Multimedia Tools and Applications, 5, 275?? (1997)

Modeling and Retrieval of Moving Objects

MOHAMMAD NABIL [3], ANNE H.H NGU AND JOHN SHEPHERD anne@cse.unsw.edu.au
School of Computer Science and Engineering, University of New South WalesSydney 2052, NSW, Australia

Received November 25, 1997; Revised March 30, 1998

Editor: XXXX

Abstract. This paper presents a symbolic formalism for modeling and retrieving video data via the moving objects contained in the video images. The model integrates the representations of individual moving objects in a scene with the time-varying relationships between them by incorporating both the notions of object tracks and temporal sequences of PIRs (projection interval relationships). The model is supported by a set of operations which form the basis of a moving object algebra. This algebra allows one to retrieve scenes and information from scenes by specifying both spatial and temporal properties of the objects involved. It also provides operations to create new scenes from existing ones. A prototype implementation is described which allows queries to be specified either via an animation sketch or using the moving object algebra.

Keywords: Multimedia Retrieval, Moving objects Model, Content-based image retrieval, Symbolic video model

1. Introduction

In recent years, we have witnessed a rapid growth of multimedia applications in fields such as entertainment, education and training. 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) and the rapid reduction in the cost of these devices. On the other hand, the software that manages multimedia data is still far from satisfactory. Early multimedia software concentrated primarily on the presentation of multimedia data, dealing with such issues as synchronization of different media streams. More important in the long term, however, will be software that provides efficient storage and retrieval of multimedia data, especially content-based retrieval. The problem of developing such software has been attracting much attention in recent years [5, 15, 2].

Image-based multimedia data can be treated in many different ways. One approach is to consider visual properties of images such as colour, texture, shape, and so on. This allows us to ask queries such as “Find all images that are mainly red in color” or “Find all images which have a mix of colors similar to this example image”. The “visual-features” approach has been actively pursued for some time and has yielded some useful systems (e.g. QBIC [11]).

Another approach is to consider the semantic composition of images in terms of the individual objects contained in them and the spatial relationships among these objects. This would enable us to ask queries such as “Find images that show a red apple *on* a table” or “Find images that show a river running *beneath* a rough stone bridge”. Note that these queries incorporate visual properties as well as spatial relationships between objects.

The preceding examples deal with static image data (e.g. photographs, drawings) where the main relationships among objects are spatial. In dynamic image data (e.g. video,

animation) the relationships include not only spatial, but also temporal and spatiotemporal ones. For video retrieval, we would like to be able to ask queries such as "Find a scene where an apple rolls along the top of a table and falls off to land beside it" or "Find a scene where a man in a blue suit enters a wood-panelled room from the left". A data model that captures the relationships among moving objects will allow such queries to be formulated and answered, and thus provide a basis for content-based video retrieval.

Video retrieval systems are a very important application for multimedia database technology. So far, however, most video retrieval systems have used textual annotations as the basis for retrieval [14, 26]. The problems with using unstructured or semi-structured text for retrieval are well-known. There has also been some work on video retrieval using simple visual features (for example [32]). The current state-of-the-art in video retrieval is reported in [31, 2].

Since video retrieval based on moving objects requires the manipulation of large volumes of both spatial and temporal information, one might expect some help from the substantial body of research in spatial databases [13, 27] and temporal databases [16]. However, research on the integration of space and time in the database area is still in its infancy [3], and a recent survey on spatiotemporal databases [1] does not show much development. Consequently, there has been little work done on content-based video retrieval using spatiotemporal features (in particular, spatiotemporal relationships) [18].

A model for moving objects must be able to deal with both the movement of individual objects and with the dynamically changing relationships among objects. To achieve this objective, we propose a graph-based symbolic model. Each node of the graph contains a description of one particular object and its movement (if any). Each edge in the graph is labeled with the temporal sequence of relationships between the objects in the end-point nodes. Each scene in a video is described by a single graph, where there is one node for each object of interest in the scene, and every pair of objects has an edge describing the time-varying relationship between them.

The remainder of this paper is organized as follows. In the following sub-section, we briefly review the 2D-PIR model for spatial relationships between static objects which forms the basis for our moving objects model. Section 2 discusses in detail the data model for moving objects. Section 3 presents an algebra for moving objects. Section 4 demonstrates retrieval of video via moving objects using the proposed algebra in specific applications. Section 5 discusses similarity based retrieval and how we derive the similarity matrices. Section 6 describes our prototype system for moving object retrieval. Section 7 compares our system with related systems. Finally, in Section 8 we present our conclusions. While this paper is intended to be largely self-contained, interested readers may wish to investigate further details of the 2D-PIR model in [23] and the experimental work in [24] as well as [20].

1.1. A Static Object Data Model (2D-PIR)

As we noted above, one useful view of an image is as a set of objects with associated spatial relationships. In this section we develop a model for describing static images according to this view. We use this model as the basis for building our model for moving objects in subsequent sections.

The first step in developing a representation for an image is to identify the individual objects. In this paper, to keep the presentation simple, we assume that each object is identified by a unique name, and we use use exact matching on these names in answering queries. However, our model is not dependent on the method of matching objects and would adapt well to a more realistic scenario where objects were identified by a collection of semantic and/or visual properties (such as text description, visual features, or even an iconic image) and similarity matching was used to match objects in answering queries.[4]

An object in an image occupies a region of space that can be defined by a set of points (pointset). Given the pointsets for two objects, we can compute a topological relationship between them. Given the pointset of a single object, we can compute its projections on each of the axes.[5] With just this information, we can construct a useful representation of the spatial relationships among objects in an image.

A 2-Dimensional Projection Interval Relationship (2D-PIR) is a symbolic representation of the directional and topological relationships between two objects in a picture which adapts three existing representation formalisms (Allen's temporal intervals [4], Chang et al.'s 2D-strings [8] and topological relationships [10]). 2D-PIR uses the projection concept from the 2D-string representation but differs from 2D-strings in using Allen's interval relationships over the projections of objects along the x and y axes, rather than Chang's $\{<, :, =\}$ relationships.

A 2D-PIR between two objects is a triple (δ, χ, ψ) where δ is the topological relationship between the objects and χ and ψ are the interval relationships between them. The domain of δ is the set $\{dt, to, ct, in, ov, co, eq, cb\}$ denoting the topological relationships *disjoint, meets, contains, inside, overlaps, covers, equal, covered_by*. The domain for χ and ψ is the set $\{<, =, m, o, d, s, f, >, mi, oi, di, si, fi\}$ denoting the interval relationships *before, equal, meets, overlaps, during, start, finish, after, meet-inverse, overlap-inverse, during-inverse, start-inverse, finish-inverse*. Note that χ represents the interval relationship along the x-axis, and ψ represents the interval relationship along the y-axis.

In the 2D-PIR scheme, an image is represented by a graph, where each node in the graph is associated with an object and each edge in the graph indicates the spatial relationship between the two objects in the end-point nodes. Every object in the image has an associated node, and there is a single directed edge between every pair of nodes. We denote the 2D-PIR between two objects A and B by $R_s(A, B)$. Note that $R_s(A, B)$ is different to $R_s(B, A)$.

Figure 1 shows an example image and its corresponding 2D-PIR graph. In this figure, the spatial relationships encoded in 2D-PIR are: $R_s(A, B) = (dt, di, di)$, $R_s(A, C) = (o, d, oi)$ and $R_s(B, C) = (dt, d, >)$. $R_s(B, A)$ is formed simply by taking the inverses of the individual components of $R_s(A, B)$, i.e. $R_s(B, A) = (dt, d, d)$.

Consider the first of these 2D-PIRs, $R_s(A, B) = (dt, di, di)$. The two di's indicate that the projection of A encloses the projection of B on both of the axes. This would normally suggest that A completely contains B, but the topological relationship tells us that they are disjoint. From this, we can that infer A partially surrounds B. This suggests that 2D-PIRs give a good insight into the structure of an image from a very compact representation.

In summary, 2D-PIR allows us to concisely express various important spatial relationships in images, and provides an effective basis for image indexing and retrieval. An important aspect of 2D-PIR is that all of the information for the graph representation can be computed

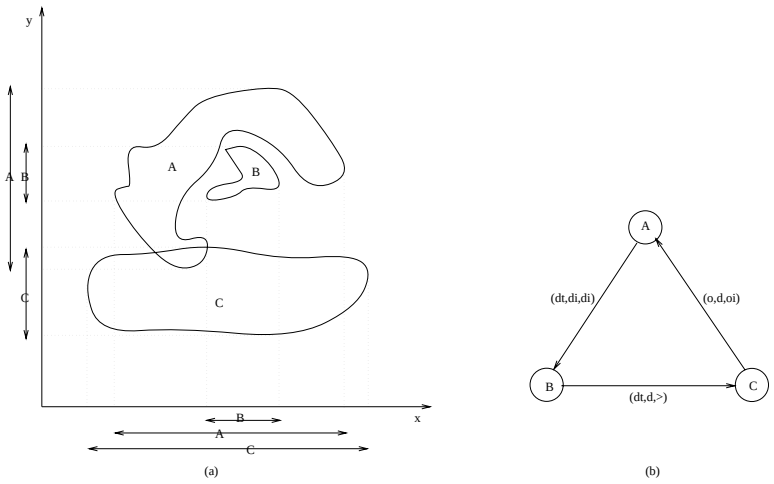


Figure 1. (a) Example image with projections, (b) 2D-PIR representation

automatically once the objects in the image have been identified. A detailed description of 2D-PIR may be found in [23].

2. A Moving Object Data Model

In the previous section, we described a model for static images based on the notion of 2D-PIRs [21, 22]. In this section, we show that a modest extension to this model allows us to represent and manipulate video data.

The obvious difference between static images and video is the introduction of a time dimension. In our work, we assume that video data can be treated as a sequence of static images (called *frames*) captured at successive points in time. The time between successive frames will typically be quite small (less than 0.1 seconds) and so a video scene will generally be made up of hundreds or thousands of images.[6] A complete video work, such as a movie, consists of hundreds or thousands of scenes.

A central notion is our model is a *time interval*, which is defined simply as a pair of time values (t_s, t_f) denoting the start (t_s) and finish (t_f) time-points of the interval. For a time interval $I = (t_s, t_f)$, the notation $start(I)$ refers to t_s and $finish(I)$ refers to t_f. The function $duration(I)$ gives the total length of an interval I (that is $duration(I) = finish(I) - start(I)$). The "+" operator concatenates time intervals as follows: if $I_1 = (t_a, t_b)$ and $I_2 = (t_b, t_c)$, then $I_1 + I_2 = (t_a, t_c)$.

In dealing with video data, we measure the time within a scene relative to the time that the first frame in the scene was captured, so that $t_s = 0$. We denote the time interval for an entire scene S by I_S, which implies that $t_f = I_S$. In a video database, we may also store the absolute time at which the scene was filmed.

As noted above, a stationary object can be defined by a pointset. We can use a single *focus point* on the object as a basis for describing its movement. The focus point could

be defined automatically (e.g. by computing the centroid of the pointset) or could be determined manually (e.g. by a user tracking the motion with a pointing device). Note that we assume that the object is "reasonably" rigid (that is, the object remains as a single connected pointset throughout its motion). We also assume that we are primarily interested in the translational motion of the object.

The motion of a moving object can be described by an ordered sequence of spatiotemporal points. Each spatiotemporal point is of the form $(x_1, x_2, \ldots x_n, t)$, where the x_i's are the n-dimensional spatial coordinates of the focus point, 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), and 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 the focus point at times $t_0, t_1, t_2, \ldots, t_n$, describes the object's movement. We call this the *point representation* of the object's motion.

Retrieval of moving objects based on their *point representation* is not feasible since we would have to consider all of the (large number of) spatiotemporal points associated with each object over every scene. To overcome this problem we use a compact representation of object motion based on *symbolic direction* and *distance*, which is more efficient to store, retrieve and process.

If we consider an object moving in a constant direction over some interval I its movement can be characterized by a triple (d, r, I) where d is the (Euclidean) *distance* travelled by the (focus point of the) object during I and r is the *direction* of the object's movement during I. The domain of d is the real numbers, while the domain of r is the set of possible directions. Directions could be measured in angular units if fine-grained representation of direction is required; in our work, we have chosen to work with the eight-direction domain typically used in the image analysis and retrieval literature (see Figure 2).

To model complex object movement, we break the motion down into a temporal sequence of iso-directional time intervals (in other words, a sequence of linear movements).[7] 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 $start(I_1) = t_0$, $finish(I_1) = start(I_2)$, $finish(I_2) = start(I_3)$ and so on. If an object does not move at all, we associate an empty track with it. If an object moves in a simple fashion (e.g. in a straight line), its track will be very short. If an object moves in a complex non-linear fashion, its track may be quite long (up to the number of time quanta making up the whole scene). At reasonably coarse direction granularity (such as our eight-direction one) object tracks tend to be considerably shorter than the point representation of object movement.

Using these ideas, a moving object is modeled as an *object descriptor* $(Id, [(d, r, I)])$ where Id is the unique name (identifier) of the object and $[(d, r, I)]$ is its track.

Being able to model individual moving objects is quite useful from a querying perspective (for example, we can ask queries about certain kinds of object trajectories). However, a more expressive and useful model would also consider the dynamically changing relationships between objects in a scene. To achieve this more expressive model, we extend our model of static images (2D-PIR graphs) described in section 1.1. Basing the new model on 2D-PIR is useful from two perspectives: first, it provides a unified model for static and dynamic

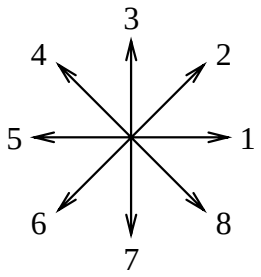


Figure 2. Directions

image data; second, it provides an easy way to integrate individual motion and dynamic relationships.

A *spatiotemporal relationship* (or st-relationship) is a spatial relationship that holds between two objects over a specified interval. We denote st-relationships by 4-tuples of the form (δ, χ, ψ, I), where the first three components are the standard components of a 2D-PIR spatial relationship, and I is a time interval over which the relationship holds. We then model the dynamic relationship between two objects as a sequence of st-relationships over successive intervals during a scene. That is, we use a sequence of the form:

$$[(\delta_1, \chi_1, \psi_1, I_1),\ (\delta_2, \chi_2, \psi_2, I_2),\ \ldots,\ (\delta_n, \chi_n, \psi_n, I_n)]$$

where the δ_i are topological relationships, the χ_i and ψ_i are interval projection relationships and the I_i are time intervals such that $finish(I_i) = start(I_{i+1})$ for $i \in \{1..n-1\}$. We use the terminology TS-2D-PIR for these temporal sequences of 2D-PIRs. For two objects A and B, the expression $R_{st}(A, B)$ denotes TS-2D-PIR describing their spatiotemporal relationship.

A moving object model for a scene is defined in a similar manner to the 2D-PIR model for static images, and is an integration of tracks and TS-2D-PIRs. A scene is represented by a graph, where each node of the graph contains an object descriptor (the object's name and its track) and each edge is labeled with the TS-2D-PIR for the objects in its end-point nodes.

Definition 1. A *scene graph* is a connected labeled digraph $S(M, R)$ where M is a finite non-empty set of object descriptors for the objects in the scene and R is a set of edges labeled by a sequence of spatiotemporal relationships (st-relationships) between pairs of objects.

Note that, in order to simplify our diagrams, we show the track for each object as a label on a circular edge leading from the object's node to itself, rather than trying to fit it into the node.

Figure 3(a) shows a scene containing a single moving object (A). The object descriptor for A 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 $i \in \{1..4\}$). Figure 3(b) is the (very simple) graph corresponding to the scene of Figure 3(a).

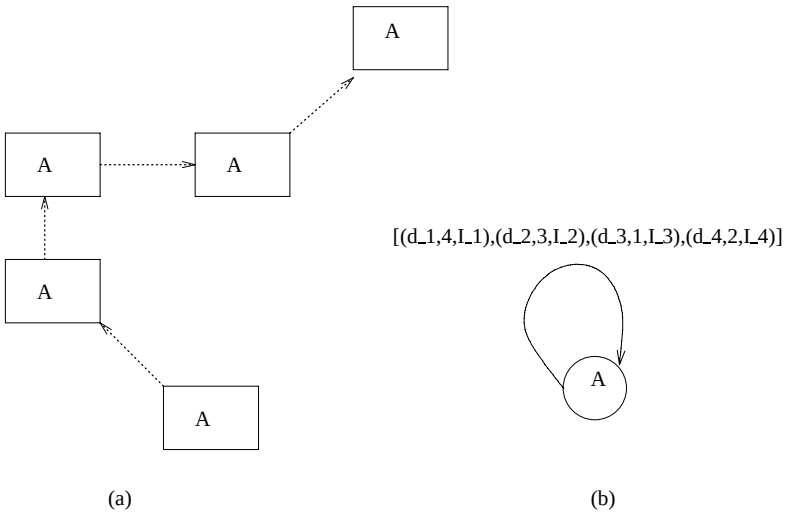


Figure 3. (a) A moving object, (b) Corresponding scene graph

Figure 4(a) shows a scene where objects A and B move towards each other until finally object A is inside object B. Figure 4(b) is the corresponding scene graph of Figure 4(a). The edge between A and B describes the changing spatial relationship between the two objects over time via a TS-2D-PIR. The first tuple $(dt, d, <, I_1)$ in the TS-2D-PIR indicates that during interval I_1: A and B are disjoint, they are at roughly the same location on the x-axis, and B is higher A on the y-axis (in other words, B is directly above A). The final tuple in the sequence (in, d, d, I_5) indicates that, near the end of the scene, object A is inside object B. The loops on nodes A and B give the tracks of the two objects. The track $[(d_a, 3, I_a)]$ indicates that A moves north for a distance d_a over the course of the scene; the track for B indicates that it moves south for a distance d_b over a slightly shorter time interval. Note that $I_a = (start(I_1), finish(I_5))$, while $I_b = (start(I_1), finish(I_4))$.

3. An Algebra for Moving Objects

In this section we define a collection of operations on scene graphs. These operations are analogous to the operations of relational algebra in providing a framework for expressing queries and the structure of query results.

In describing the operations of the algebra, we use the object-oriented *message passing* notation, $Obj\ op(args)$, where Obj is an object, op is an operation (message) and $args$ are the argument(s) of op. In this context, "objects" can be scenes, sets of scenes, moving

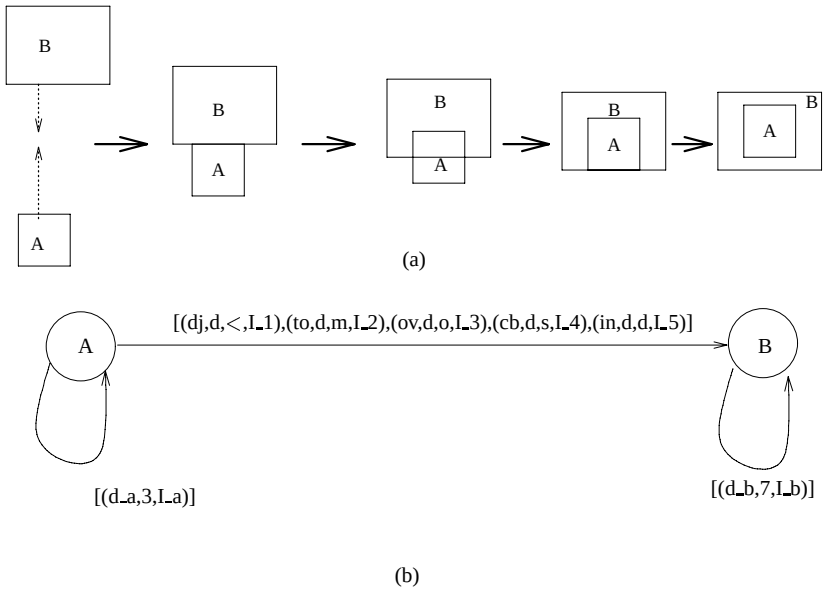


Figure 4. (a) Two moving objects (b) Corresponding scene graph

object descriptors, 2D-PIRs, TS-2D-PIRs, and so on. We use the term "database objects" to describe such objects to distinguish them from the moving objects within the video. We make heavy use of *operator overloading* in defining our operations, and also assume that many operators can be invoked by either a single database object or a set of database objects. In the latter case, the operator is applied to each object in the set.

In order to describe the semantics of moving object operations, we need some additional operations for extracting the components of scene graphs. If $S(M, R)$ is a scene graph as defined above, then $Objs(M)$ (or, by overloading, $Objs(S)$) is the set of object names in the scene.

In order to conform to our object-oriented notation, we re-define the $duration$ operator so that, for an interval $I = (t_s, t_f)$, $I\ duration$ returns the length $t_f - t_s$ of the interval.

The projection and overlay operations return a single scene. The selection operation returns a scene or a set of scenes. The extraction operations return either sequences of 2D-PIR relationships, sequences of (d, r) pairs, or sequences of time intervals. Finally, the spatial matching operation is a special type of string matching operation which returns a boolean value. Operations can be combined using standard function composition, wherever the output of the first operation is appropriate as an input to the second operation.

3.1. Projection (π)

The projection operations create a new scene based on an existing scene. The new scene is a "subset" of the original scene. Let $S(M, R)$ be a scene and $N = \{n_1, n_2, \ldots, n_k\}$ be the names of objects in S (i.e. $N \subseteq Objs(M)$).

Definition 2. Object projection: $S\ \pi(n_1, n_2, \ldots, n_k)$
The *object projection* of objects $n_1, n_2, \ldots, n_k$ from scene $S(M, R)$ is a scene $T(U, V)$ where $U \subseteq M$ and $V \subseteq R$. That is, T is a subgraph of S containing nodes and object descriptors for $n_1, n_2, \ldots, n_k$ and edges with associated TS-2D-PIR for each pair of objects in $Objs(U)$.

Figure 5 illustrates this operation. The projection of scene S on objects *b* and *C* creates the scene T containing only objects *b* and *C*.

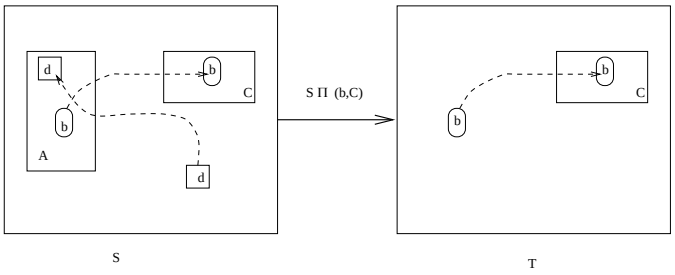


Figure 5. Projection: $S\ \pi(b, C)$

This operation is needed if a user wants to isolate a subset of the objects for a specific purpose, such as to observe their trajectories, or to observe how subsets of objects are related in a scene. Sometimes it is useful to isolate objects over only part of a scene. To do this, the following projection operation, called *spatiotemporal projection*, is provided.

Definition 3. Spatiotemporal projection: $S\ \pi(n_1, n_2, \ldots, n_k, I)$
The *spatiotemporal projection* of a scene $S(M, R)$ is an object projection of S on objects $n_1, n_2, \ldots, n_k$ over interval I. The result of this projection is a new scene containing objects $n_1, n_2, \ldots, n_k$ and their relationships during interval I. Formally, $S\ \pi(n_1, n_2, \ldots, n_k, I)$ is a connected labelled digraph $T(U, V)$ where U is the set of object descriptors for $\{n_1, n_2, \ldots, n_k\}$ and V is set of spatiotemporal relationships among $\{n_1, n_2, \ldots, n_k\}$ over interval I.

Figure 6 illustrates a spatiotemporal projection of the scene S on objects *b* and *d*.

Note that for a scene $S(M, R)$, $S\ \pi(Objs(M), I)$ yields a segment of the scene (i.e. a simple temporal projection); we use the notation $S\ \pi(I)$ to indicate this.

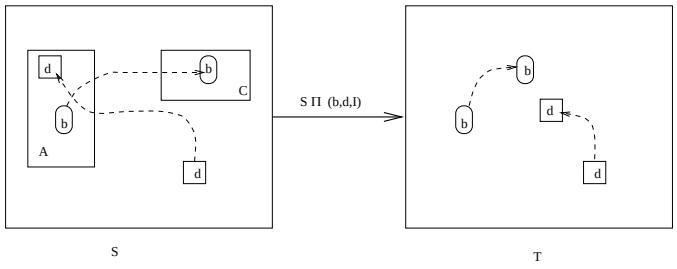


Figure 6. Spatiotemporal projection: $S\ \pi(b, d, I)$

3.2. Selection (σ)

The *selection* operation finds a set of scenes that satisfy a given predicate from a database of scenes. Let S_s and T_s be variables denoting sets of scenes and P be a variable denoting a predicate on some kind of database object.

Definition 4. Selection: $S_s : S\sigma(P)$
The *selection operation* on a scene collection S_s under predicate P is a set of scenes T_s such that $T_s = \{S \in S_s | P(S)\}$ and $\nexists X \in (S_s - T_s) | P(X)$. The S variable ranges over the set S_s, and will often be used as (part of) an argument to the predicate P. The ":" is used to bind the iteration variable (S) to the scene set (S_s).

Predicates used in selection can include such operations as: testing whether particular objects appear in a scene and similarity comparisons based on trajectories or spatiotemporal relationships (both the spatial part and temporal part). We describe some of the more useful predicates in section 3.6.

3.3. Overlay (Ω)

The overlay operation combines two different scenes to form a new scene. Let $S_1(M_1, R_1)$ and $S_2(M_2, R_2)$ be two scenes and τ be a time delay such that $\tau < duration(I_{S_1})$.

Definition 5. Overlay: $S_1\ \Omega(S_2, \tau)$
The *overlay* of scene S_2 on scene S_1 is a new scene $S(M, R)$ where $Objs(M) = Objs(M_1) \cup Objs(M_2)$ and R is a set of relationships among $Objs(M)$. The objects from S_2 do not appear in the result scene until time interval τ has elapsed (i.e. all interval values in S_2 are offset by τ).

Note that τ is used to synchronise between S_1 and S_2. If $\tau = 0$ then S_1 and S_2 start at the same time and we used the shorthand $S_1\ \Omega(S_2)$ to denote their overlay.

Overlay is a powerful operation that can be used to layer scenes to build up a complex scenes from existing ones. It can also be used effectively in conjunction with projection to

create a variety of scene effects. This is particularly useful in animation or video systems. Figure 7 illustrates the overlay operation.

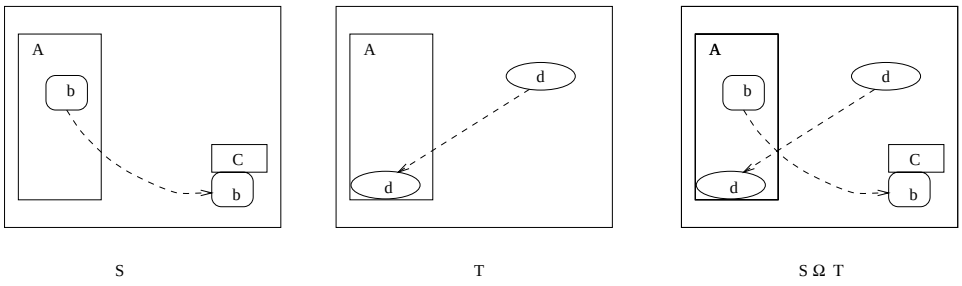


Figure 7. Overlay: $S\ \Omega(T)$

3.4. Spatial extraction (γ)

The *spatial extraction* operations project spatial information from a scene graph, filtering out associated temporal information. There are two variants: one produces a sequence of (d, r) pairs showing the *trajectory* of a single object; the other produces a sequence of 2D-PIR relationships showing the spatial relationships between two objects over the course of a scene.

Definition 6. Trajectory extraction: $S\ \gamma(A)$
If S is a scene containing an object descriptor $(A, [(d_1, r_1, I_1), (d_2, r_2, I_2), \ldots, (d_n, r_n, I_n)])$, then the *trajectory extraction* of object A is the sequence $[(d_1, r_1), (d_2, r_2), \ldots, (d_n, r_n)]$.

Definition 7. 2D-PIR extraction: $S\ \gamma(A, B)$
If S is a scene containing $R_{st}(A, B) = [(p_1, I_1), (p_2, I_2), \ldots, (p_n, I_n)]$, then $S\ \gamma(A, B)$ is $[p_1, p_2, \ldots, p_n]$ (i.e. the sequence of spatiotemporal relationships between A and B).

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.

Consider, for example, applying spatial extraction to the scene in figure 4. $S\ \gamma(A, B)$ returns the 2D-PIR sequence $[(dt, d, <), (to, d, m), (ov, d, o), (cb, d, s), (in, d, d)]$, that is, the label for the edge between objects A and B with the interval component removed from each list element. $S\ \gamma(A)$ returns $[(d_a, 3)]$, the trajectory for A.

3.5. Temporal extraction (ω)

The *temporal extraction* operation gives the time interval component of a spatiotemporal relationship between objects. In other words, it allows us to determine over what period a particular sequence of 2D-PIR spatial relationships holds.

Let A and B be objects in scene S and $[p_1, p_2, \ldots, p_n]$ be a sequence of 2D-PIR spatial relationships:

Definition 8. Temporal extraction: $S\ \omega(A, B, [p_1, p_2, \ldots, p_n])$
The *temporal extraction* of a scene S is an interval $I = I_1 + I_2 + \ldots + I_n$ such that there is a subsequence $[(p_1, I_1), (p_2, I_2), \ldots, (p_n, I_n)]$ in $P_{st}(A, B)$ in S.

If there is no interval satisfying the sequence of spatial relationships, then the distinguished interval value $Never$ is returned. 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. Any $Never$ intervals will be excluded from this set.

Note that the sequence of spatial relationships can be specified using a shorthand notation that will be described in Section 3.7. If R is omitted (i.e. $S\ \omega(A, B)$), the operation will return the interval during which both A and B are present in the scene. If only a single object is specified (i.e. $S\ \omega(A)$), the result will be the time interval during which A is present in the scene.

Consider, for example, applying temporal extraction to the scene in Figure 4.
$S\ \omega(A, B, [(to, d, m), (ov, d, o)])$ returns the interval $I_2 + I_3$. $S\ \omega(A, B)$ returns I_S since both objects appear throughout the scene.

3.6. Predicates

Because video scenes are complex entities, there are many possible tests that we might wish to make on their contents. In this subsection, we present a number of the more useful simple predicates. In the next subsection, we consider the more complex test operation of spatial matching.

Testing for the existence of objects in a scene is handled by the $contains$ predicate. Let $N = \{n_1, n_2, \ldots, n_k\}$ be object names and S be a scene. Then $S\ contains\ N$ is true iff $N \subseteq Objs(S)$. For a single object A, $S\ has(A)$ is shorthand for $S\ contains(\{A\})$. The $contains$ operation is also defined on sets of scenes and returns $true$ if any of the scenes in the set contains the specified objects.

We use Allen's interval-based relationships [4] to construct predicates involving temporal relationships between events. For example, in the query "find a scene where A enters B while C passes B", a temporal operation *during* is needed to test the temporal relationships between "enters" and "passes".

Possibly the most interesting test operation for video data is similarity scene comparison (ς). This allows us to pose queries such as: "Find scenes that contain an object whose trajectory is similar to the trajectory drawn in a window". Similarity measures normally return real number values; these can be converted into boolean values for use in test predicates by applying a threshold. If $D(O_1, O_2)$ is a similarity (distance) measure on two objects O_1 and O_2 and ϑ is a threshold, then we can define $\varsigma(O_1, O_2) = D(O_1, O_2) < \vartheta$. We discuss in detail how to measure similarity for trajectories, st-relationships and scenes in section 5.3. In our later use of the ς operator, we assume that the similarity is evaluated in the context of a globally specified threshold.

3.7. Spatial matching (η)

The spatial matching operation is used to compare sequences of 2D-PIR spatial relationships.

Definition 9. Spatial match: $L\ \eta(P)$
Let L be a 2D-PIR sequence of the form $[(\delta_1, \chi_1, \psi_1), (\delta_2, \chi_2, \psi_2), \ldots, (\delta_n, \chi_n, \psi_n)]$, let P be a 2D-PIR pattern (see below) and let L^* be the set of all subsequences of L. $L\ \eta(P)$ is *true* if $(\exists L_1 \in L^* | (Matches(L_1, P))$ and *false* otherwise.

A *2D-PIR pattern* can be:

- a normal 2D-PIR sequence of the form $[(\delta_1, \chi_1, \psi_1), (\delta_2, \chi_2, \psi_2), \ldots, (\delta_n, \chi_n, \psi_n)]$,
- it can have any of the δ_i, χ_i or ψ_i replaced by "?" (the "don't care" symbol)
- or it can have ".." (the "any sequence" symbol) interposed between any pair of 2D-PIR triples.

If L is a 2D-PIR sequence and P is a 2D-PIR pattern, then $Match(L, P)$ is defined as:

- a δ_i, χ_i or ψ_i value in P matches the same value in the corresponding position in L
- a "?" in P matches any value in the corresponding position in L
- a ".." in P matches any 2D-PIR sequence in L; if a ".." can match more than one sequence, the shortest is chosen.

We illustrate the use of 2D-PIR patterns on some examples. The event "object A enters object B from the left" can be represented by the pattern: $[(dt, <, ?), (to, m, d), (ov, o, d), (ic, d, d)]$. We require that A is initially to the left ($<$ on the x-axis) of B, but we don't care about its initial vertical position. The event "object A enters object B" can be captured by the pattern: $[(dt, ?, ?), (to, m, d), (ov, o, d), (ic, d, d)]$. We don't care where B starts, as long as it is outside A. We can abbreviate this further by noting that if A starts outside and finishes inside, it must have touched and overlapped B. Thus an equivalent pattern would be: $[(dt, ?, ?)..(ic, d, d)]$. The event "object A enters B and then leaves again" can be represented by the pattern: $[(dt, ?, ?)..(ic, d, d)..(dt, ?, ?)]$. In other words, A starts outside B, eventually appears inside it, and finally appears outside B again.

3.8. Complexity of operations

The complexity of the various operations depends primarily which parts of the database they act on. For example, the *project* and *extract* operators act on individual scenes, so the cost will be related to the complexity of the scene not to the total size of the database. In fact, the only operation whose complexity is determined by the size of the database is *select*. If not supported by indexing (which is the current state of our system), *select* requires time proportional to the number of scenes stored in the database.

The *project* and *spatial extract* operations take object identifiers and produce information associated with these objects. The complexity of this depends on how easy it is to find the objects within the scene graph. If we assume that scene graphs are implemented efficiently (e.g. with hashing on object names), then the complexity of locating the objects and their associated information will be $O(1)$.

The *temporal extract* operation takes an object pair and a 2D-PIR pattern and matches the pattern with the TS-2D-PIR for the object pair. As above, we assume that it is simple to find the TS-2D-PIR, so the major cost of this operation comes from the matching operation. If the length of the TS-2D-PIR is L_1 and the length of the pattern is L_2, then the cost of matching will be $O(L_1 L_2)$.

The costs of topological and interval comparisons are trivial, since these relationships involve very simple data structures, and are generally pre-computed and stored in the database. The costs of scene and trajectory similarity comparison are considerably higher, and will be discussed in Section 5.

4. Applications of Moving Object Algebra

In this Section we demonstrate the power of our moving object algebra by showing how it can be used to answer complex spatiotemporal queries for a number of specific applications. We assume that we have available an unlimited supply of variable names to identify objects, scenes and sets of scenes (databases), and we make use of an **is** operator to associate names to expressions of arbitrary types. We also assume that standard quantification operations ($\exists$, $\forall$) are available as predicates with implicit iteration that allows them to bind the values of the quantified variables to appropriate database objects, one at a time.

4.1. Animal Movement Querying System

The Animal Movement Querying System tracks the movement of animals using a Global Positioning System (GPS) device attached to each animal. The system could be used, for example, 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, birds, etc.).

Our data model is ideally suited to this kind of application because all of the objects in the system are readily identified and easy to track automatically. The GPS could be used to obtain time series data on the moving objects; for example, hourly reports on the position of each animal. The area of interest can be divided into a set of regions (static objects) through which animals can move. From such data, it is straightforward to automatically compute the scene graphs (tracks and st-relationships). Note that we assume that the map regions are oriented such that the x-axis interval relationships "$<$" and "$>$" correspond to *west* and *east* and the y-axis interval relationships "$<$" and "$>$" correspond to *north* and *south* respectively.

Using this system, biologists might wish to pose queries such as "Show the migration path of a particular animal" or "Find animals that stay in a particular region longer than a certain time" (an indication that the animals might breed in this region).

Assume that the animal migration is supplied as a sequence of high-magnification satellite photographs of the region under study and that the GPS information allows us to extract

tracks automatically from this video data. The data is recorded as a collection of spatiotemporal data in a database called $MigrationDB$. This database would contain scene-graphs for each of the periods under study (the scenes in this video data), along with the actual video sequences showing animal movement.

We illustrate how five representative queries can be answered on this database using moving object algebra. It is also possible (and more appropriate) to specify some of these queries via a sketch-based query facility. Such a facility is available in our system, and we discuss this in the video retrieval example.

Q1. ***Give me the migration path of a specific animal named*** $Elephant$***.***
The answer to this query is a trajectory which can be displayed graphically. If we know the name of the particular scene showing the migration of this animal (say $ElephantMigration$), then the query can be expressed as follows:

$$ElephantMigration\ \gamma(Elephant)$$

Recall that the γ operator returns precisely the trajectory information required to answer this query.

If we do not know the name of the scene, then we must use selection to first find the scene and then extract the animal's trajectory. In this case the query can be expressed as:

$$(MigrationDB : M\ \sigma(M\ has(Elephant)))\ \gamma(Elephant)$$

Note the use of the iteration variable M, which will be bound successively to each scene in the database. We assume here that there is only one matching scene that shows the migration. If there were multiple scenes containing $Elephant$, the γ operator would extract trajectories for each of them.

Q2. ***How long is animal*** A ***inside region*** R***?***
The answer to this query is a time value, the length of the interval over which the relationship $A\ inside\ R$ holds. The query can be expressed as:

$$AInsideR\ \textbf{is}\ MigrationDB : M\ \sigma(M\ \omega(A, R, [(ic, d, d)]) \neq Never)$$

$$(AInsideR\ \omega(A, R, [(ic, d, d)]))\ duration$$

First, we need to find the scene where animal A is inside region R (once again, we assume that σ will give us one single scene rather than a set of scenes). Note that there is no operation to directly test whether two objects have a specified relationship at some stage during a scene. We achieve this by first attempting to extract the temporal interval (via ω) over which the relationship holds, and then testing that the extraction produced a real interval rather than $Never$. We associate the matching scene with the variable $AInsideR$. Next, we extract the interval during which animal A is inside R and compute its duration.

Q3. ***Find scenes where animal*** A ***enters region*** R ***from the west side and leaves from the north side.***
The result of this query is a set of scenes where each scene contains a particular kind

of movement. We describe the movement via a 2D-PIR pattern[8] and use the trajectory matching operator (η) to determine which scenes contain such movement.

$$EnterWest \textbf{ is } [(dt, <, ?), (to, m, d), (ov, o, d), (ic, d, d)]$$

$$ExitNorth \textbf{ is } [(ic, d, d), (ov, d, oi), (to, d, mi), (dt, ?, >)]$$

$$MigrationDB : M\ \sigma((M\ \gamma(A, R))\ \eta(EnterWest + ExitNorth))$$

First we define the spatiotemporal sequence that represents the event of "entering a region from the left (west)" and "leaving by the top (north)". We then use that sequence with the η operator to match sequences relating animal A and region R (which are extracted using γ). Note that the 2D-PIR pattern is reasonably complex to specify; it would be better for users to be able to draw the movement. This facility is supported by our prototype system, and we give examples of its use in Section 4.3.

Q4. ***Find scenes where some animal passes through region*** R**.**
This query is similar to Q3, except that the direction from which A enters R and the direction A exits R are ignored. The result is a list of scenes from $MigrationDB$ where there is at least one object that satisfies the spatiotemporal relationship "passes through". The query is expressed as:

$$MigrationDB : M\ \sigma(\exists A \mid (M\ \gamma(A, R))\ \eta([(dt, ?, ?)..(ic, d, d)..(dt, ?, ?)]))$$

For each animal, the query first extracts the 2D-PIR sequence (via γ) giving the spatial relationship between the animal and region R. It then compares this (using η) to the 2D-PIR pattern describing the "passes through" relation (essentially "outside, then inside, then outside again"). If any of the animals tested satisfy the condition, then the existential quantifier returns a *true* result to trigger the selection of that scene.

4.2. Animation Database

The following examples show how we can create a new scene by using the *object projection* and *overlay* operations on a set of existing scenes. Let us imagine we have a database of animated scenes. In the database we have a scene ($CatAndMouse$) containing several mice and two cats, where one of the cats is catching a mouse. We also have a scene (called $DogsAndPond$) that contains several dogs running towards a pond and then drinking from the pond. We want to create a new scene that tells the story "a cat catches a mouse while a dog runs to a pond and drinks water from it".

In the first scene we are interested only in a cat called tom which is catching a mouse called $jerry$. The following expression extracts the portion of the scene that contains the "cat catches mouse" part of our story:

$$MouseChase \textbf{ is } CatAndMouse\ \pi(tom, jerry)$$

In the second clip we are interested only in a particular dog called $bulldog$ that is running towards the $pond$ and then drinking. Here is the expression that extracts the scene containing "a dog running to a pond and then drinking from it":

$$DogRunDrink \textbf{ is } DogsAndPond\ \pi(bulldog, pond)$$

To create the new scene, we simply use the overlay operator on the two extracted scenes:

$$MouseChase\ \Omega(DogRunDrink)$$

Now, suppose that we are no longer interested in the dog running, and only want to see it drinking for the later part of the chase. If both clips were 60 seconds long, and we knew that the drinking sequence occupied the last 30 seconds of the $DogsAndPond$ scene, we could produce the new scene via:

$$DogDrinking \textbf{ is } DogsAndPond\ \pi(bulldog, pond, (30, 60))$$

$$MouseChase\ \Omega(DogDrinking, 30)$$

4.3. Video Retrieval System

This example shows the use, in a video retrieval application, of the scene similarity comparison operator (ς) mentioned in Section 3.6. This example also demonstrates the necessity of a sketch-based query facility for moving objects. The example queries assume that we have a collection of video scenes and their corresponding scene graphs stored in a database called $SceneDB$.

Q1. ***Find scenes that contain an object whose movement is similar to the movement of object*** X ***in scene*** $S5$**.**
This query can be answered by comparing the similarity of the trajectory of object X in $S5$ with the trajectory of any object from the scenes stored in $SceneDB$. The query can be expressed as:

$$SceneDB : S\ \sigma(S\ \varsigma(S5\ \gamma(X)))$$

This works by first extracting X's trajectory from $S5$ (γ), then applying the similarity test (ς) to all of the scenes in $SceneDB$. Note that the test performs a (thresholded) trajectory similarity comparison between object X and each of the objects in each of the scenes. If any of the objects in some scene S has a trajectory that is similar to X's trajectory, then scene S will be retrieved.

Q2. ***Find scenes that contain a person moving like the trajectory drawn in a window.***
The trajectory sketched in the window is first converted into a sequence of (d, r) pairs called T. To answer the query, we need to compare this trajectory to the spatial projection of the person objects[9] in each of the scenes in the database. The query can be specified as follows:

$$SceneDB : S\ \sigma(\exists P \in Objs(S) \mid (S\ \gamma(P))\ \varsigma(T))$$

The quantifier ($\exists$) involves an implicit iteration over the objects in each scene. For each object, we extract its trajectory (γ) and then compare (ς) it with the trajectory (T) sketched in the window. A scene S will be retrieved if it contains an object whose trajectory is similar to T. Once again, we assume that the trajectory similarity match is conducted using a global threshold value.

Q3. ***Find scenes that show a racing car X overtaking a group of other cars as shown in Figure 8.***

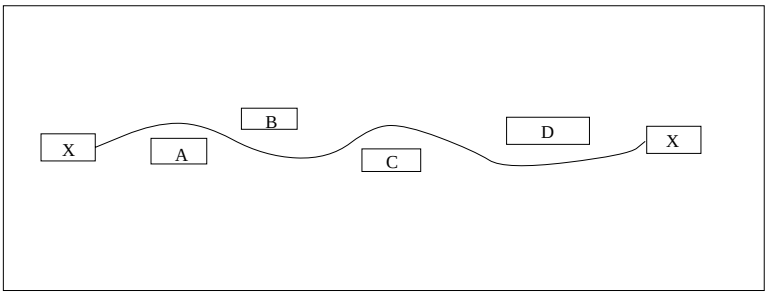


Figure 8. Car X overtakes A, B, C and D

Consider the task of devising a set of 2D-PIR patterns to describe the st-relationships between car X and all of the other cars. It would simply be too tedious to make the query feasible. However, given the sketch in the figure, it would be quite straightforward to automatically produce a scene graph to model this situation. This scene graph (which we denote by the name $Sketch$) forms the basis for the query. Answering the query is then simply a matter of using scene graph similarity matching to find the appropriate scenes in the video:

$$SceneDB : S\ \sigma(S\ \eta(Sketch))$$

This approach also extends naturally to the kind of similarity-by-example retrieval that is currently available for static images in systems such as QBIC [11]. If we already have access to a particular scene $MyScene$ in our $SceneDB$, we can request the system to find similar scenes simply by performing similarity matching in the same manner as above:

$$SceneDB : S\ \sigma(S\ \eta(MyScene))$$

We discuss the process of scene similarity matching in more detail in Section 5.

Q4. ***Find a scene in which object X enters region R while object Y departs from the same region.***

This query is different from the previous queries since we need to consider the temporal relation between two events ("X entering R" and "Y departing from R"). Another way of looking at this is that the time interval when "X enters R" must overlap with the interval when "Y leaves R". This can be expressed as follows:

$EnterInto$ **is** $[(dt, ?, ?), (to, ?, ?), (ov, ?, ?), (ic, ?, ?)]$

$ExitFrom$ **is** $[(ic, ?, ?), (ov, ?, ?), (to, ?, ?), (dt, ?, ?)]$

$SceneDB : S\ \sigma((S\ \omega(X, R, EnterInto))\ o\ (S\ \omega(Y, R, ExitFrom)))$

The first step in answering this query is to determine the intervals in each scene S that correspond to "X enters R" and "Y leaves R". This is accomplished via the temporal extraction operator (ω). Note that if either event does *not* occur in a scene, the value $Never$ will be returned. The intervals are then checked for any overlap via the o interval operator. If either of the intervals is $Never$ or if there is no overlap, then the test will fail and the scene will not be selected.

5. Scene Retrieval

In the previous section, we showed that one of the most useful kinds of video retrieval operations was similarity comparison against either a sketch or an existing scene. Since scenes are represented by *scene graphs* in our system, this similarity comparison will actually be carried out by measuring the similarity between scene graphs. The process of answering a query begins with a query scene graph (for a query based on a sketch we must first construct the scene graph; for a query against an existing video scene, we assume that the scene graph is already stored in the database). To produce the answer to the query, the system computes the similarity between the query graph and the stored graph for each scene in the database, and accepts those queries for which the similarity exceeds a globally specified threshold. In this section, we describe the process of scene graph construction and define the scene graph similarity measure.

5.1. Scene graph construction

The input to the scene graph construction process is a video scene that has been pre-processed to identify all objects, note their pointsets and track their movement (we noted above that this would require manual assistance). The output from this process is, naturally, a scene graph (the structure of which was described in Section 2).

Note that the scene graph stores only one of the two possible TS-2D-PIRs for each pair of objects (i.e. it stores either $R_{st}(A, B)$ or $R_{st}(B, A)$). Which of these is actually stored is determined by using an ordering on the objects (for example, order based on unique identifying name or based on left-to-right, bottom-to-top position in the first image of a scene); the ordering is used on line (12) of the algorithm below. The scene graph only contains $R_{st}(X, Y)$ if $X < Y$ according to this ordering. The following algorithm describes the scene graph construction process.

Algorithm: scene graph construction
Input: A pre-processed video scene (objects are identified,
object positions, pointsets and projections are available for each frame)

Output: A scene graph $S(M, R)$ representing the scene
begin
(1) M = {}; R = {};
(2) **for** each frame F **do**
(3) t = time of frame F relative to start of scene
(4) **for** each object X in F **do**
(5) **if** X is not already in M **then**
(6) include new object descriptor $(X, [])$ in M
(7) **else**
(8) compute (d, r, I) from $current$ and $previous$ positions
(9) append (d, r, I) to track for X
(10) **endif**
(11) save $current$ position as $previous$
(12) **for** each object Y greater than X in F **do**
(13) compute $current\ R_{st}(X, Y)$
(14) **if** there is no edge for (X, Y) in R **then**
(15) include new edge $(X, Y, [])$ in R
(16) save $current\ R_{st}(X, Y)$ as $existing\ R_{st}(X, Y)$
(17) set time for $startOfR_{st}(X, Y)$ to t
(18) **else**
(19) **if** F is the last frame **or**
$current\ R_{st}(X, Y)$ is different to $existing\ R_{st}(X, Y)$ **then**
(20) compute (δ, χ, ψ, I) based on $existing\ R_{st}(X, Y)$
and interval from $startOfR_{st}(X, Y)$ until t
(21) append (δ, χ, ψ, I) to TS-2D-PIR for edge (X, Y)
(22) save $current\ R_{st}(X, Y)$ as $existing\ R_{st}(X, Y)$
(23) set time for $startOfR_{st}(X, Y)$ to t
(24) **endif**
(25) **endif**
(26) **endfor**
(27) **endfor**
(28) **endfor**
end.

This algorithm has two potential problems: it may have to deal with a very large number of frames for even a relatively short scene, and it may produce long trajectories and st-relationships (if object movement is complex). We propose the notion of *key frames* as a way of addressing these two problems.

The idea behind *key frames* is that the algorithm does not need to examine every frame in the scene. This reduces the number of frame computations (i.e. iterations of the loop at statement (2)), and reduces the length of trajectories and st-relationships. Of course, this comes at a cost: scenes may no longer be represented as accurately as previously, since some object motion and hence some elements in the st-relationships may be omitted. This would certainly be true if key frames were implemented simply by using every k^{th} frame in the scene, but our approach uses various rules to do better than this.

The first and last frames of a scene are always chosen as key frames. An initial set of key frames can be built by simply choosing every k^{th} frame (where k is a small constant, such as 30). Other frames can be added to this set based on "significant changes" in the scene, as determined by various heuristics. For example, frames in which moving objects first appear in or disappear from the scene are chosen as key frames. If an object appears in frame i and disappears in frame $i+n$, then intermediate frames such as $i+\frac{n}{2}$ and $i+\frac{n}{4}$, etc. are also chosen. Once a set of key frames is available, it is used in statement (2) of the above algorithm instead of iterating over the entire scene.

5.2. Topological and interval relationship neighbourhood graphs

Topological and interval relationship neighbourhood graphs provide a way of describing *changes* in the spatial and topological relationships between moving objects. They form the basis of mechanisms to reason about these changes; for example, to infer intermediate spatial relationships if we know only the starting and finishing relationships. They also provide the basis for the distance measures used in scene similarity matching.

5.2.1. Topological relationship neighbourhood graph Consider two moving objects A and B which are *disjoint* (i.e. $A\ dt(B)$). If we monitor the topological relationship between these two objects over time, it must either remain as *disjoint* (if the objects never approach each other) or become *touches* (if they move together). Note that to is the only possible immediate successor relationship to dt if A and B move "normally" (i.e. they do not jump through space).

Definition 10. Topological neighbours
Topological relationships T_1 and T_2 between two spatial objects are *neighbours* if T_1 can be directly transformed into T_2 by continuously moving the objects.

In order to use the notion of topological neighbours, we need to construct a *neighbourhood graph*, that shows the neighbourhood property over all of the topological relationships. We derive this graph by considering three scenarios for two objects A and B, as shown in Figure 9.

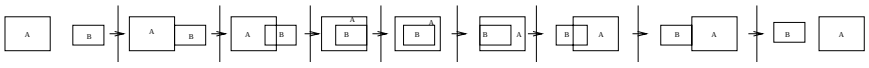
Each scenario begins with $A\ dt(B)$ and the objects move towards each other. We observe the following sequences of topological relationships for these cases:

1. *disjoint, touches, overlaps, covers, contains, covers, overlaps, touches, disjoint*

2. *disjoint, touches, overlaps, equals, overlaps, touches, disjoint*

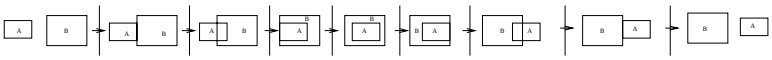
3. *disjoint, touches, overlaps, covers, inside, coveredBy, overlaps, touches, disjoint*

Whenever two relationships are adjacent in any of these sequences, they are topological neighbours. Note that we do not need to consider any other cases, such as two objects passing and overlapping, since they lead to a shorter sequence of topological relationships, with no additional neighbours.



Case 1: A is larger than B.

Case 2: A is the same size as B.



Case 3: A is smaller than B.

Figure 9. Scenarios for topological relationship neighbours

Using this information, we can build a graph where the nodes are topological relationships and an edge between relationships T_1 and T_2 indicates that they are neighbours. This is the topological relationship neighbourhood graph and is shown in Figure 10.

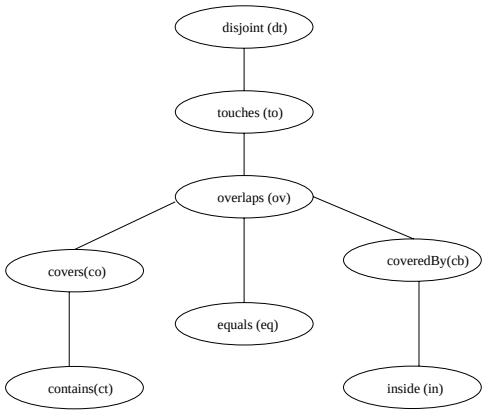


Figure 10. Topological relationship neighbourhood graph

5.2.2. Interval relationship neighbourhood graph In the previous section, we considered the changing topological relationships between two moving objects. The same kind of analysis can be applied to the interval relationships on the projections of the objects on the axes. For example, if an object A starts left of ($<$) object B and the two objects move towards each other, the intervals will eventually meet, overlap, and A might ultimately be to the right of B ($>$).

Definition 11. Interval neighbours
Interval relationships P_1 and P_2 between two spatial objects are *neighbours* if P_1 can be directly transformed into P_2 by continuously moving the objects.

We derive the interval relationship neighbourhood graph following the approach for *B-neighbours* in Freksa [12]. Figure 11 shows the derived graph.

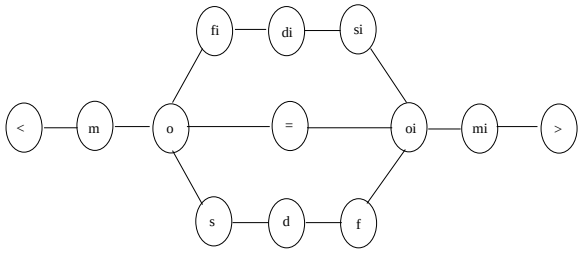


Figure 11. Interval relationship neighbourhood graph

5.3. Similarity functions for scene matching

In order to perform scene similarity matching, we need to have available a measure of the "distance" between pairs of scenes. This distance measure should be based on the stored information about scenes, that is, on the contents of their scene graphs. Since scene graphs contain two different kinds of elements (object descriptors and st-relationships), our approach is to develop distance measures for each element type and then combine the distances for corresponding element pairs to form the overall scene distance measure.

In developing these distance measures, we do not require a notion of "object similarity". We either use equality against an object specified in the query, or, if we are using quantifiers, we simply bind the variable to all available objects. For example, if we take a query such as "find scenes containing an object that moves like object A in scene S", we are not interested in the identity of any matching object, but simply on whether the object has a track that is similar to A's track. In general, object identity tells us which distance measures we need to compute and the distance computations themselves are independent of the identity of the objects involved.

5.3.1. Trajectory similarity In considering whether two objects move in a similar manner, the relevant information that we have available in the scene graph is the track of each object. Recall that tracks contain interval information which indicates the period over which a movement occurs (relative to the beginning of the scene). In comparing movement similarity, we are not generally concerned with *when* the movement occurs but rather on whether there is a similar sequence of movements in the track. Thus, we base the object movement similarity measure on trajectories rather than tracks.

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 pairs,

i.e., $[(d_1, r_1), (d_2, r_2), \ldots, (d_n, r_n)]$, the distance between two trajectories is determined by the distances among the corresponding elements of the trajectories.

Let r_a and r_b be two directions (refer to figure 2 for the eight directions). The distance between r_a and r_b, $D(r_a, r_b)$, is defined by a minimum angle between r_a and r_b. For example $D(1, 2) = 45$ and $D(1, 8) = 45$ and $D(1, 4) = 135$.

We wish to convert this distance measure into a normalised similarity measure where a value of 0 means "totally dissimilar" and 100 indicates "identical". Let us denote the maximum distance between two directions by $MaxDir$ (if we measure angles in degrees, as above, clearly $MaxDir = 180$). The similarity (ς) between two directions r_a and r_b is thus defined as:

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

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

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

Let $T_1 = [(d_{p1}, r_{p1}), \ldots, (d_{pn}, r_{pn})]$ and $T_2 = [(d_{t1}, r_{t1}), \ldots, (d_{tm}, r_{tm})]$ be two trajectories, and let $head(T)$ and $last(T)$ denote the first and last elements of a trajectory T. Assume that $length(T_2) \geq length(T_1)$ and that T_2 contains k substrings $l_1, l_2, \ldots, l_k$ such that $length(l_i) \leq length(T_1)$. its last element.

We compute the similarity of the two sequences by considering all possible sequence "alignments". There are three cases to consider: first, T_1 $overlaps$ T_2 if $head(T_1)$ precedes $head(T_2)$ and $last(T_1)$ lies between $head(T_2)$ and $last(T_2)$; second, T_1 in T_2 if $head(T_1)$ coincides with or follows $head(T_2)$ and $last(T_1)$ precedes or coincides with $last(T_2)$; third, T_1 $overlapped\ by$ T_2 if $head(T_1)$ lies between $head(T_2)$ and $last(T_2)$ and $last(T_1)$ follows $last(T_2)$.

The similarity for each of these cases is computed slightly differently, depending on which elements of the trajectory sequences are actually considered in the computation. If $n = length(T_1)$ and $m = length(T_2)$, and k is the iteration variable, the three cases are computed as follows:

For T_1 $overlapping$ T_2:

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

For T_1 in T_2:

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

For T_1 $overlapped\ by$ T_2:

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

The similarity (ς) between two trajectories is then defined as:

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

An 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 $\varsigma(r_a, r_b)$.
One important property of this definition is elaborated by the following theorem:

THEOREM 1 *$\varsigma(T_1, T_2)$ is scale invariant.*

Proof: Let T_1 and T_2 be two trajectories where T_1 is created by scaling down T_2 (i.e. multiplying all distances in T_2 by a factor in $[0..1]$). Let T_1 be $[(d_{p1}, r_{x1}), \ldots, (d_{pn}, r_{xn})]$ and T_2 be $[(d_{t1}, r_{x1}), \ldots, (d_{tn}, r_{xn})]$. $\varsigma(T_1, T_2)$ will be maximum for 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 $length(T_1) = length(T_2)$.

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

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

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

since $d_{pi} < d_{ti}$ (T_1 is smaller in scale as compared to T_2)

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

which means $T_1 = T_2$. □

5.3.2. TS-2D-PIR similarity In the previous subsection, we described how to measure the similarity of the movements of two objects. Video data, however, typically contains multiple objects in a scene and so to measure scene similarity we also need to take account of the interactions among objects. To measure the similarity of two collections of moving objects, we need to measure the similarity between their corresponding TS-2D-PIRs. However, for the same reasons mentioned above for trajectory similarity, we need only to consider the spatial components of TS-2D-PIR (i.e. the 2D-PIR sequence).

The degree of similarity between 2D-PIR relationships, is dependent on the degree of similarity between components of 2D-PIRs. 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 for both

topological relationships (δ) and interval relationships (χ and ψ). These similarity metrics are based on distances within the neigbourhood graphs defined in Section 5.2; the details of the derivation of distance metrics table for 2D-PIRs is given in [23]. This measure has a minimum value of 0 (for identical 2D-PIRs) up to a maximum distance of 20.785. Based on distance measure for topological and interval relationships, the distance between two 2D-PIR relationships, p_1, p_2 is defined using the following Euclidean distance formula:

$$D(p_1, p_2) = \sqrt{(D(\delta_1, \delta_2))^2 + (D(\chi_1, \chi_2))^2 + (D(\psi_1, \psi_2))^2}$$

However, our objective is to measure similiarity between two pictures in term of ς. This implies deriving ς from the Euclidean distance formula. As before, we use a normalized measure with values in the range $[0..100]$ (0 means "totally dissimilar", 100 means "identical"). The similarity between two 2D-PIR relationships, p_1 and p_2 is defined using the formula:

$$\varsigma(p_1, p_2) = 100 \times \frac{20.785 - D(p_1, p_2)}{20.785}$$

Based on this, we can define a similarity measure for 2D-PIR sequences. Let

$$ST_1 = [(\delta_{1,1}, \chi_{1,1}, \psi_{1,1}), (\delta_{1,2}, \chi_{1,2}, \psi_{1,2}), \ldots, (\delta_{1,n}, \chi_{1,n}, \psi_{1,n})]$$

and

$$ST_2 = [(\delta_{2,1}, \chi_{2,1}, \psi_{2,1}), (\delta_{2,2}, \chi_{2,2}, \psi_{2,2}), \ldots, (\delta_{2,n}, \chi_{2,n}, \psi_{2,n})]$$

be the spatial components of st-relationships (i.e. 2D-PIR sequences). The similarity between ST_1 and any ST_2 is defined as:

$$\varsigma(ST_1, ST_2) = \frac{\sum_{i=1}^{n} \varsigma(p_{1,i}, p_{2,i})}{n}$$

5.3.3. 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 similarity between scenes (or, more strictly, between scene graphs).

Let $S_1(M_1, R_1)$ and $S_2(M_2, R_2)$ be two scene graphs. Let $N = Objs(M_1) \cap Objs(M_2)$ be the set of objects common to both scenes. Let $Ts_1 = \{S_1\ \gamma(X) | X \in N\}$ be the trajectories of all of the common objects in S_1, and $Ts_2 = \{S_2\ \gamma(X) | X \in N\}$ be the trajectories of these objects in S_2. Let $STs_1 = \{S_1\ \gamma(X, Y) | X, Y \in N \wedge X < Y\}$ and $STs_2 = \{S2\ \gamma(X, Y) | X, Y \in N \wedge X < Y\}$ be the sets of 2D-PIR sequences for the two scenes. The distance between S_1 and S_2 is then computed as a weighted sum of the trajectory distances and st-relationship distances on the common objects:

$$D(S_1, S_2) = w_1 \sum_{X \in N} D(S_1\gamma(X), S_2\gamma(X)) + w_2 \sum_{X,Y \in N} D(S_1\gamma(X, Y), S_2\gamma(X, Y))$$

where w_1 and w_2 are weights indicating the relative importance of the trajectory and spatial components.

The scene *similarity* measure is defined in terms of this distance measure, normalized into $[0..100]$ as before. In this case, however, the value of the maximum distance depends on the number of terms in the summation. Let us denote the number of common objects by N_n and the number of st-relationships by N_st, and denote the maximum scene distance in terms of these by $MaxScene_{N_n,N_st}$. Thus, the similarity between two scenes S_1 and S_2 is defined as:

$$\varsigma(S_1, S_2) \;=\; 100 \times \frac{MaxScene_{N_n,N_st} - D(S_1, S_2)}{MaxScene_{N_n,N_st}}$$

5.4. Complexity of scene matching

The computation of scene distance involves two summations over the set of common objects. Let us denote the size of this set as N, where N will typically be of the same order as the size of the object sets for each individual scene. Each component in the first summation (over the N objects) requires a trajectory distance calculation; the cost of this is proportional to the product of the lengths of the individual trajectories. The second summation is over all *pairs* of common objects in the scene (which requires $O(N^2)$ terms). Each component in this summation requires a 2D-PIR sequence distance calculation; the cost of this is proportional to the length of the st-relationship sequence in each scene. If T denotes the average trajectory length, and R is the average length of the st-relationship sequences, then the time taken for the similarity calculation is $O(NT^2 + N^2R)$.

6. A Prototype Video Retrieval System

The architecture for our prototype video retrieval system is shown in Figure 12. The system consists of seven major components: user interface, scene identification, query interpreter, scene graph builder, insertion engine, retrieval engine and scene database. Note that the query interpreter is essentially just a parser for text-based queries (like those in Section 4).

6.1. 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, plays the animation (to generate st-relationships between objects); the system then generates a scene graph and stores it in the database. The current prototype treats every moving object as having equal constant speed. Our model is, of course, capable of dealing with objects moving at different and non-uniform speeds, but this type of object movement is sufficient to enable us to conduct our experiments.

In creating a scene graph from a video source, the system interacts with a user to identify and track objects in the scene. This feature is not implemented in the current prototype, although most of the functionality is available in our static image retrieval system [24].

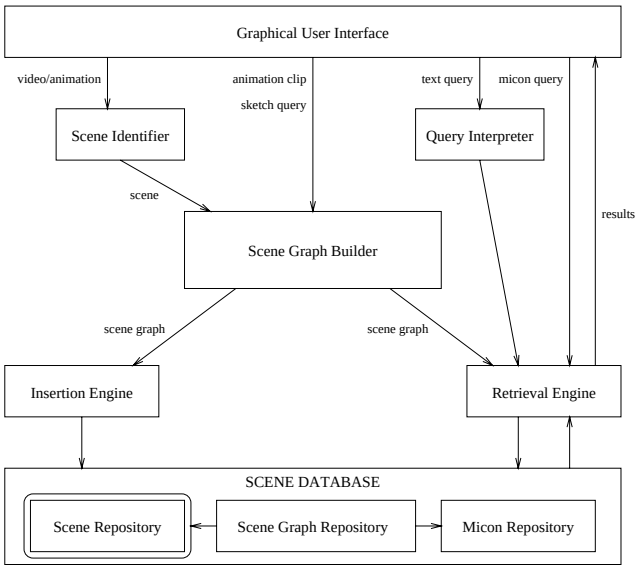


Figure 12. Architecture of Prototype Video Retrieval System

A user has two options in formulating queries: posing a query using a query language (text-based) or posing a query using a graphical query language (sketch-based). With the graphical query language, the user may pose either trajectory-similarity queries or scene-similarity queries. For trajectory-similarity queries, the trajectory is sketched in a window and the system finds scenes containing objects moving with a similar trajectory. For scene-similarity queries, either the entire scene is sketched (as if creating an animation) or a known scene is identified to be used as the basis for the similarity matching.

6.2. Scene identification

"Real world" video sources such as movies consist of a sequence of scenes. A scene (or clip) is a (typically short) sequence of video data shot for a continuous period using a single camera (some other video retrieval workers call this a *shot*). Each scene contains a number of (possibly) moving objects and consists of a sequence of frames (static images). Since it is not generally useful to return an entire movie as the answer to a query, our retrieval system deals with all video data at the scene level. In our system, scenes are accessed primarily by their representation as scene graphs through scene-similarity queries. However, a name may also be associated with a scene for use in retrieval (e.g. $CatAndMouse$ in Section 4.2).

One problem with treating video data as scenes is that the data is supplied as a continuous sequence of images, with no separation between scenes. One approach to scene separation is to use manual methods, but this approach is clearly not practical for large video databases. A second approach is to employ an existing method to automatically determine scene boundaries. Some successful techniques for this problem have been developed over the

last few years based on ideas such as adjacent frame colour similarity [28, 19, 25]. A good survey of these methods can be found in [7].

The current prototype assumes that scene identification has already been done, which is trivial for user-supplied animations where each animation corresponds to a single scene.

6.3. Scene graph builder

The major task of the *scene graph builder* is to transform a video or animation scene into a scene graph as described in Section 5.1. In building a scene graph from a sequence of frames, the graph builder is aided by an image processing module and the user in identifying objects in the scene. While this feature is not implemented in the current prototype, we provide a brief explanation here of how it can be achieved.

The image processing module will identify objects contained in the scene starting from the first frame. This identification process is a non-trivial problem and a fully automatic approach is not feasible using current technology, except in specific domains such as video surveillance. Our approach requires manual intervention in segmenting and naming objects in frames. The user is not required to segment every frame, since we only consider a subset of the frames (the *key frames*).

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 the TS-2D-PIR representation for the animation can be generated completely automatically.

6.4. Insertion engine

The *insertion engine* is responsible for storing videos and scene graphs in the file system. While it is possible to store entire videos (in a format such as MPEG), it is more likely that the system in future will simply store references (e.g. URLs) to the original video sources. The current prototype only stores user-generated animations, and these are actually stored as sequences of operations that can recreate the animation through the GUI.

Scene graphs are stored simply as binary data structures. In future versions of the system, we plan to incorporate indexing capabilities to enable faster retrieval and better filtering of scenes. One type of indexing that could be very simply implemented, would be to build signatures indicating which objects were present in each scene. This could be used to filter out irrelevant scenes and save on scene similarity computations for queries where we require a match on a specified set of objects.

6.5. Retrieval engine

Retrieval of scenes from a scene database can be performed via *trajectory-based*, *scene-based* or *text-based* methods. For all similarity-based queries, the system will make use of a threshold value that is either specified by the user before the query is run or is a default value chosen to yield “reasonable” precision and recall over a range of queries.

Trajectory-based queries are initiated by drawing a trajectory in a window. The system will find scenes from the scene database that contain an object whose trajectory is similar to the user-sketched trajectory. The system also allows the user to specify a name for the moving object; the query will then match only scenes that contain the specified object moving with the given trajectory.

A scene-based query can be posed either by a scene name or by an animated scene. To retrieve scenes similar to an existing scene, the user must specify the name of the existing scene. For retrieval based on an animated scene, the user first creates the animation as specified above for animation insertion. The system then plays the animation to set up the scene graph, and sends the scene graph to the retrieval engine where scene-similarity matching is performed.

The system also supports a simple text-based query language to retrieve scenes. This type of query specification is based on the moving object algebra described in Section 3 and is similar to the query specification used in traditional database systems.

The user is not only able to retrieve scenes from the database, but can also create new scenes based on existing scenes by using the *projection* and *overlay* operations. The easiest way to create a new scene is via the query language. However, the GUI also provide facilities to build new animated scenes from existing ones.

6.6. Scene database

The scene database is partitioned into three repositories: video/animation clips, scene graphs and motion icons (*micons*). In the first, we store either original video/animation clips or references (URLs) to the original clips. The micons provide summaries of the actual clips and are stored locally since they may be presented to the user in the first stage of the retrieval process. Scene graphs also are stored locally, since they are central to the image retrieval process. Each stored scene graph contains:

- pointers to the original video and iconic versions of the scene
- a set of descriptors for the objects in the scene
- a set of spatiotemporal relationships among the objects.

7. 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 pioneering efforts in the area of spatiotemporal information and moving objects. To complete our discussion on this topic, we compare our model with related work by Lee [17], Dimitrova [9], Bimbo [6], Shearer [29], and Sistla [30]. One point worth noting at the outset is that our model is unique in its support of operations for constructing new scenes from existing ones via the *projection* and *overlay* operations; none of the other approaches support this.

In Lee's model [17], the basic unit of data is a "video record", which is comparable to our basic unit of a "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.

Lee's definition of "track" is more like our notion of "trajectory", since it does not contain temporal information. However, Lee's track notion is richer than our trajectory, since it considers four classes of motion, taking into account the translation and rotation of objects as well as camera motion. The disadvantage of Lee's rich motion modelling is that track-based queries are difficult to specify. In contrast, queries on trajectories in our model are very simple; a user need only draw a trajectory in a window and the system can automatically produce a trajectory to use in a query. In addition, we define a similarity measure for retrieval on trajectories, which is lacking in Lee's model.

Our data model is more expressive than Lee's in a number of important areas. Since our tracks contain temporal as well as spatial data, it is possible to ask queries such as "how long is an object moving in a certain direction" under our model. Also, our model addresses spatiotemporal relationships among objects in a scene, so it is possible to ask queries about how objects relate to other objects during motion.

Dimitrova's model for moving objects [9] combines object attributes such as name, type and shape of objects with trajectories and object movement. Even though our model does not specifically mention object attributes, these could be included in a straightforward manner. A query language, called EVA, and its visual counterpart, VEVA, are provided for video clip retrieval. This model is comparable to ours only in its ability to query based on an object trajectory.

A trajectory in Dimitrova's model is represented by a chain code, which is similar to our trajectory but with distances removed (in other words, a sequence of directions). Dimitrova's model supports both exact-match and similarity-match queries on the chain-codes describing an object's motion. Dimitrova's model does not support any notion of spatiotemporal relationships, so that one cannot ask queries regarding the relations among the objects in a scene.

Bimbo [6] provides an interesting model for moving objects using spatiotemporal logic. The logic is an extension of temporal logic, with the operator *until* as the main operator. Bimbo's model does not include tracks or trajectories, and so cannot answer the important class of queries based on these notions.

Bimbo's spatiotemporal logic is comparable to our spatiotemporal relationship model, with the main difference being that he uses a logic approach and we use an algebraic approach. However, queries involving spatiotemporal relationships can be formulated and answered under both schemes. Bimbo's model, however, treats moving objects as points (in our model they are pointsets), and so cannot deal with queries related to duration, overlapping and containment.

In Bimbo's model, a scene is represented by spatiotemporal logic assertions, while our model represents scenes as symbolic graphs. In retrieving similar scenes, Bimbo compares the spatiotemporal logic representation of a query scene with the spatiotemporal logic representation of scenes in a database using exact match. Our model, on the other hand, provides both exact and similarity matching for scenes.

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, a visual query editor is provided which will translate the visual query to a corresponding

spatiotemporal logic query. In such a system, is not clear how a query like "find a scene where two objects A and B enter region C and where A enters C before B does and where finally A and B collide inside C" can be supported. Such queries, as noted above, are relatively simply specify in our data model.

Shearer [29] proposed a model that extends 2D B-strings for moving objects. 2D B-strings can represent both directional and topological relationships. However, Shearer's scheme can only model topological relationships on rectangular objects (or on the bounding rectangle of complex objects). This model can only be used for scene matching, and has limited support for similarity matching on scenes (only three types of matching).

Shearer's model is considerably less expressive than ours since it provides no support for modelling either spatiotemporal relationships between objects or the trajectories of individual objects.

A new model for moving objects called MOST (Moving Objects Spatio-Temporal) has been proposed by Sistla [30]. This model tackles the problem more from a database perspective and attempts to devise an approach to handling rapidly changing "data" such as the position of an object. Rather than representing a moving object as a single database record that is constantly being updated, or by a set of $(time, location)$ records, it stores the moving object as a record with static information and a *function* to compute the object's location at any specified time. Whether this approach would be useful for representing video data would depend on how easy it was to develop the movement functions for the objects in a scene. This does not seem like a trivial task.

MOST has a comparable concept to our spatiotemporal relationship called a "database history". However, the representation of database history is not formally defined, leaving it open to question whether it can handle spatiotemporal queries such as those described above. A query in MOST is like other traditional database queries (MOST uses an SQL-like syntax) enriched with spatial operations and temporal logic operators such as the *until* operator. There is no definition of scene similarity retrieval.

In summary, our model has a number of unique features, including: the combination of tracks and spatiotemporal relationships into a single framework, the ability to create a new scene based on existing scenes using the projection and overlay operations, and support for exact as well as similarity retrieval. Table 0 summarizes the above comparison.

8. Conclusion

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

A moving object is modelled as a pair *(symbol,track)* (called an object descriptor). The symbol uniquely identifies the object and the *track* describes how the object moves. A scene is modeled as a graph where the nodes contain object descriptors, and the edges are labeled by spatiotemporal relationships (based on our 2D-PIR model).

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

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.
Query based 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
Creation of new scenes	No	No	No	No	No	Yes

Table 0. Comparison of our moving objects model with related models

also for constructing new scenes from existing ones. We illustrated in specific applications how to retrieve moving object(s) stored in a database and create new scenes from existing ones using the available operations.

A scene graph is constructed from a sequence of images using a simple, efficient method. The construction leads to a graph structure for which an efficient scene-matching algorithm has been 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. For scene similarity, we integrate trajectory similarity and sequence similarity.

We have implemented a prototype moving object retrieval system. The system supports sketch-based queries (graphical queries) and a symbolic query language. We consider both kinds of queries to be necessary to effectively handle a wide range of user queries.

We have compared our model with several related models and found that it provides more features: it supports both trajectories and spatiotemporal relationships; it supports approximate and exact match queries; it allows the generation of new scenes from existing scenes. The latter feature is unique to our model.

In summary, the major contribution of this paper is a model for dynamic multimedia data. Based on this model we develop formal similarity measures on trajectories, spatiotemporal relationships and scenes so that similarity retrieval on dynamic multimedia data can be conducted according to user needs. A set of powerful operations on moving objects are derived from the model. We have shown how to apply these operations for asking queries about moving objects in a number of useful applications.

At present, our work supports only translational motion and assumes that the moving objects are reasonably rigid. Hence, complex motions such as dust-storms or tornadoes or exploding objects cannot be represented. Nor do we currently support rotation and scaling of images. Also, our work depends on being able to identify scenes in videos, identify the objects in each image in a scene, and track the motion of objects over a temporal sequence of images. The first of these tasks is currently the subject of much work, and some reasonable results are emerging (e.g. [7]). The other two tasks cannot be accomplished automatically at present; although semi-automatic methods are available for object segmentation and tracking (e.g. [9]).

Future work in this area includes investigating the completeness of our proposed operations and developing efficient query processing schemes. Another interesting and challenging aspect for future work is the development of indexing for scenes to improve the efficiency of retrieval.

Acknowledgments

We would like to thank the annomymous referees for the very helpful comments.

Notes

1. Information Systems Group, Department of Agroindustrial Technology, Bogor Agricultural University, Indonesia
2. INFORMATION SYSTEMS GROUP, DEPARTMENT OF AGROINDUSTRIAL TECHNOLOGY, BOGOR AGRICULTURAL UNIVERSITY, INDONESIA

3. INFORMATION SYSTEMS GROUP, DEPARTMENT OF AGROINDUSTRIAL TECHNOLOGY, BOGOR AGRICULTURAL UNIVERSITY, INDONESIA
4. On a related point, in order to implement our model, we assume that some mechanism exists to identify and track the movement of individual objects in a picture. While this process cannot currently be fully automated, we assume that image processing methods will eventually produce an automatic solution. For the present, we can use existing semi-automatic techniques for the extraction of objects from images and the tracking of object movement.
5. In this paper, we concern ourselves primarily with 2-dimensional scenes from video. In fact, our model extends quite naturally to 3-dimensional image data, or indeed n-dimensional data.
6. We use the term "scene" to denote a sequence of images from a single camera over a contiguous time period. Other video retrieval work uses the term "shot" to apply to such a sequence.
7. This is not unreasonable, given that a video can be regarded as a sequence of still images separated by very small time intervals, and we do not actually know the motion of objects in between the two image instants.
8. Assume that the repeated (ic, d, d) component in the patterns will be elided by the concatenation $(+)$ operation.
9. We would need to consider all objects if no semantic information about objects were available.

References

1. T. Abraham and J.F. Roddick. 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. ACM. Special feature on visual information management. *Communications of the ACM*, 40(12), December 1997.
3. K. K. Al-Taha, R. T. Snodgrass, and M. D. Soo. Bibliography on spatiotemporal databases. *ACM SIGMOD Record*, 22(1):59–67, March 1993.
4. J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983.
5. E. Bertino, M. Damiani, and P. Randi. Multimedia data handling in a knowledge representation system. In *Future Database '92: Proceedings of the Second Far-East Workshop on Future Database Systems*, volume 3 of *Advanced Database Research and Development Series*, pages 133–141, Kyoto, Japan, April 1992. World Scientific.
6. A.D. 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.
7. J.S. Boreczky and L.A. Rowe. Comparison of video shot boundary detection techniques. In *Storage and Retrieval for Image and Video Databases IV*, volume 2670 of *SPIE Proceedings*, pages 170–179, San Diego/La Jolla, California, February 1996. The International Society for Optical Engineering.
8. 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.
9. N. Dimitrova and F. Golshani. Motion recovery for video content classification. *ACM Transactions on Information Systems*, 13(4):408–439, October 1995.
10. M. J. Egenhofer. Point-set topological spatial relations. *Int. J. Geographical Information Systems*, 5(2):161–174, 1991.
11. M. Flickner, H. Sawhney, W. Niblack, 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.
12. C. Freksa. Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54(1-2):199–227, 1992.
13. O. Guenther and A. Buchmann. Research issues in spatial databases. *ACM SIGMOD Record*, 19(4):61–68, December 1990.
14. R. Hjelsvold. Video information contents and architecture. In *Advances in Database Technology - EDBT'94. Fourth International Conference on Extending Database Technology*, volume 779 of *Lecture Notes In Computer Science*, pages 259–272. Springer-Verlag, Cambridge, U.K., March 1994.
15. K. Kaneko, S. Kuroki, and A. Makinouchi. Design of 3D CG data model of "move" animation database system. In *Future Database '92: Proceedings of the Second Far-East Workshop on Future Database*

Systems, volume 3 of *Advanced Database Research and Development Series*, pages 364–372, Kyoto, Japan, April 1992. World Scientific.

16. N. Kline. An update of the temporal database bibliography. Technical Report TEMPIS No. 58, Department of Computer Science, The University of Arizona, 1993.
17. S-Y. Lee and H-M. Kao. Video indexing – an approach based on moving object and track. In *Storage and Retrieval for Image and Video Databases*, volume 1908 of *SPIE Proceedings*, pages 25–36. The International Society for Optical Engineering, February 1993.
18. K. Li, N. Badji, and R. Laurini. Modeling moving objects in spatiotemporal databases. Technical report, Department of Information and Communication Engineering, KAIST, 1992.
19. 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.
20. M. Nabil. *Modelling and Retrieving Multimedia Data Using Spatial and Temporal Information*. PhD thesis, School of Computer Science and Engineering, The University of New South Wales, 1997.
21. M. Nabil, A. H. H. Ngu, and J. Shepherd. Modelling spatiotemporal relationships in multimedia database. In *IEEE Workshop in Spatial and Temporal Interaction: Representation and Reasoning*, pages 171–189, Singapore, November 1994.
22. 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, Singapore, November 1994.
23. M. Nabil, A. H. H. Ngu, and J. Shepherd. Picture similarity retrieval using 2d projection interval representation. *IEEE Transactions on Knowledge and Data Engineering*, 8(4):533–539, 1996.
24. M. Nabil, J. Shepherd, and A. H.H. Ngu. An image retrieval system for distributed environments. In *Eighth International Workshop on Research Issues in Data Engineering (RIDE98) – Continuous-Media Databases and Applications*, pages 67–74, Orlando, Florida, February 1998.
25. 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.
26. 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.
27. H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley Publishing Co., 1990.
28. S.Devadiga, D.A. Kosiba, U. Gargi, S. Oswld, and R. Kasturi. A semiaoutomatic video database system. In W. Niblack and R. C. Jain, editors, *Storage and Retrieval for Image and Video Databases III*, volume 2420 of *SPIE Proceedings*, pages 262–267. The International Society for Optical Engineering, February 1995.
29. K. Shearer, S.K. Vankatesh, and D. Kieronska. The visitors guide: a simple video reuse application. Department of Computer Science, Curtin University of Technology, Australia, 1996.
30. A.P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and querying moving objects. In *Proceedings of the Thirteenth International Conference on DataEngineering (ICDE'97)*, pages 422–432, Birmingham, U.K., April 1997.
31. C. T. Yu and W. Meng. *Principles of Database Query Processing for Advanced Applications*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, 1998.
32. H.-J. Zhang, S. W. Smoliar, and J. H. Wu. Development of a video database system. In *Digital Libraries: Current Issues, Digital Libraries Workshop*, volume 916 of *Lecture Notes in Computer Science*, pages 253–264. Springer-Verlag, May 1994.