natural language or any traditional query language. However, a sketch or an existing scene has a corresponding scene graph. Retrieving based on scene graph (scene graph matching) is not only possible but also relatively simple.

Q4. Find scenes in SceneDB that are similar to a scene requested by a user.

Suppose the scene that shown in a window is identified by the symbol *myscene*. This query requires scene similarity matching, and can be expressed as follows:

$SceneDB : S\ where(SsimilarTo(myscene)$

Q5. Find a scene in SceneDB that contains a person entering a region from the right.

Let us assume that the person is identified by the symbol Person and the region is identified by the symbol RegionX. The following expression answers this query:

$$SceneDB : S\ where([(dt, >, ?)..(ic, di, di)]\ match(S\ extractS(Person, RegionX)))$$

The expression $S\ extractS(Person, RegionX)$ extracts the spatial component of the spatiotemporal relationship between Person and RegionX, which is a sequence of 2D-PIR. The expression $[(dt, >, ?)..(ic, di, di)]$ specifies the sequence of 2D-PIR relationships implied in the expression "enter from right". The basic idea is that initially the relationship is disjoint and right $((dt, >, ?))$ and the final relationship is inside $((ic, di, di))$. So, the expression $[(dt, >, ?)..(ic, di, di)]\ match(S\ extractS(Person, RegionX))$ checks whether $[(dt, >, ?)..(ic, di, di)]$ matches with $S\ extractS(Person, RegionX)$. If it matches, then the scene S will be retrieved.

Q6. Find a scene in which object X enters region R while object Y departs from region R.

This query is different from the previous queries since we need to consider the temporal relation between object X entering region R and object Y departing from region R. The time interval where object X is inside region R must overlap with the time interval where object Y is also inside region R. This can be expressed as follows:

$$SceneDB : S\ where(S\ extractT(X, R, [(ic, di, di)])$$
$$o(S\ extractT(Y, R, [(ic, di, di)]))$$

Let us consider the above expression carefully. The expression $S\ extractT(X, R,$

$[(ic, di, di)]$ extracts the interval of the spatial relationship (ic, di, di) of TS-2D-PIR between X and R in Scene S. Let us call it I_1. Similarly, $S\ extractT(Y, R, [(ic, di, di)]$ extracts the interval of the same spatial relationship of TS-2D-PIR between Y and R. Let us call it I_2. We then check whether I_2 overlaps I_1 with Allen's operator o (i.e $I_1.o(I_2)$).

Q7. Find a scene in which object X enters region R from the left while object Y departs from region R toward the right.

This query is can be expressed precisely by nested expression. The idea is to find scenes where object X enters region R from the left and we find the intended scene from this.

$SceneDB : S\ where([(dt, <, ?)..(ic, di, di)]$
$\quad match(S\ extractS(X, R))) : S1\ where(S1extractT(X, R, (ic, di, di))$
$\quad o(S1\ extractT(Y, R, [(ic, di, di)])))$

We can use the naming facility to make the above query more readable as follows:
$XenterRfromLeft\ is$
$\quad SceneDB :\ S\ where([(dt, <, ?)..(ic, di, di)]\ match(S\ extractS(X, R)))$
$XenterRfromLeft :\ S\ where(SextractT(X, R, (ic, di, di))$
$\quad o(S\ extractT(Y, R, [(ic, di, di)])))$

7.6 Animation Database

The above examples show how to retrieve a scene(s) from a scene database using available operations in our model. The following examples show how we can create a new scene by projecting or uniting existing scenes.

Let us imagine we have a database of scenes (clips) of animations. We want to create a new clip that tell a story "a cat is catching a mouse while a dog is running towards a pond and drinking water". In the database we have a clip that contains some animals and two cats. One of the cats is catching a mouse. In the database we have also a clip that contains several dogs are running towards a pond and then drinking. Now, how can we construct a new clip that specifies "a cat is catching a mouse while a dog is running towards a pond and drinking water"? The following show how this is possible using the power of *object projection* and *overlay* operations.

In the first clip we are interested only on a cat (called tom) which is catching a mouse (called jerry). Suppose the clip is called CatAndMouse. The following is the expression that satisfy restriction.

$CatAndMouse\ project(tom, jerry)$

This expression creates a new clip that contains "a cat is catching a mouse".

In the second clip we are interested only on a particular dog (called bulldog) which is running towards a pond (called aPond) and then drinking. Let us assume the clip is called DogDrinking. Here is the expression that creates a new clip that containing "bulldog is running towards a pond and then drinking".

$DogDrinking\ project(bulldog, aPond)$

Now we are ready to create a new clip "a cat is catching a mouse while a dog is running toward a pond and drinking water". Here is the expression (using the overlay operation).

$CatAndMouse\ project(tom, jerry)$
$\quad overlay(DogDrinking\ project(bulldog, aPond))$

Now, suppose we do not want the whole scene of "bulldog is running toward a pond and then drinking". We just want a scene "a drinking dog". We know that the length of the second clip is approximately 60 seconds. And we are sure that the last 30 seconds of this clip show the bulldog is drinking. Here is the expression that creates a new clip of "drinking dog".

$DogDrinking\ project(bulldog, aPond, 30, end)$

This will create a new clip from the the last 30 seconds of the clip "DogDrinking".

7.7 Summary

We have demonstrated the expressive power of our model through several applications. We illustrated that a query can be specified using MOQL as well as a graphical language. Some queries are easily expressed using MOQL, especially simple queries and queries involving temporal relationships. Other queries are easily expressed using a graphical language (animation), especially for similarity-based queries and query involving many objects with very complex relationships. Note that in our model graphical queries do not need to be translated to MOQL since a graphical query will directly be transformed into a scene graph. A graph comparison will be carried out for retrieving similar scenes.

Chapter 8

Conclusion

8.1 Concluding Remarks

A novel graph structure has been used successfully to represent both static and dynamic multimedia data. This graph structure, with nodes being objects and edges being relationships (spatial relationships for static data and spatiotemporal relationships for dynamic data), abstracts away many details from an actual image/picture or image sequence (moving picture) and at the same time it retains a detailed summary of spatial or spatiotemporal relationships among objects. This graph structure provides a representation for static and dynamic multimedia data.

Each individual object (spatial object) is modelled as a pointset. Using the concept of boundary and interior of pointsets, a complete and mutually exclusive topological relationships can be derived. Using the concept of projection of pointsets on the x-axis and y-axis a set of interval relationships are derived. By combining topological relationships with these interval relationships the 2D Projection Interval Relationship (2D-PIR) is constructed. So, 2D-PIR is not only built on top of the sound mathematical pointset concept, but it is also a powerful spatial relationship concept (which can represents 169 spatial relationships robustly).

The 2D-PIR graph where nodes are symbols representing objects and edges are 2D-PIR relationships is used to construct a symbolic picture from a real image. We develop a simple algorithm for 2D-PIR graph construction and an efficient structure for storing the graph called *bi-array*. The problem of matching between images is reduced to matching between their corresponding 2D-PIR graph representations.

In dealing with similarity matching between images, we develop a similarity measure based on the concept of topological and interval distances in 2D-PIR. The similarity measure is experimentally demonstrated to be an effective measure of similarity retrieval of images. We also show that our symbolic picture representation is reflection invariant (both horizontal and vertical) and rotation invariant to a certain degree.

We develop a simple architecture for an image retrieval system. The heart of the system is the *symbolic image transformer* that semi-automatically transforms a "real world" image into a symbolic picture (2D-PIR graph). 2D-PIR graphs support graphical queries directly, therefore sketch-based and iconic-based queries are supported naturally (without being translated to a lower level language). The architecture is network-based, hence distributed picture storage is one of its features.

When we compare 2D-PIR similarity retrieval with Gudivada's similarity retrieval, 2D-PIR similarity retrieval performs better than Gudivada's on all collections that we used. This is because 2D-PIR representation provides more information than Gudivada's (which ignores topological relationships).

We also compare our model for static multimedia data to several other related models that use spatial relationships. Our model is more powerful since it provides finer spatial relationships with well defined similarity measures. We also provide a formalism for 2D-PIR to support reflection and rotation invariance which is not addressed in other systems.

Our model of static multimedia data can be extended directly to model dynamic multimedia data (moving objects). This is the outstanding feature of our model which is not addressed in other models.

A moving object is modelled as a pair *(symbol,track)*. The symbol uniquely identifies the object and the *track* describes how the object moves. We then consider modelling several moving objects in a scene. A scene is a graph where the nodes represent the objects, the loops in the graph are labelled by tracks, and other edges are labelled by spatiotemporal relationships. A track can be seen as a sequence of spatiotemporal relationships between an object with itself at different locations and times. The scene graph is an extension of a 2D-PIR graph to incorporate moving objects.

The moving object model is equipped with several operations on moving objects and scenes which are useful, not only for retrieving moving objects in a scene database, but also for constructing new scenes from existing ones.

A scene graph is constructed from a sequence of pictures using a simple, efficient method. The construction leads to a graph structure for which an efficient scene-matching algorithm can be developed. Scene retrieval can be based on exact scene matching or similarity scene matching. We define similarity measures on trajectories and on sequences of 2D-PIR. The similarity measure on 2D-PIR sequences is developed based on the successful 2D-PIR picture similarity measure. For scene similarity, we integrate trajectory similarity and sequence similarity (corresponding to loops and edges in a scene graph).

We develop a moving object retrieval system. The system supports sketch-based queries (graphical queries) and a query language. We consider both visual features and traditional query languages as necessary, since some queries are difficult to specify in a traditional query language, but are rather easily specified in a graphical form. Yet, some other queries are more easily specified in a traditional query language than in a graphical form.

In the examples, we illustrate how to retrieve moving object(s) stored in a database and create new scenes from existing ones using the available operations. We also show that a scene graph can be used for specifying a query: that is, sketch a query and this sketch is transformed to a scene graph. Thus, scene graphs not only provide a representation for moving objects but can also be used to retrieve moving objects (that is, they support a direct graphical query facility).

We compare our model with several related models. So far our model provides more features over other models. Our model supports both tracks/trajectories and spatiotemporal relationships. In addition new scenes can be generated from existing scenes in our model. This feature is unique in our model.

In summary, the major contribution this thesis is a unified model for static and dynamic multimedia data. Based on this model we develop formal similarity measures on 2D-PIR, trajectories, spatiotemporal relationships and scenes so that similarity retrieval on static and dynamic multimedia data can be conducted according to user needs. A set of powerful operations on moving objects are derived from the model. Based on these operations we build a query language for moving objects. The model also supports graphical query on static and dynamic multimedia data.

8.2 Future Work

2D-PIR is a powerful representation formalism for spatial relationships. Possible future work is to incorporate 2D-PIR with Geographical Information Systems (GIS). Queries that involve spatial relationships can be expressed using examples by drawing spatial objects in a window. The system could then compute 2D-PIR and pass the result to a query interpreter. Using 2D-PIR a user can specify more precise spatial relationships than in traditional spatial relationships. In addition, a user friendly interface for expressing spatial relationships between objects is supported by 2D-PIR.

2D-PIR combines the work of Allen[3] (temporal relationships) and Egenhofer [27] (topological relationships). Reasoning on temporal (interval) relationships was developed by Allen [3]. Temporal reasoning based on semi-intervals has also been developed [35]. Reasoning on topological relationships has been developed too [28]. Another avenue for future research is to investigate a reasoning mechanism based on 2D-PIR. For example, given 2D-PIR relationships between A and B and between B and C, can we derive a 2D-PIR relationship between A and C?

Currently, there is much research on content-based image retrieval of static multimedia data using visual properties of objects in an image. Extending our model to incorporate visual properties of objects is another interesting avenue for further research. Since boundaries of objects (boundaries of pointset) are computed, we can extract the shape of objects. From boundaries we extract interiors, hence we can extract colour and texture of objects as well. These visual features can be added as properties of nodes both in a 2D-PIR graph and a scene graph. The graph becomes "bigger"; however, more queries can be posed on node properties, relationships among nodes or the whole graph.

Indexing of pictures based on 2D-PIR has been explored [12] using the R+ tree. An efficient approach to index similar images using spatial relationship was presented in [119]. Indexing of pictures using visual properties has been addressed by many researchers such as [31, 62]. One of the future avenue is to investigate a unified index method that combine visual and 2D-PIR properties.

In this thesis we propose an algebra for moving objects. The algebra has several powerful operations for moving objects and scenes. We have not investigated whether this algebra is complete or not. One of the future avenue for future research is to investigate the completeness of the algebra.

We have defined operations and built a query language for moving objects. Query optimisation issues in moving object have not been addressed, therefore an efficient query processing is another avenue for future research.

Another interesting but challenging aspect for future work is indexing on scenes containing moving objects. In an image, the relationships among objects are static. In a scene of moving objects, the relationships among objects may change over time (spatiotemporal relationship). In addition moving objects have tracks which should be considered as well. Spatiotemporal relationships and tracks are far more complex than spatial relationships.

Sistla [101] introduced time-based functions to predict future states. Basically, this function is a track. In our model a track is represented by a sequence of points and time associated with them. The time-based function can be derived from a track, hence, in fact, we have a basis for extending our model to support queries about the future behaviour of objects in scenes. If time-based function is available for each object, than we can pose queries on future states (that is, we can make predictions!).

Bibliography

[1] T. Abraham and J. Roddck. Survey of spatio-temporal databases. Technical Report CS-96-011, Advanced Computing Research Centre, School of Computer and Information Science, The University Of South Australia, December 1996.

[2] K. K. Al-Taha, R. T. Snodgrass, and M. D. Soo. Bibliography on spatiotemporal databases. *ACM Sigmod Record*, 22(1):59–67, March 1993.

[3] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[4] M. C. Allmen. *Image Sequence Description Using Spatiotemporal Flow Curves: Toward Motion-Based Recognition.* PhD thesis, Department of Computer Science, The University of Wisconsin – Madison, 1991.

[5] F. Arman, A. Hsu, and M.-Y. Chiu. Feature management for large video databases. In *Proceedings of SPIE: Storage and Retrieval for Image and Video Databases*, pages 2–12. The International Society for Optical Engineering, February 1993.

[6] R. Barber, W. Equitz, M. Flicker, W. Niblack, D. Petkovic, and P. Yanker. Efficient query by image content for very large image database. In *IEEE Compcon Spring*, pages 17–19. 1993.

[7] P. B. Berra, C.-Y. R. Chen, A. Ghafoor, and T. D. C. Little. Issues in networking and data management of distributed multimedia systems. In *Proceeding of Sysmposium on High-Performance Distributed Computing*, September 1992.

[8] E. Bertino, M. Damiani, and P. Randi. Multimedia data handling in a knowledge representation system. In *Future Database '92*, volume 3 of *Advanced Database Research and Development Series*, pages 133–141. World Scientific, April 1992.

[9] E. Bertino, F. Rabiti, and S. Gibbs. Query processing in a multimedia document system. *ACM Transactions on Office Information Systems*, 8(1):1–41, January 1988.

[10] A. Bimbo, E. Vicario, and D. Zingoni. Symbolic description and visual querying of image sequences using spatio-temporal logic. *IEEE Transaction on Knowledge and Data Engineering*, 7(4):609–621, August 1995.

[11] J. Boreczky and L. Rowe. Comparison of video shot boundary detection techniques. In *Proceedings SPIE: Storage and Retrieval for Image and Video Databases IV*, volume 2670, pages 170–179, February 1996.

[12] W. Chae. An indexing scheme for 2D-PIR based image databases. Master's thesis, Computer Science and Engineering, The University Of New South Wales, Australia, September 1996.

[13] C.-C. Chang, J.-H. Jiang, and J.-J. Shen. Organization of pictorial databases for spatial match retrieval. In *Proc. International Symposium on Next Generation Database Systems and Their Applications*, pages 205–210. 1993.

[14] S. K. Chang, Q. Y. Shi, and C. W. Yan. Iconic indexing by 2D strings. *IEEE Transactions on Pattern Recognition and Machine Intelligent*, PAMI-9(3):413–428, 1987.

[15] C. Chen and C. Chang. An object-oriented similarity retrieval algorithm for iconic image databases. *Pattern Recognition Letters*, 14:465–470, 1993.

[16] H.-J. Chen and T. D. C. Little. Physical storage organizations for time-dependent multimedia data. In *4th Interntaional Conference of Data Organization and Algorithms (FODO'93)*, October 1993.

[17] S. Christodoulakis and A. Analyti. Guest editorial: Special issue on multimedia information systems. *Information Systems*, 20(6):443–444, 1995.

[18] T.-S. Chua, S.-K. Lim, and H.-K. Pung. Content-based retrieval of segmented images. In *Proceedings of The Second Annual ACM Multimedia Conference*, 1994.

[19] T.-S. Chua and L.-Q. Ruan. Video retrieval and sequencing system. *ACM Transactions on Information Systems*, 13(4):373–407, October 1995.

[20] E. Clementini and P. Felice. A comparison of methods for reprsentating topological relationships. *Information Science Applications: An International Journal*, 3(3):149–178, May 1995.

[21] E. Clementini, P. Felice, and P. van Oostrom. A small set of formal topological relationships suitable for end-user interaction. In *Advances in Spatial Databases: Third International Symposium–SSd'93*, volume 692 of *Lecture Notes in Computer Science*, pages 277–295. Springer-Verlag, 1993.

[22] Z. Cui, A. G. Cohn, and D. A. Randell. Qualitative and topological relationship in spatial databases. In D. Abel and B.-C. Ooi, editors, *Advances in Spatial Databases*, volume 692 of *Lecture Notes in Computer Science*, pages 296–315. Springer-Verlag, 1993.

[23] A. de Moor. A method for structuring the multimedia application development process. Master's thesis, Tilburg University, February 1993.

[24] N. Dimitrova. Motion extraction and video database retrieval. Technical report, Department of Computer Science and Engineering, Arizona State University, June 1994.

[25] N. Dimitrova and F. Golshani. Rx for semantic video database retrieval. In *Proceedings Second Annual ACM Multimedia Conference*, October 15-20 1994.

[26] N. Dimitrova and F. Golshani. Motion recovery for video content classification. *ACM Transactions on Information Systems*, 13(4):408–439, October 1995.

[27] M. J. Egenhofer. Point-set topological spatial relations. *Int. J. Geographical Information Systems*, 5(2):161–174, 1991.

[28] M. J. Egenhofer. Reasoning about binary topological relation. In *Advances in Spatial Databases: Second International Symposium, SSD '91*, number 525 in Lecture Notes in Computer Science, pages 143–160. Springer-Verlag, 1991.

[29] M. J. Egenhofer and K. K. Al-Taha. Reasoning about gradual changes of topological relationships. In A. Frank, I. Campari, and U. Formentini, editors, *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, number 639 in Lecture Notes in Computer Science, pages 196–219. Springer-Verlag, 1992.

[30] M. J. Egenhofer and J. Sharma. Topological relations between regions in r^2 and z^2. In *Advances in Spatial Databases: Third International Symposium, SSD '93*, number 692 in Lecture Notes in Computer Science, pages 316–336. Springer-Verlag, 1993.

[31] C. Faloutsos, W. Equitz, M. Flickner, W. Niblack, D. Petkovic, and R. Barber. Efficient and effective querying by image content. *J. of Intelligent Information Systems*, 3:231–262, July 1994.

[32] M. Flickner, H. Sawhney, W. Niblac, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The QBIC system. *IEEE Computer*, pages 23–32, September 1995.

[33] H. Freeman. On the encoding of arbitrary geometric configurations. *IEEE Trans. Elec. Computers*, EC-10:260–268, 1961.

[34] H. Freeman. Computer processing of line drawings. *Comput. Surveys*, 6:57–97, 1974.

[35] C. Freksa. Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54(1-2):199–227, 1992.

[36] S. K. Gadia. Parametric databases: seamless integration of spatial, temporal, belief and ordinary data. *Sigmod Record*, 22(1):15–20, March 1993.

[37] S. K. Gadia, V. Chopra, and U. S. Tim. An SQL-like seamless query language for spatio-temporal data. In R. Snodgrass, editor, *Proceedings of the International Workshop on an Infrastructure for Temporal Databases*, pages Q1–Q20. June 1993.

[38] A. Galton. Towards an integrated logic of space, time, and motion. In *Proc. IJCAI 93*, pages 1150–1555. Morgan Kaufman, 1993.

[39] S. Ghandeharizadeh and L. Ramos. An evaluation of three virtual replican strategies for continuous retrieval of multimedia data. In *International Symposium on Next Generation Database Systems and Their Applications*, pages 188–197, September 1993.

[40] S. Gosh. File organization: The consecutive retrieval proverty. *Communication of ACM*, 15:802–808, 1972.

[41] R. Gould. *Graph Theory*. The Benjamin/Cummings Publishing Company, Inc, California, 1988.

[42] W. I. Grosky and R. Mehrota. Index-based object recognition in pictorial data management. *Computer Vision, Graphics, and Image Processing*, 52:416–436, December 1990.

[43] W. I. Grosky, P. Neo, and R. Mehrota. A pictorial index mechanism for model-based matching. *Data & Knowledge Engineering*, 8:309–325, 1992.

[44] V. Gudivada. On spatial similarity measures for multimedia applications. In W. Niblack and R. C. Jain, editors, *Storage and Retrieval for Image and Video III*, volume 2420 of *SPIE P Proceedings Series*, pages 363–372. February 1995.

[45] V. Gudivada and V. Raghavan. Design and evaluation of algorithms for image retrieval by spatial similarity. *ACM Transactions on Information Systems*, 13(1):115–144, April 1995.

[46] O. Guenther and A. Buchmann. Research issues in spatial databases. *ACM Sigmod Record*, 19(4):61–68, December 1990.

[47] D. Hernandez. Mantaining qualitative spatial knowledge. In A. U. Frank and I. Campari, editors, *Spatial Information Theory, A Theoretical Basis for GIS*, volume 716 of *Lecture Notes in Computer Science*, pages 37–53. Springer-Verlag, 1993.

[48] R. Hjelsvold. Video information contents and architecture. In *Lecture Notes In Computer Science*, pages 259–272. 1994.

[49] IBM. Qbic(tm)–IBM's query by image content. http://wwwqbic.almaden.ibm.com/.

[50] H. Ishikawa, F. Suzuki, F. Kozakura, A. Makinouchi, M. Miyagishima, Y. Izumida, M. Aoushima, and Y. Yamane. The model, language, and implementation of an object-oriented multimedia knowledge based management system. *ACM Transcation on Database Systems*, 18(1):1–50, March 1993.

[51] T. Ishimaru and S. Uemura. Object-oriented database system for portable multimedia object. In *International Symposium on Next Generation Database Systems and Their Applications*, pages 174–179, September 1993.

[52] H. V. Jagadish. A retrieval technique for similar shape. In *Proceeding of the 1991 ACM SIGMOD International Conference on Management of Data*, pages 208–217, May 29–31 1991.

[53] C. S. Jensen. A consensus glossary of temporal database concepts. Institute for Electronic Systems Department of Mathematics and Computer Science, AalBorg University, Denmark, November 1993.

[54] E. Jungert. Extended symbolic projections as a knowledge structure for spatial reasoning. In *Pattern Recognition: 4th International Conference*, volume 301, pages 343–351. March 1988.

[55] E. Jungert. Qualitative spatial reasoning from the observer's point of view–toward a generalisation of symbolic projection. *Pattern Recognition*, 27(6):801–813, 1994.

[56] E. Jungert. Rotation invariance in symbolic slope projection as a means to support spatial reasoning. In *Workshop on Spatial and Temporal Interaction: Representation and Reasoning*, pages 166–170. 1994.

[57] D. Kahn. *Topology: an Introduction to point-set and algebraic area*. Williams and Wilkins, 1975.

[58] K. Kaneko, S. Kuroki, and A. Makinouchi. Design of 3D CG data model of "move" animation database system. In *Future Database '92*, volume 3 of *Advanced Database Research and Development Series*, pages 364–372. World Scientific, April 1992.

[59] K. Kawagoe. Continuous media data management. *Sigmod Record*, 20(3):74–75, September 1991.

[60] A. Khokhar and A. Ghafoor. A heterogeneous processing (hp) framework for multimedia query processing. In *IEEE Workshop on HP 1994*.

[61] N. Kline. An update of the temporal database bibliography. Technical Report TEMPIS No. 58, Department of Computer Science, The University of Arizona, 1993.

[62] R. Kurniawati, J. Jin, and J. A. Shepherd. Techniques for supporting efficient content-based retrieval in multimedia databases. Technical report, School of Computer Science and Engineering, University of New South Wales, Sydney, 2052, Mar. 1997. Accepted for publication.

[63] G. Langran. *Time in Geographic Information Systems*. Taylor & Fancis, London, 1992.

[64] K. Lee, W. Kim, J. Hahm, and S. Lee. Temporal spesification and synchronization for multimedia database queries. In *International Symposium on*

Next Generation Database Systems and Their Applications, pages 198–204, September 1993.

[65] S. Lee, M. Yang, and J. Chen. Signature file as a spatial filter for iconic database. *Journal of Visual Languages and Computing*, (3):373–397, 1992.

[66] S.-Y. Lee and F.-J. Hsu. 2D C-string: A new spatial knowledge representation for image database systems. *Pattern Recognition*, 23(10):1077–1087, 1990.

[67] S.-Y. Lee and F.-J. Hsu. Picture algebra for spatial reasoning of iconic images represented in 2D C-string. *Pattern Recognition Letters*, 12:425–435, July 1991.

[68] S.-Y. Lee and F.-J. Hsu. Spatial reasoning and similarity retrieval of images using 2D C-string knowledge representation. *Pattern Recognition*, 25(3):305–318, 1992.

[69] S.-Y. Lee and H.-M. Kao. Video indexing – an approach based on moving object and track. In *Proceedings of SPIE: Storage and Retrieval for Image and Video Databases*, pages 25–36. The International Society for Optical Engineering, February 1993.

[70] S.-Y. Lee, M. Shan, and W.-P. Yang. Similarity retrieval of iconic image database. *Pattern Recognition*, 22(6):675–682, 1989.

[71] K. Li, N. Badji, and R. Laurini. Modeling moving objects is spatiotemporal databases. Technical report, Department of Information and Communication Engineering, KAIST, 1994.

[72] T. D. C. Little, G. Ahanger, R. J. Folz, J. F. Gibbon, F. W. Reeve, D. H. Schelleng, and D. Vankatesh. A digital on-demand video service supporting content based queries. In *Proceedings of First International Conference on Multimedia*, pages 427–436, August 1993.

[73] T. D. C. Little, A. Gafoor, C.-Y. R. Chen, C. S. Chang, and P. Berra. Multimedia synchronization. *IEEE Data Engineering Bulletin*, 14(3):26–35, September 1991.

[74] G. Lu. Image retrieval based on shapes. In *Visual'96: First International Conference on Visual Information System*, pages 126–135. February 1996.

[75] H. Lu, B.-C. Ooi, and K.-L. Tan. Efficient image retrieval by color content. Department of Information Systems and Computer Science, National University of Singapore, 1994.

[76] C. Meghini, F. Rabitti, C. Thanos, and C. N. delle Ricerche. Conceptual modelling of multimedia documents. *IEEE Computer*, 24(10):23–30, October 1991.

[77] R. Mehrota and J. E. Gary. Feature-based retrieval of similar shapes. In *9th International Conference on Data Engineering*, pages 108–115, April 19-23 1993.

[78] R. Mehrota and J. E. M. Gary. Similar-shape retrieval in shape data management. *IEEE Computer*, pages 57–62, September 1995.

[79] B. Mehtre, M. Kankanhalli, A. Narasimhalu, and G. Man. Color matching for image retrieval. Technical Report TR94–138–0, Institute System Science, 1994.

[80] H. Mo, S. Satoh, and M. Sakauchi. A study of image recognition using similarity retrieval. In *Visual'96: First International Conference on Visual Information Systems*, pages 136–151, February 1996.

[81] J. R. Munkres. *Topology A First Course*. Prentice-Hall, Inc., Englewood Cliffs,New Jersey, 1975.

[82] M. Nabil. Interaction between topological and interval relationships in two dimensional space. In *Proceedings of Pan-Sydney Area Workshops on Visual Information Processing*, pages 28–34, November 1996.

[83] M. Nabil, A. Ngu, and J. Shepperd. Modelling moving object in multimedia databases. In *Database Systems for Advanced Applications, DASFAA'97*, volume 6 of *Advanced Database Research and Development Series*, pages 67–75, April 1997.

[84] M. Nabil, A. H. H. Ngu, and J. Shepherd. Modelling spatiotemporal relationships in multimedia database. In *Workshop in Spatial and Temporal Interaction: Representation & Reasoning*, pages 171–189, November 8 1994.

[85] M. Nabil, A. H. H. Ngu, and J. Shepherd. A unified representation of spatial and temporal information in multimedia database. In *Proceedings of The Third International Conference on Automation, Robotics and Computer Vision*, pages 374–378, November 9-11 1994.

[86] M. Nabil, A. H. H. Ngu, and J. Shepperd. Picture similarity retrieval using 2D projection interval representation. *IEEE Transactions on Knowledge and Data Engineering*, 8(4):533–539, 1996.

[87] M. Nabil, J. Shepperd, and A. H. H. Ngu. 2D projection interval relationships: A symbolic representation of spatial relationships. In *Advances in Spatial Databases: Forth International Symposium, SSD'95*, volume 951 of *Lecture Notes in Computer Science*, pages 292–309. Springer-Verlag, 1995.

[88] A. Nagasaka and Y. Tanaka. Automatic video indexing and full-video search for object appearances. In *Visual Database Systems II*, pages 113–127. Elsevier Science Publishers, 1992.

[89] Y. Nah and S. Lee. Object-relationship model for conceptual modelling of multimedia data. In *Future Database '92*, volume 3 of *Advanced Database Research and Development Series*, pages 364–372. World Scientific, April 1992.

[90] W. Niblack, R. Barber, W. Equitz, M. Flicker, E. Glasman, D. Petkovic, P. Yanker, C. Falautsos, and G. Taubin. The QBIC project: Querying im-

ages by content using color, texture, and shape. Research Report, Report Rj 9203(81511), IBM Almanden Research Center, San Jose, CA., 1993.

[91] E. Oomoto and K. Tanaka. OVID: Design and implementation of a video-object database system. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):629–643, August 1994.

[92] D. Papadias and T. Sellis. The semantics of relations in 2D space using representative points: Spatial indexes. In A. U. Frank and I. Campari, editors, *Spatial Information Theory, A Theoretical Basis for GIS*, volume 716 of *Lecture Notes in Computer Science*, pages 234–247. Springer-Verlag, 1993.

[93] D. J. Peuquet and Z. Ci-Xiang. An algorithm to determine the directional relationship between arbitrary-shaped polygons in the plane. *Pattern Recognition*, 20(1):65–74, 1987.

[94] K. Rohr. Towards model-based recognition of human movements in image sequences. *CVIP: Image Understanding*, 59(1):94–115, January 1994.

[95] L. Rowe, J. Boreczky, and C. Eads. Indexes for user access to large video databases. In *Proceedings SPIE: Storage and Retrieval for Image and Video Databases II*, volume 2185, pages 150–161, February 1994.

[96] C. L. Ruggles. Structuring image data within a multi-media information system. *International Journal of Geographical Information Systems*, 6(3):205–222, 1992.

[97] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addision-Wesley Publishing Co., 1990.

[98] B. Scassellati, S. Alexopoulus, and M. Flickner. Retrieving images by 2D shape: a comparison of computation method with human perceptual judgements. In *Proceedings SPIE'94: Storage and Retrieval for Image and Video Databases II*, pages 2–14, February 1994.

[99] S.Devadiga, D. Kosiba, U. Gargi, S. Oswld, and R. Kasturi. A semiaoutomatic video database system. In *Proceedings of SPIE: Storage and Retrieval for Image and Video Databases*, pages 262–267. The International Society for Optical Engineering, February 1995.

[100] K. Shearer, S. Vankatesh, and D. Kieronska. The visitors guide: a simple video reuse application. Departement of Computer Science, Curtin University of Technology, Australia, 1996.

[101] A. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and querying moving objects. In *Proceeding of ICDE'97*, April 1997.

[102] A. Sistla and C. Yu. *Retrieval of Pictures Using Approximate Matching*, chapter Five, pages 101–112. Springer-Verlag, 1996.

[103] A. P. Sistla, C. Yu, and R. Haddad. Reasoning about spatial relationships in picture retrieval systems. In *Proceedings of VLDB'94*, September 1994.

[104] A. P. Sistla, C. Yu, C. Liu, and K. Liu. Similarity based retrieval of pictures using indices on spatial relationships. In *Proceedings of VLDB'95*, September 1995.

[105] S. Smoliar and H.-J. Zhang. Content-based video indexing and retrieval. *IEEE Multimedia*, 1(2):62–72, 1994.

[106] M. A. Stricker. Bounds for the descrimination power of color indexing techniques. In *Proceedings SPIE'94: Storage and Retrieval for Image and Video Databases II*, pages 15–24, February 1994.

[107] M. Tabuchi and Y. Muraoka. MeSOD: the metric spatial object data model for a multimedia application: Hyperbook. In *1989 IEEE Compcon Spring*, pages 396–400, February 1989.

[108] M. Tabuchi, Y. Yagawa, M. Fujisawa, A. Negishi, and Y. Muraoka. Hyperbook: A multimedia information system that permits incomplete queries. In

International Conference on Multimedia Information Systems '91, pages 3–16. Institute of System Science National University Singapore, McGraww-Hill Pub. Co., 1991.

[109] T. Takahashi, N. Shima, and F. Kishino. An image retrieval method using inqueries on spatial relationships. *Journal of Information Processing*, 15(3):441–449, 1992.

[110] J. van Leeuwen. Graph algorithm. In *Handbook of Theoritical Computer Science*. The MIT Press, 1990.

[111] L. Vieu. Logical framework for reasoning about space. In *Spatial Information Theory: A Theoretical Basis for GIS*, volume 716 of *Lecture Notes in Computer Science*, pages 25–35. Springer-Verlag, 1993.

[112] Virage Inc. Virage visual information retrieval. http://www.virage.com/.

[113] R. Weiss and A. Duda. Composition and search with a video algebra. *IEEE Multimedia*, 2(1):12–25, 1995.

[114] D. Woelk, W. Kim, and W. Luther. An object-oriented approach to multimedia databases. In *Proceedings of the ACM SIGMOD Conference*, pages 592–606, 1986.

[115] M. F. Worboys. Object-oriented approach to geo-referenced information. *International Journal of Geographical Information Systems*, 8(4):385–399, 1994.

[116] M. F. Worboys. A unified model for spatial and temporal information. *The Computer Journal*, 37(1):26–33, 1994.

[117] C. Yu, K. Nair, C. Liu, and Y. Aslandogan. Experiences in content based approximate picture retrieval. Department of Electrical Engineering and Computer Science, University of Illinois at Chicago, 1994.

[118] H. J. Zhang, S. W. Smoliar, and J. H. Wu. Development of a video database system. In *Workshops on Digital Libraries*, May 1994.

[119] X. Zhou and C. Ang. Retrieving similar pictures in pictorial database by an improved hashing table. Technical Report TRA1/97, Department of System and Computer Science, The National University of Singapore, January 1997.

[120] K. Zimmermann. Enhancing qualitative spatial reasoning combining orientation and distance. In A. U. Frank and I. Campari, editors, *Spatial Information Theory, A Theoretical Basis for GIS*, volume 716 of *Lecture Notes in Computer Science*, pages 68–76. Springer-Verlag, 1993.

Appendix A

Similarity retrieval experiments

Result similarity experiments using 2D-PIR and Gudivada's methods on images of Sydney.

Query Id			0	20	40	50	60	80
S1	2D-PIR	Precision	29.8	37.8	70.0	92.9	100	100
		Recall	100	100	100	92.9	21.4	7.1
	Gudivada	Precision	29.8	60.8	62.5	66.7	66.7	100
		Recall	100	100	35.7	14.3	14.3	14.3
S2	2D-PIR	Precision	34.0	36.4	64.0	71.4	91.7	100
		Recall	100	100	100	93.8	68.8	31.3
	Gudivada	Precision	34.4.	34.8	57.1	55.6	57.6	68.8
		Recall	100	100	100	93.8	87.5	68.8
S6	2D-PIR	Precision	38.3	45.2	90.0	100	100	100
		Recall	100	100	94.7	84.2	33.3	5.2
	Gudivada	Precision	38.3	69.2	90.0	100	100	100
		Recall	100	94.7	50.0	22.2	22.2	22.2
S7	2D-PIR	Precision	24.6	36.9	83.3	100	100	100
		Recall	100	100	88.2	47.1	11.8	5.7
	Gudivada	Precision	24.6	61.5	92.3	100	100	100
		Recall	100	94.1	70.6	11.8	11.8	11.8
S8	2D-PIR	Precision	22.9	36.2	88.3	85.7	100	100
		Recall	100	100	88.2	35.3	29.4	5.9
	Gudivada	Precision	22.9	64.0	100	100	100	100
		Recall	100	94.1	29.4	29.4	5.9	5.9
Geometric mean	2D-PIR	Precision	29.3	38.4	77.5	89.3	98.3	100
		Recall	100	100	94.0	65.7	27.9	8.3
	Gudivada	Precision	29.3	56.5	80.2	81.4	82.6	92.8
		Recall	100	96.5	51.7	25.0	18.1	17.2

Result similarity experiments using 2D-PIR and Gudivada's methods on images of antelopes.

Query Id			0	20	40	50	60	80
A3	2D-PIR	Precision	36.8	50.0	80.0	80.0	100	100
		Recall	87.5	87.5	50.0	50.0	12.5	12.5
	Gudivada	Precision	36.8	46.7	66.7	66.7	66.7	66.7
		Recall	87.5	87.5	25.0	25.0	25.0	25.0
A15	2D-PIR	Precision	19.4	37.5	100	100	100	100
		Recall	100	85.7	71.4	42.9	28.6	14.2
	Gudivada	Precision	19.4	55.6	80.0	100	100	100
		Recall	100	71.4	57.1	28.6	28.6	28.6
A16	2D-PIR	Precision	22.7	33.3	60.0	100	100	100
		Recall	100	80.0	60.0	60.0	60.0	20.0
	Gudivada	Precision	22.7	23.5	100	100	100	100
		Recall	100	80.0	60.0	60.0	60.0	60.0
A26	2D-PIR	Precision	50.0	50.0	83.3	83.3	100	100
		Recall	100	100	100	100	60.0	40.0
	Gudivada	Precision	50.0	55.6	55.6	55.6	62.5	62.5
		Recall	100	100	100	100	100	100
A2	2D-PIR	Precision	38.5	38.5	55.6	55.6	100	100
		Recall	62.5	62.5	62.5	62.5	42.8	28.6
	Gudivada	Precision	38.5	38.5	38.5	38.5	38.5	50.0
		Recall	62.5	62.5	62.5	62.5	62.5	62.5
Geometric mean	2D-PIR	Precision	31.5	41.3	74.0	82.0	100	100
		Recall	88.6	82.5	66.7	60.4	35.3	21.0
	Gudivada	Precision	31.5	42.0	64.8	67.8	69.4	73.1
		Recall	88.6	79.2	55.7	48.5	48.5	48.5

Result similarity experiments using 2D-PIR and Gudivada's methods on images of falcons.

Query Id			0	20	40	50	60	80
B1	2D-PIR	Precision	62.5	71.4	71.4	71.4	83.3	100
		Recall	100	100	100	100	100	80.0
	Gudivada	Precision	62.5	66.7	66.7	66.7	75.0	100
		Recall	100	80.0	80.0	80.0	60.0	60.0
B5	2D-PIR	Precision	18.9	66.7	100	100	100	100
		Recall	100	100	50.0	50.0	50.0	50.0
	Gudivada	Precision	18.9	100	100	100	100	100
		Recall	100	50.0	50.0	50.0	50.0	50.0
B10	2D-PIR	Precision	100	100	100	100	100	100
		Recall	75.0	75.0	75.0	50.0	50.0	50.0
	Gudivada	Precision	100	100	100	100	100	100
		Recall	75.0	25.0	25.0	25.0	25.0	25.0
B11	2D-PIR	Precision	50.0	57.1	57.1	57.1	57.1	100
		Recall	80.0	80.0	80.0	80.0	80.0	80.0
	Gudivada	Precision	50.0	66.7	66.7	66.7	60.0	100
		Recall	80.0	80.0	80.0	80.0	60.0	60.0
B14	2D-PIR	Precision	62.5	71.4	71.4	71.4	80.0	100
		Recall	100	100	100	100	80.0	20.0
	Gudivada	Precision	62.5	60.0	75.0	75.0	100	100
		Recall	100	60.0	60.0	60.0	40.0	40.0
Geometric mean	2D-PIR	Precision	51.7	72.0	71.8	78.1	82.4	100
		Recall	90.3	90.3	78.6	78.6	69.3	50.2
	Gudivada	Precision	51.7	76.8	80.3	80.3	85.2	100
		Recall	90.3	54.5	54.5	54.5	44.8	44.8

Result similarity experiments using 2D-PIR and Gudivada's methods on images of sunset.

Query Id			0	20	40	50	60	80
M3	2D-PIR	Precision	33.3	45.0	83.3	100	100	100
		Recall	100	81.8	45.5	27.3	18.2	9.1
	Gudivada	Precision	33.3	33.3	100	100	100	100
		Recall	100	100	27.3	27.3	27.3	27.3
M13	2D-PIR	Precision	33.3	43.8	100	100	100	100
		Recall	88.9	77.9	44.4	44.4	22.2	22.2
	Gudivada	Precision	33.3	33.3	33.3	33.3	33.3	33.3
		Recall	88.9	88.9	88.9	88.9	88.9	88.9
M20	2D-PIR	Precision	12.5	15.4	36.4	36.4	100	100
		Recall	100	100	100	100	50.0	25.0
	Gudivada	Precision	12.5	13.8	50.0	50.0	50.0	66.7
		Recall	100	100	50.0	50.0	50.0	50.0
M25	2D-PIR	Precision	37.5	50.0	100	100	100	100
		Recall	100	75.0	50.0	50.0	25.0	25.0
	Gudivada	Precision	37.5	37.5	37.5	37.5	37.5	37.5
		Recall	75.0	75.0	75.0	75.0	75.0	75.0
M26	2D-PIR	Precision	18.5	19.0	75.0	75.0	75.0	100
		Recall	83.3	57.1	42.9	42.9	42.9	28.6
	Gudivada	Precision	18.5	66.7	66.7	66.7	100	100
		Recall	83.3	42.9	28.6	28.6	28.6	28.6
Geometric mean	2D-PIR	Precision	24.9	31.0	74.4	77.1	94.4	100
		Recall	88.9	77.1	53.4	48.2	29.3	20.3
	Gudivada	Precision	24.9	32.9	52.9	52.9	57.4	60.8
		Recall	88.9	77.9	48.2	48.2	48.2	48.2

Result similarity experiments using 2D-PIR and Gudivada's methods on images of monkeys.

Query Id			0	20	40	50	60	80
K1	2D-PIR	Precision	20.0	28.9	100	100	100	100
		Recall	100	100	50.0	50.0	25.0	25.0
	Gudivada	Precision	20.0	100	100	100	100	100
		Recall	100	50	50.0	50.0	50.0	50.0
K5	2D-PIR	Precision	37.5	60.0	100	100	100	100
		Recall	75.0	75.0	50.0	50.0	50.0	25.0
	Gudivada	Precision	37.5	75.0	75.0	75.0	75.0	75.0
		Recall	75.0	75.0	75.0	75.0	75.0	75.0
K6	2D-PIR	Precision	28.5	50.0	100	100	100	100
		Recall	100	100	100	100	100	100
	Gudivada	Precision	28.5	33.3	40.0	50.0	50.0	50.0
		Recall	100	100	100	100	100	50.0
K8	2D-PIR	Precision	28.5	57.1	80.0	100	100	100
		Recall	100	100	100	100	50.0	25.0
	Gudivada	Precision	28.5	57.1	100	100	100	100
		Recall	100	100	100	100	75.0	50.0
K12	2D-PIR	Precision	37.5	75.0	100	100	100	100
		Recall	100	100	50.0	50.0	50.0	25.0
	Gudivada	Precision	37.5	75.0	75.0	75.0	75.0	75.0
		Recall	100	100	100	100	100	100
Geometric mean	2D-PIR	Precision	29.6	51.8	95.6	100	100	100
		Recall	94.4	94.4	66.0	66.0	50.0	33.0
	Gudivada	Precision	29.6	64.1	74.2	77.6	77.6	77.6
		Recall	94.4	82.2	82.2	82.2	77.6	62.3

Result similarity experiments using 2D-PIR and Gudivada's methods on images of hawks.

Query Id			0	20	40	50	60	80
H4	2D-PIR	Precision	45.0	45.0	62.5	83.0	100	100
		Recall	100	100	100	100	40.0	20.0
	Gudivada	Precision	45.0	57.0	100	100	100	100
		Recall	100	80.0	20.0	20.0	20.0	20.0
H5	2D-PIR	Precision	73.0	73.0	100	100	100	100
		Recall	87.5	87.5	62.5	37.5	25.0	12.5
	Gudivada	Precision	73.0	71.4	100	100	100	100
		Recall	87.5	62.5	25.0	25.0	25.0	12.5
H11	2D-PIR	Precision	70.0	70.0	100	100	100	100
		Recall	100	100	100	100	71	42.0
	Gudivada	Precision	70.0	70.0	70.0	70.0	90.0	71.0
		Recall	100	100	100	100	100	71.0
H13	2D-PIR	Precision	20.0	27.0	100	100	100	100
		Recall	100	100	67.0	67.0	33.0	33.0
	Gudivada	Precision	20.0	100	100	100	100	100
		Recall	100	67.0	67.0	67.0	67.0	33.0
H17	2D-PIR	Precision	70.0	70.0	100	100	100	100
		Recall	87.5	87.5	75.0	62.5	62.5	25.0
	Gudivada	Precision	70.0	70.0	70.0	70.0	75.0	100
		Recall	87.5	87.5	87.5	87.5	75.0	50.0
Geometric mean	2D-PIR	Precision	50.3	53.4	91.0	96.3	100	100
		Recall	94.8	94.8	79.3	69.1	43.0	24.4
	Gudivada	Precision	50.3	72.4	86.7	86.7	92.4	93.4
		Recall	94.8	78.2	49.4	49.4	47.9	31.1

Result similarity experiments using 2D-PIR and Gudivada's methods on images of owls.

Query Id			0	20	40	50	60	80
O1	2D-PIR	Precision	26.7	26.7	100	100	100	100
		Recall	80.0	80.0	80.0	60.0	40.0	40.0
	Gudivada	Precision	26.7	36.4	100	100	100	100
		Recall	80.0	80.0	80.0	80.0	80.0	80.0
O2	2D-PIR	Precision	20.0	23.0	60.0	75.0	100	100
		Recall	100	100	100	100	100	100
	Gudivada	Precision	20.0	25.0	27.3	27.3	27.3	33.3
		Recall	100	100	100	100	100	100
O10	2D-PIR	Precision	41.0	100	100	100	100	100
		Recall	100	100	85.7	71.4	42.8	14.3
	Gudivada	Precision	41.0	100	100	100	100	100
		Recall	100	71.4	71.4	71.4	28.6	28.6
O15	2D-PIR	Precision	47.1	66.7	100	100	100	100
		Recall	100	62.5	62.5	62.5	37.5	12.5
	Gudivada	Precision	100	100	100	100	100	100
		Recall	62.5	62.5	62.5	662.5	62.5	62.5
O3	2D-PIR	Precision	35.3	45.0	100	100	100	100
		Recall	100	83.3	83.3	83.3	83.3	16.7
	Gudivada	Precision	100	100	100	100	100	100
		Recall	83.3	83.3	83.3	83.3	83.3	83.3
Geometric mean	2D-PIR	Precision	32.5	45.0	90.3	94.4	100	100
		Recall	95.6	83.9	81.4	74.1	61.7	24.0
	Gudivada	Precision	32.5	61.9	77.1	77.1	77.1	80.3
		Recall	96.6	83.9	78.5	78.5	65.3	65.3

Result similarity experiments using 2D-PIR and Gudivada's methods on images of mountains.

Query Id			0	20	40	50	60	80
W1	2D-PIR	Precision	9.5	28.6	60.0	100	100	100
		Recall	100	100	100	25.0	25.0	25.0
	Gudivada	Precision	9.5	37.5	50.0	50.0	100	100
		Recall	100	75.0	50.0	25.0	25.0	25.0
W2	2D-PIR	Precision	14.3	19.0	42.9	50.0	100	100
		Recall	100	100	60.0	40.0	20.0	20.0
	Gudivada	Precision	14.3	37.4	40.0	100	100	100
		Recall	100	60.0	40.0	20.0	20.0	20.0
W5	2D-PIR	Precision	18.8	33.3	66.7	100	100	100
		Recall	100	100	66.7	66.7	66.7	33.3
	Gudivada	Precision	18.8	75.0	75.0	75.0	75.0	75.0
		Recall	100	75.0	75.0	75.0	75.0	75.0
W8	2D-PIR	Precision	16.7	30.0	62.5	83.3	100	100
		Recall	100	100	71.4	71.4	28.6	14.3
	Gudivada	Precision	16.7	50.0	71.4	66.7	100	100
		Recall	100	71.4	71.4	28.6	28.6	14.3
W13	2D-PIR	Precision	13.3	21.0	50.0	60.0	100	100
		Recall	100	100	75.0	75.0	25.0	25.0
	Gudivada	Precision	13.3	33.3	42.8	42.8	100	100
		Recall	100	75.0	75.0	75.0	25.0	25.0
Geometric mean	2D-PIR	Precision	14.2	25.8	55.7	75.8	100	100
		Recall	100	100	73.5	51.4	30.0	22.6
	Gudivada	Precision	14.2	33.8	54.0	64.0	94.4	94.4
		Recall	100	79.7	60.4	38.1	30.1	26.6