Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Sciences 160 (2004) 131–152
INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL
www.elsevier.com/locate/ins

Spatio-temporal querying in video databases

Mesru Koprulu [a], Nihan Kesim Cicekli [a,b,*], Adnan Yazici [a]

[a] *Department of Computer Engineering, Middle East Technical University, Ankara 06531, Turkey*
[b] *Department of Computer Science, University of Central Florida, Orlando, FL 32816 USA*

Received 5 August 2002; received in revised form 31 July 2003; accepted 7 August 2003

Abstract

A video data model that supports spatio-temporal querying in videos is presented. The data model is focused on the semantic content of video streams. Objects, events, activities, and spatial properties of objects are main interests of the model. The data model enables the user to query fuzzy spatio-temporal relationships between video objects and also trajectories of moving objects. A prototype of the proposed model has been implemented.

1. Introduction

As a result of recent advances in the computer technology, multimedia systems like VOD (video on demand) services, geographical information systems, digital video libraries, teleconferences, security systems etc., have become ordinary parts of our life. With the high usage of multimedia systems in our daily life, some new requirements have arisen, like querying the multimedia data. Content-based queries on images have been studied for a relatively long time and many solutions have been developed for storing and querying images in databases. Querying video databases, however, is a relatively new concept.

Although querying video databases is a newer concept, there has been a considerable amount of work on querying the content of videos [1,8,11–

* Corresponding author. Address: Department of Computer Engineering, Middle East Technical University, Ankara 06531, Turkey. Fax: +90-312-210-1259.
E-mail address: nihan@ceng.metu.edu.tr (N.K. Cicekli).

0020-0255/$ - see front matter
doi:10.1016/j.ins.2003.08.011

13,16,18]. Some of the existing work use annotation-based modeling [5,8], some use physical level video segmentation approach [7,18], and some have developed object based modeling approaches [1,6,9,11,13,16]. Annotation-based video models allow free text or attribute/keyword annotations in order to describe semantic attributes of video data. The physical level video segmentation approach does not address semantic concepts, instead the video data is described as a stream of small segments with application specific temporal and spatial properties. Object based models focus on the modeling of semantic content of the video data using object-oriented modeling techniques. Among the object-based models, the Advanced Video Information System (AVIS) [1] introduces indexing based on the objects and events of interest in the video. The model is capable of performing complex and compound content-based queries within a well-defined index structure. However it lacks spatial information and thus, it does not support spatial and spatio-temporal queries.

Multimedia data contains various data types, which are either structured or unstructured. Especially for unstructured data types, uncertainty is an important issue. There are three different types of uncertainty in multimedia data [15]. (i) One refers to fuzzy objects, that is, objects that do not have well defined boundaries. (ii) Objects may be related by ''fuzzy relations'', i.e., a pair of objects may belong to multiple relations with different grades of membership. (iii) Ambiguous interpretations of spatial relations may cause uncertainty in query processing. Uncertainty and fuzzy spatial queries in multimedia databases have been widely studied by some researchers. For instance, [4,15,17] deal with the last two types of uncertainty in spatial databases. Aygun and Yazici [3,19] use a fuzzy object-oriented conceptual data model to represent object-oriented concepts in video data.

In this paper we present a video data model that supports fuzzy spatio-temporal querying. The data model is developed as an extension to the AVIS model [1]. We have chosen to use AVIS as a basis for our study, since it is a flexible model that can be used for any kind of video data. It is application-independent, and allows the user to store as much semantic meta-data as required. Our extended data model has the following properties and contributions: (1) it supports modeling the semantic content of video data including the spatial properties of objects. The extensions to AVIS include changing the association map, so that it can represent spatial properties of objects, being able to store spatial data together with objects in the frame segment tree, and introducing an array structure for objects to store spatial data associated with them, (2) the model allows us to perform spatio-temporal queries on the video, including querying spatial relationships between objects in the video and querying moving objects in the video, (3) the model also allows inclusion of fuzziness in spatial and spatio-temporal queries; therefore, one is able to do fuzzy querying, and (4) a prototype implementation of the proposed model is developed. The prototype implementation includes, in addition to all basic

queries supported by AVIS, querying spatial properties of objects, (fuzzy) spatio-temporal relationships between objects and (fuzzy) object trajectory queries, a user-friendly data entry interface and the handling of more than one video at a time.

The rest of the paper is organized as follows: Section 2 presents a summary of the related work. Section 3 describes the basic data model. In Section 4, we introduce our extended data model that can handle spatial properties of objects and spatial relationships between objects. Section 5 discusses the query types that are supported by our model. Our extended model allows (fuzzy) spatio-temporal queries on video objects. The implementation of the model and a number of spatio-temporal queries are briefly presented in Section 6. The last section concludes the paper with some remarks for future extensions.

2. Background

Multimedia data has different formats. Video is one of those formats, and it is the most complex one. Video data is usually a combination of other simple multimedia data types, like sound, image, text, etc. Each data type contained in a video brings its own properties to be owned by the video. Video data has both temporal and spatial properties. Besides, it has a semantic content like all multimedia data types. Moreover, the volume and unstructured format of video data make it difficult to manage, access, and compose video segments into video documents. A video database system must be capable of storing and querying every property of a video object.

Since video data is a very rich information source, each user may extract different information from the same video according to their point of interests. Some may be interested in trajectories of moving objects in video, while others may be interested in the occurrences of objects and events in the video. We need some pre-work on the video data, in order to have an indexed representation of video data and its content. Because of these different application requirements, different modeling techniques on video data have been developed. Here we present a summary of these different modeling techniques.

2.1. Models based on physical level video segmentation

The main idea in this approach is to handle the video as small pieces, instead of a huge bulk of video stream. A video stream is composed of video segments that are created using different techniques. One of those techniques is histogram matching. Each considerable change in the histogram causes a new segment to be created.

Another technique is proposed in the algebraic video data model of Weiss et al. [18]. In this technique a set of algebraic operations are applied to a video

stream recursively. As a result, a set of video segments is created. The created segments are then defined with their main characteristics, which may change according to the application itself. The primary property of these segments is that they are temporally disjoint from each other. News video databases are the best application areas for this type of models.

The study of Gibbs et al. [7] includes how to model stream-based temporal multimedia data, from a temporal point of view. This study did not address semantic concepts, such as, objects, events, roles, players, etc. Also, integration of this system with traditional databases is not stated in the study.

2.2. Models based on annotation layering

The main weakness of video segmentation-based models is their lack of flexibility. An alternative approach is to use a layered annotation representation, so that it is possible to segment videos semantically. In this approach, each semantic segment is assigned an annotation describing the semantic content of the segment. These annotations are stored in a database in order to perform keyword-based querying on the video content.

Davenport et al. [5] propose such a model and call it "stratification model". This model uses the segmentation approach and is a combination of two approaches. Their point of view is that, if a video can be annotated at the finest grain, the lower level of granularity can be built up from the existing annotations. They suggest the idea of a data camera, which, in addition to recording videos, will record some of the lower levels of annotation data required.

Videotext model [8] is also an annotation-based video model. It allows free text or attribute/keyword annotations in order to describe semantic attributes of video data. It also handles temporal relationships between any two-frame sequences. These models allow video annotations to be added, removed and modified, independent of the raw video data underlying the model.

2.3. Models based on video objects

As the object-based modeling of video data has become more popular, traditional object-oriented databases have been more of interest in multimedia data modeling area. A video object is given different meanings in different approaches.

Oomoto and Tanaka [13], have defined an object-oriented data model, named OVID. In their study, a video object is a set of frames that contain an event, a person, or some other entity in the video, or any combination of these. Notable features of the model are to provide a mechanism to share common descriptive data among video objects, called "interval-inclusion based inheritance", and to provide operations to composite video objects. Their study also

includes a special query language called VideoSQL, which is an extended version of SQL for video databases.

In [16], Theodoridis et al. propose a different point of view to the video data. They propose efficient indexing schemes on spatio-temporal properties of salient objects in a video. They use three-dimensional *R*-trees for indexing spatio-temporal data, where time is represented by the third dimension of the *R*-tree.

Kuo and Chen [11] have developed a video query language (CVQL), which uses spatial and temporal relationships between salient objects as query predicates. In their study a macro mechanism is introduced to simplify the query specification and an elimination-based query processing algorithm is described.

A rule-based video database system architecture is proposed in [6]. The system incorporates both spatio-temporal and semantic query facilities. Their rule-based approach eliminates the need for the computation of relations at the time query processing. Video clips are segmented into shots using spatial relationships between objects. The system has a set of inference rules to reduce the number of facts stored in the knowledge base.

In the logical hypervideo data model [9], the focus is on the objects that are of interest (defined as hot objects, in their study). Hot objects are represented with their semantic associations with other logical video abstractions, including other hot objects. The associations are modelled as video hyperlinks. The proposed model is capable of handling spatial and temporal queries as well as semantic queries on the objects.

3. Basic video data model

Our data model is based on the data model used in AVIS (Advanced Video Information System) [1]. In the basic model, the main focus is on objects, events, and activities, called *entities*, appearing in the video. The video *objects* may be characters in a movie such as Rambo, James Bond, etc. or they may be objects such as a car, a tree, a ball, etc. An *activity* is accepted as the subject of a given frame sequence. It is the type of the action performed in a video (e.g. walking, eating, kicking a ball). Multiple activity types may occur simultaneously in a frame sequence. An *event* is an instance of an activity type. It consists of a unique activity type, a set of roles in the activity, and objects as the actors of roles in the activity. For example, in the event "Philip is eating a cake", the activity type is eating, the role is eater, and the actor is Philip.

Each entity has a set of frame sequences attached to it. These are the frames in which the entity appears in the video. The collected information about entities is indexed with some special index structures. During query processing, these index structures are used in order to retrieve information about entities, and more importantly, to make correlations between entities.

3.1. Frame sequences and association maps

A frame sequence is a set of contiguous frames containing a semantically important data, like an object or an event. A frame sequence can be considered as a shot defined by segmenting a video stream using a semantic entity. An association map is constructed from frame sequences as a visualization tool to view the segmentation of the video stream. Adali et al. give a formal description of a frame sequence in their paper [1].

Definition 1. Let ENT be a set of entities in a video stream. An association map is a function, λ, such that for each entity $ent \in \mathrm{ENT}$, $\lambda(ent)$ is the set of frame sequences including entity ent. $\lambda(ent)$ is a solid set of frame sequences (a solid set is a set of frame sequences with no adjoining frame sequences).

An association map represents the main idea in this data model. An association map corresponds to horizontal line segments on the Cartesian plane, where x-axis is the time and y-axis is the set of entities. A line segment in the association map represents the occurrence of an entity in the video during the corresponding frame interval. The model provides means to represent entities of interest in a video, frame sequences to identify the occurrence time intervals of entities, and association maps to combine entities and frame sequences into one graph. Thus, association maps can be stored using any method to store such collinear line segments, such as the well-known segment tree. The next step is to have an effective indexing strategy for an efficient retrieval of video data.

3.2. Frame-segment tree

Indexing is mainly based on a segment tree, which is named as the *frame-segment tree* (FST). The FST is constructed from the line segments in the association-map. A great deal of the line segments for different entities overlap in the association map. The overlapping line segments are divided into smaller segments by imaginary vertical lines, so that overlapping sections of line segments for different entities are considered as a separate node in the frame segment tree. A list of entities that appear in the time interval represented by a node is stored along with the node in the tree.

FST has the following properties: Each node in the frame segment tree represents a frame-sequence $[x, y)$ starting at frame x and including all frames up to, but not including, frame y. Note that y is not included in the interval, because if it were, then, the next interval $[y, z)$ would also include frame y which would be confusing. Each node is associated with a set of numbers that correspond to the identities of video objects and events that appeared in the entire frame-sequence represented with that node. Thus, for example, if a node N represents the frame sequence $[i, j)$ and object o occurs in all frames in $[i, j)$,

then object o labels node N (unless object o labels an ancestor of node N in the tree). If an object exceeds the frame interval of a node, then the object is represented in more than one node. For any node, if an object does not appear in all frames of that node, then, the object is inserted into the lists of sub-nodes of the node.

3.3. Array structures

There are three array structures that provide additional indices to the nodes of the FST. These are *ObjectArray*, *EventArray,* and *ActivityArray*. Each entity in the database is loaded into the corresponding array depending on the type of the entity, using a hashing method. The ObjectArray is an array whose ith element contains an object with hash number i. Each element in the array is associated with a list of numbers each of which represents a node that contain object i in the frame segment tree. Each element also contains additional information about the object, like the name of the object.

The EventArray stores event descriptions. Each event has an identity calculated from the unique properties of the event. The EventArray also stores pointers to the corresponding nodes in the frame segment tree.

The ActivityArray stores the activity types together with the events associated with it. Activity type and event entities have a 1:N relation in between. Therefore, an activity type may point to more than one event, whereas, an event can point to only one activity type. ActivityArray does not store pointers to the FST like other two arrays. When an access to FST is needed from this array, this is done indirectly over the EventArray. For example, when we want to find the frame intervals of a given activity type, we first find the events pointed to by the activity record, then we find the frame intervals of each of these events. For all arrays, pointers to the frame segment tree nodes are kept ordered according to the frame interval they represent.

4. Modeling spatio-temporal relations in video

One important issue in multimedia databases is to represent spatial relationships between objects in an image or a video. Handling spatial relationships in video databases is more difficult than in image databases, since video data has time-dependent properties. Some examples of spatial queries that can be applied to video data are as follows:

- Find all frames, where Bulent Ecevit *is at the right of* George Bush, and Tony Blair *is at the left*.
- What is the *spatial relationship* between the victim and the murderer, when the murder happens?

- Find the name of the Formula 1 pilot that is *behind* Michael Schumacher between frames 100 and 120.

These examples may be propagated by adding new spatial predicates or by combining the existing ones. Such queries cannot be answered in the basic AVIS model. We need to extend the basic data model in order to represent the spatial properties of objects so that such queries can be handled.

In the model proposed here, our main objective is to present an approach for modeling spatial properties. More specifically, we represent the location of an object by dynamically defined rectangular areas within a time interval. A rectangular area covers all the appearances of the object during that time interval. With the addition of spatial properties of objects, we are able to handle spatial queries along with some fuzzy specifications to the model. The trajectory of an object can also be computed within this data model. In the rest of this section, we discuss the representation of spatial properties of objects and how different spatial relationships between objects are computed and show how we handle the processing of various spatio-temporal queries.

4.1. Spatial properties of video objects

In general, a spatial property of an object contains information about the spatial location of an object in a video frame. In our model we use the two-dimensional coordinate system, in which the spatial property of the object is defined to be the object's position as it appears on the screen. As stated in [12,14], a common strategy is to use approximations for describing an object spatially. The most efficient method for the two-dimensional coordinate system is to use MBRs (minimum bounding rectangles). An MBR is an imaginary rectangle that is accepted to be the minimum rectangle that covers all parts of an object. We also use the MBR approach in approximating the location of objects in video streams. The granularity of the coordinate system is an important issue, since a minimum of pixel-level granularity makes it hard to apply to video data. In our model, the user-specified rectangular areas on the screen represent the locations of objects.

Definition 2. The location of an object A_i is defined by the rectangular area (a region) $[(X_{1i}, Y_{1i}), (X_{2i}, Y_{2i})]$ where $X_{1i} \leqslant X_{2i},\ Y_{1i} \leqslant Y_{2i}$.

This representation can be extended considering the time dimension that covers the duration in which this spatial property is true. Thus, we define the spatial property of an object as follows:

Definition 3. The spatial property of an object A is a $tuple(R, I)$, where, R is a rectangular area (a region) which covers a minimal area in which object A appears during the time interval $I = [t_i, t_f]$. R is obviously not the minimum-bounding rectangle of A. At any time t in $[t_i, t_f]$, object A may be located at anywhere in R. It is an approximation of a MBR.

Object A may be either static or moving in the rectangle R during the interval I. When the object A is spatially static in R, then the rectangle R is actually the minimum-bounding rectangle of A. When A is moving in R, then we assume that the rectangle R is chosen so that A has a uniform movement in R, i.e., in a reasonable distance and as a single moving character. For example, while focusing on an object in the frame, if camera zooms out the object while it is moving, or if the object itself goes away, the size of rectangle that covers the object gets smaller. In this case, for a fine result, the scenes of zoom out should be handled in separate rectangles.

4.2. Spatio-temporal relationships between objects

In a given frame, the spatial relationship between two objects can be defined using the spatial relationship between the MBRs of each object. This property gives us the ability to find the spatio-temporal relationships between any two objects in a frame sequence. We use the spatial properties of video objects to define the spatio-temporal relationship between them in a video stream.

We developed a rule base for representing each spatial relationship (UP, LEFT, etc.) between objects [10]. Li et al. have a formal definition of these relations for a similar rule base [12]. They extended Allen's temporal interval algebra [2] into two-dimensional space in order to define spatial relationships between rectangular areas. They divide spatial relations into two groups; *directional and topological*. Directional relations include *south*, *north*, *west*, *east*,*northwest*, *northeast*, *southwest*, and *southeast*. Topological relations include *equal*, *inside*, *contain*, *cover*, *covered by*, *overlap*, *touch*, and *disjoint*. We use the notation of [12] during this formalization. Table 1 shows temporal interval relations between two objects. These relations are used for defining spatial relations in Table 2.

In our notation, we use *left*, *right*, *top*, and *bottom* instead of *west*, *east*, *north*, and *south*, respectively. Our rule base covers the relations *top*, *bottom*, *right*, *left*, *top-right*, *top-left*, *bottom-right*, *bottom-left*, *overlaps*, *equal*, *inside*, *contain*, *touch*, and *disjoint*. Any two regions satisfying the definition of any of these relations given in Table 2, have those corresponding spatial relationship(s) between the objects they contain.

Such a rule base can help the calculation of spatial relationships between two static objects, like in simple images, accurately. However in videos, objects are not static. They may be moving or they may seem to be moving because of

Table 1
Temporal interval relations

Relation	Symbol	Inverse	Meaning
A before B	b	bi	*AAA BBB*
A meets B	m	mi	*AAABBB*
A overlaps B	o	oi	*AAA* *##BBB*
A during B	d	di	*##AAA* *BBBBBBBB*
A starts B	s	si	*AAA* *BBBBBB*
A finishes B	f	fi	*###AAA* *BBBBBB*
A equal B	e	e	*AAA* *BBB*

Table 2
Definitions of spatial relations

Relation	Definition
A bottom B	$A_x\{b,bi,m,mi,o,oi,d,di,s,si,f,fi,e\}B_x \wedge A_y\{b,m\}B_y$
A top B	$A_x\{b,bi,m,mi,o,oi,d,di,s,si,f,fi,e\}B_x \wedge A_y\{bi,mi\}B_y$
A left B	$A_x\{b,m\}B_x \wedge A_y\{b,bi,m,mi,o,oi,d,di,s,si,f,fi,e\}B_y$
A right B	$A_x\{bi,mi\}B_x \wedge A_y\{b,bi,m,mi,o,oi,d,di,s,si,f,fi,e\}B_y)$
A top-left B	$(A_x\{b,m\}B_x \wedge A_y\{bi,mi,oi\}B_y) \vee (A_x\{o\}B_x \wedge A_y\{bi,mi\}B_y)$
A top-right B	$(A_x\{bi,mi\}B_x \wedge A_y\{bi,mi,oi\}B_y) \vee (A_x\{oi\}B_x \wedge A_y\{bi,mi\}B_y)$
A bottom-left B	$(A_x\{b,m\}B_xA_y\{b,m,o\}B_y) \wedge (A_x\{o\}B_x \wedge A_y\{b,m\}B_y)$
A bottom-right B	$(A_x\{b,m\}B_x \wedge A_yb,m,oB_y) \vee (A_x\{oi\}B_x \wedge A_y\{b,m\}B_y)$
A overlaps B	$A_x\{d,di,s,si,f,fi,o,oi,e\}B_x \wedge A_y\{d,di,s,si,f,fi,o,oi,e\}B_y$
A equal B	$A_x\{e\}B_x \wedge A_y\{e\}B_y$
A inside B	$A_x\{d\}B_x \wedge A_y\{d\}B_y$
A contain B	$A_x\{di\}B_x \wedge A_y\{di\}B_y$
A touch B	$(A_x\{m,mi\}B_x \wedge A_y\{d,di,s,si,f,fi,o,oi,m,mi,e\}B_y)\vee$ $(A_x\{d,di,s,si,f,fi,o,oi,m,mi,e\}B_x\wedge A_y\{m,mi\}B_y)$
A disjoint B	$A_x\{b,bi\}B_x \vee A_y\{b,bi\}B_y$

a moving camera or zooming in or out. Therefore we need to deal with spatial relationships that change over time. At a certain snapshot of the video, one object may appear to be exactly to the left of another object. But within a period of time, '*o1 is to the left of o2*' might not be very exact all the time. Therefore we introduce fuzziness to the definition of spatial relationships between two objects in videos. We call this a fuzzy spatio-temporal relationship and define it as follows:

Definition 4. Let A_i and A_j be two objects. Their fuzzy spatio-temporal relationship during time interval I_k is $A_i(\alpha,\mu,I_k)A_j$ where α is one of the relations given in Table 2, and μ is the value of membership. $A_i\alpha\mu A_j$ is true during the interval I_k.

Notice that this definition is similar to the definition of spatial relations between two moving objects given by Li et al. [12]. Their definition includes both directional and topological relations. Two different operators are used representing each relation type. We have used only one operator α for both directional and topological relations. However we use a fuzzy membership value, μ, which is in $[0,1]$ for any spatial relationship. The membership value determines how much the spatial positions of two objects satisfy a given spatial relation.

According to the formulas given in Table 2, we present some examples of LEFT relation between two objects in Fig. 1. In this figure, each occurrence of object A is on the LEFT of object B with different membership values. Each positioning of A and B satisfies the formula for the definition of LEFT given in Table 2. For a given time interval I and a membership μ, this relationship will be specified by $A(\mathrm{LEFT},\mu,I)B$.

The membership value is used in the second step for the calculation of a relationship. The first step is to determine if any two objects satisfy the conditions stated in Table 2. If these conditions are not satisfied, membership value μ is assumed to be 0. Otherwise, we calculate the membership value of the relationship using the centers of rectangles. The angle between the x-axis and the line between centers of the rectangles are used to calculate the membership value. Table 3 illustrates the relation between the angle and the membership value for each relation. Notice that, in this table, we represent only relations, *top*, *left*, *top-left*, and *top-right*. The reason is that, these relations are inverses of *bottom*, *right*, *bottom-right*, and *bottom-left* respectively. That is, $\mathrm{TOP}(A,B)$ is equal to $\mathrm{BOTTOM}(B,A)$, and so on. We also used this idea in our implementation. Therefore, these four relations are enough for the calculation process. The calculation of membership values allows us to perform fuzzy queries on video data. The user is able to specify the preciseness of the result of any query.

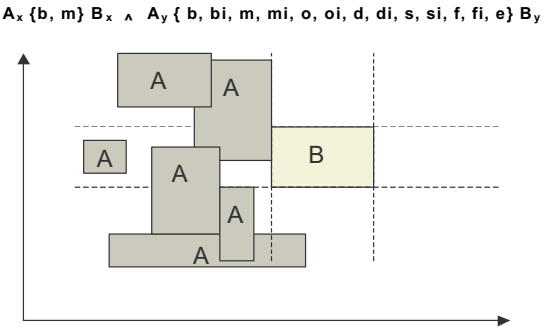

Fig. 1. Examples for LEFT relationship for two objects.

Table 3
Relation between the membership value and angle between the centers of rectangles

Relation	Angle (∗)	Membership value
Top	$\arctan(x/y)$	1-(angle/90)
Left	$\arctan(y/x)$	angle/90
Top-left	$\arctan(x/y)$	1-(abs(angle-45)/45)
Top-right	$\arctan(y/x)$	1-((angle-45)/45)

*x is the horizontal distance between centers of two rectangles.
*y is the vertical distance between centers of two rectangles.

4.3. Data structures

In this section we discuss how we incorporate spatial properties of video objects into the basic data model. Three structures are affected from these changes. These are: ObjectArray, frame segment tree, and the association map.

4.3.1. New object array

In the original ObjectArray each element points to a set of frame intervals associated with it. In the new ObjectArray, each element contains the information for an object together with all the frame interval/region pairs. That is, an object would have a list of interval/region pairs (i, r) that shows the interval $[i_{\text{start}}, i_{\text{end}})$ in which the specified object appears in region r. The new structure of the object array is shown in Fig. 2. Thus, an item in the ObjectArray holds a list of object records. An object record contains the name of the object, a list of (*interval*, *region*) pairs, showing the intervals during which the object appears in the stated region. Additionally, for each (*interval*, *region*) pair, there is a list of pointers each of which points to the nodes in the frame segment tree, which are covered by that interval. This list of pointers provides high speed in accessing the corresponding tree nodes for any object. In this way, we benefit from the tree traversal cost.

4.3.2. New frame segment tree

In order to implement the region concept, the node structure in the frame segment tree is modified. In our model, a tree node is defined to contain a list of

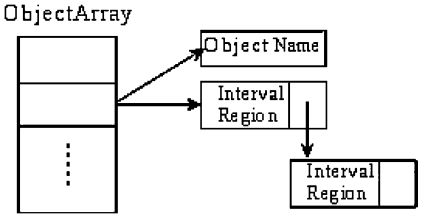

Fig. 2. The new object array structure.

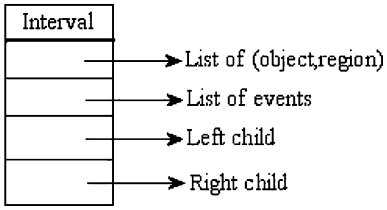

Fig. 3. Node structure of frame segment tree.

(*object*, *region*) pairs. This means that, each (o_i, r_i) pair in the list states that, "object o_i appears in region r_i within the time interval represented by that node". Fig. 3 shows the node structure that we use in the new frame segment tree. Each node also contains a list of events.

Storing objects with their regions has an important effect on the frame segment tree. An object may appear in more than one region during a set of continuous frames. For example, a plane crossing the screen from left to right, visits more than one region. In the basic data model, this entity can be represented in only one FST-node, since the frames are continuous. However, in the spatio-temporal extension, we divide this single interval into sub-intervals, when the region of the object changes. This means that each of these sub-intervals is represented with a separate tree node. As a result, the number of nodes in the tree increases. This affects the cost of operations on the tree proportional to the order of $\log n$, where n is the number of nodes in the tree.

4.3.3. New association map

The association map is changed in such a way that the time interval in which an object appears in a region could be represented individually. As a result, in the new association map we represent (*interval*, *region*) pairs of objects along with the intervals of event occurrences.

An object can have different spatial properties in consecutive time intervals. Therefore, we need to draw the association map so that these consecutive time intervals could be differentiated. In the implementation, we used different colors for these consecutive time intervals. However, in Fig. 4, which presents a sample drawing of the new association map, we used different patterns. In this figure, events are represented with basic lines, since we do not need to segment a continuous time interval for an event, and we use blocks with different patterns for objects in order to represent the location changes in a continuos time interval. For example, in Fig. 4, boy has appeared in two separate time intervals, but he is seen in three different locations in the first time interval and five different locations in the second time interval.

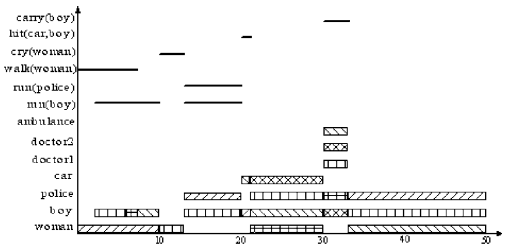

Fig. 4. The new association map.

5. Supported query types

The model introduced in this paper allows all types of queries that are listed in [1]. In addition to these query types, our model supports querying the spatial properties of objects and the spatio-temporal relations between objects as well. We list the basic spatial queries into three categories:

- *Regional queries:* Given an object and time interval, one may query the regions it appears. Or, given an object and its region, one may query the frames.
- (*Fuzzy*) *Spatio-temporal relationship queries:* Given two objects and the spatial relationship between them we can query the frames in the video. Here we use fuzziness in the definition of the spatio-temporal relationship.
- (*Fuzzy*) *Spatio-temporal queries* (also called *trajectory queries*)*:* Given an object we may query its trajectories. Or, given a trajectory of an object, we can query the time intervals.

5.1. Regional queries and fuzzy spatial relationship queries

Given object and interval, *find regions:* This is a rather simple query to handle with the data model of this study. An example query for this type can be "Give a list of locations of the ball between fifth and tenth minutes". While calculating, we first locate the object in the array. Then, we trace the (*interval*, *region*) list of the object and retrieve regions of all intervals that intersect with the given interval in the query. A list of regions is returned as a result. The regions are represented with their x–y coordinates.

Given object and region, *find frames:* An example of this type of query is "List all frame intervals during which a plane appears in a given region". The calculation is very similar to the previous one, since regions and intervals are kept in the same structure: The object is located in the array, and then all items in (*interval*, *region*) list are visited to find the given region. When the region is

found, the corresponding interval is added to the resulting list. The intervals are sorted and consecutive ones are merged to obtain a solid list of intervals. When searching for the region we apply fuzziness since the region given in the query is only an approximation of the exact location of the object. Thus the equality of the regions is checked within a threshold.

Given objects and a spatial relation in between, find frames: This type of queries are of the form "Find the set of time intervals, in which object A is seen at left of object B, with a threshold value of 0.7". This is a fuzzy spatial relationship query. The model allows us to calculate this query with a low cost, which is proportional to the number of occurrences of the objects. We trace the regions of A, and for each interval, we look at the corresponding tree node, to find the object B in a suitable region. If B occurs and if its region satisfies the relationship within the threshold value, then the interval of current node is returned. As the result of the query, we have a solid set of frame intervals in which the given relationship occurs at least 70%.

5.2. Trajectory queries

Trajectory is the route of an object on the video in consecutive frames. In a trajectory query, we search both the time interval and the spatial location of an object. A trajectory is composed of a set of sorted (*interval*, *region*) pairs, such that, any interval except the first one is the successor of the previous one in time, and any region except the first one is a neighbour of its predecessor. A formal definition of trajectory is given as the following:

Definition 5. A trajectory of an object o is a set $T_o = \{(I_n, R_n) | n > 1\}$ such that:

(i) I_n is the time interval $[t_{ns}, t_{nf}]$,
(ii) R_n is the rectangular area defined for object o in time interval I_n,
(iii) For any i, where $1 < i \leqslant n$,
- $t_{(i+1)s} = t_{if} + 1$,
- R_{i+1} is a neighbour of R_i, where neighbourhood is defined in Definition 6.

Definition 6. Given two rectangles R_1 and R_2, with coordinates $(r_1x_1, r_1x_2, r_1y_1, r_1y_2)$ and $(r_2x_1, r_2x_2, r_2y_1, r_2y_2)$ respectively, R_1 is a neighbour of R_2 if and only if one of the following conditions is satisfied:

(i) $r_1x_1 = r_2x_2 \wedge [r_1y_1, r_1y_2]$ and $[r_2y_1, r_2y_2]$ overlaps,
(ii) $r_1x_2 = r_2x_1 \wedge [r_1y_1, r_1y_2]$ and $[r_2y_1, r_2y_2]$ overlaps,
(iii) $r_1x_1 = r_2y_2 \wedge [r_1x_1, r_1x_2]$ and $[r_2x_1, r_2x_2]$ overlaps,
(iv) $r_1x_2 = r_2y_1 \wedge [r_1x_1, r_1x_2]$ and $[r_2x_1, r_2x_2]$ overlaps, where, *overlaps* is taken from Allen's temporal algebra. Since this algebra can be applied to any one-dimensional space, we used it for x-axis and y-axis.

As an example of querying trajectories, consider a query of the form: "Find trajectories of object A, that start at region R_1, and end at region R_2", where region R_1 and region R_2 are specified in the query interface (typically with the mouse). This kind of query allows us to find all trajectories that start at any region on the screen and end at any region on the screen. As a result of the query we obtain a list of linear or non-linear trajectories of the object, each of which starts in R_1, and ends in R_2. Our model allows us to find the trajectories of objects that pass the screen from left to right, or diagonally through the screen, or from top to bottom, or in the reverse directions. The returned trajectories may be not only linear trajectories but also non-linear, even spiral.

Consider the example query: "Find the trajectory of an object going from left to the right of screen". Here the left of screen and the right of screen are not precisely defined regions. We need to use approximations for these regions. The spatial property definitions of the object may not exactly fit in these stated regions. Therefore we use uncertainty in processing this kind of queries. We match the actual object regions with these imprecise regions by using approximation techniques. The degree of uncertainty is specified by the user in a trajectory query. A degree of preciseness is requested with each query in order to match the starting and the ending rectangles.

The neighbourhood of two rectangles may also involve uncertainty. We implemented a constant uncertainty degree proportional to the areas of two rectangles. The importance of the distance between the boundaries of two objects may differ according to the size of objects. For example, a distance of 100 m between two players in a football match is a long distance. However if we had two islands with 100 m of distance in between, we would consider them as very close because 100 m is almost negligible relative to the size of islands. The same logic is used in determining the degree of neighbourhood between two rectangles. This degree may be user-specified or a commonly accepted value.

6. Implementation

We developed a prototype system that supports the model presented in this paper. We first discuss the data entry and query presentation techniques for the video content, then we present the query processing algorithms with sample queries.

6.1. Data entry and query presentation

In our implementation, we have a graphical user interface for data entry. The snapshot of the user interface is shown in Fig. 5. This is a user-friendly interface, that allows watching the video, pause and step forward and backward on the video. The user can select the rectangular area that covers the

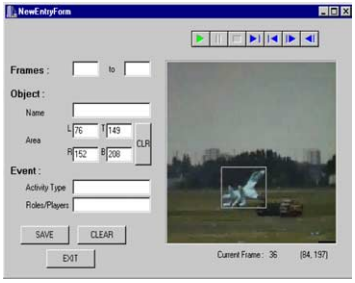

Fig. 5. Data entry form.

location of an object with the mouse. The events are also entered using this interface. Events are identified with their activity type and a set of *role:object* pairs. Event and object data can be entered together or separately. All the necessary data to construct the association map, thus the FST and arrays, are collected at the end of the data entry phase.

In query development interface the available data is presented to the user so that the user can generate queries using the existing data. Different users can express their queries in different phrases. For example, an activity can be expressed as both ''playing'' and ''play'', depending on the user. If the user does not know which one is used, s/he can generate wrong queries. Similarly, *role:actor* sets of events are rather prune to this kind of mistakes. In order to prevent it, we use pull-down list boxes for objects, events, activity types, videos, relations, and threshold values. Threshold values are editable, in order to give the user the opportunity to specify a more detailed value. In this way, we allow the user to see what is included in the database and also they are prevented from using invalid notations and keywords. A snapshot of the query development interface is given in Fig. 6. We have buttons for each query type. The

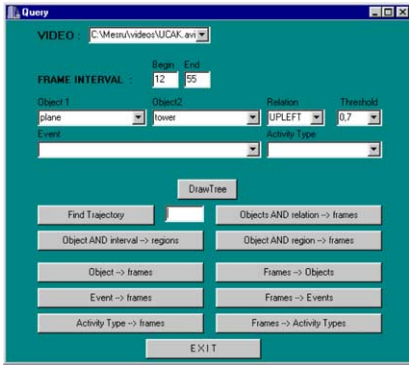

Fig. 6. The query development window.

required information is specified in the upper half of the window. In case of a missing parameter, the user is warned after selecting the query type.

The query types supported by the model produce different types of results. For some queries the output is a list of items, i.e. textual results, and some queries result in a list of video clips that satisfy the conditions in the query. We call the former "list-type query results" and the latter "frame-type query results". In frame-type query results, we merge the adjacent frame intervals and make it a single interval, thus, a single video clip. Each interval is presented to the user and the user can play whichever s/he selects.

6.2. Query processing

When the query window is activated, the user is prompted to select a video from the database. Following the selection of the video, all data about the video are loaded from the database into the arrays and the frame segment tree. The model allows all of the query types explained in the previous section. The query types that are also supported in AVIS [1] are implemented more or less in the same way. Here we present the algorithms to process the spatial queries that are supported by our extended model.

6.2.1. Regional queries

Sample query 1: "*List all regions where the ball appears between fifth and tenth minutes*".

This is a rather simple query in our model. We first locate the object in the ObjectArray. Then, we trace the (interval, region) list of the object and retrieve regions of all intervals that intersect with the given interval. The algorithm is as follows:

- *Locate the object in the ObjectArray*
- *Retrieve interval-region list of object*
- *For all pairs in the list, do*
  If given interval and current interval overlaps then
  Add the current region to the result set
- *Remove duplicate regions from the list*

A unique list of regions is returned as a result. This is a kind of query that returns list-type results.*Sample query 2:* "*List all frame intervals during which a plane appears in a given rectangular area with a given threshold value*".

The calculation is very similar to the previous one, since regions and intervals are kept in the same structure: The object is located in the array, and all items in (*interval*, *region*) list are visited to find the given region. However, approximation is used to match the regions of object with the given region. This query produces a frame-type query result. Here is the algorithm:

- *Locate the object in the ObjectArray*
- *Retrieve interval-region list of object*
- *For all pairs in the list, do*
  If given area and current rectangle are similar, then
  Add the current interval to the result set
- *Sort the intervals*
- *Concatenate adjacent intervals*

6.2.2. Fuzzy spatial relationship queries

Sample query 3: "*Find all frames, in which object A is at the left of object B with threshold value of 0.7*".

We calculate the relationships using some parameters, like rectangular coordinates. We refine our results with the user-specified threshold value. The angle between the centers of rectangles is used for this calculation. The ratio of the angle to 90° is used in a different style for each relation. The algorithm is as follows:

- *Locate objects O1 and O2 in the object array*
- *Retrieve interval-region list of O1*
- *For all items in the list*
  - *Retrieve FST-nodes associated for interval of current item*
  - *For all nodes in FST-nodes list*
    - *Retrieve object-region list of current node*
    - *For all items in list*

  If current object is O2, then
    - *Calculate relationship between region of O1 and region of current object-region pair*
    - *If membership of relationship > given membership, then*
      Add current interval to the resulting interval list

Finally, all intervals in the resulting lists are sorted, and then any adjacent intervals are merged to form a single interval. And then, the resulting intervals are presented in a window, with the opportunity to play corresponding part of video stream for each interval.

6.2.3. Fuzzy spatio-temporal (trajectory) queries

Sample query 4: "*Find all trajectories of object A, starting from rectangle R*1 *and ending at rectangle R*2, *with a threshold value of 0.8*".

The main idea in the query evaluation is to find the neighbour regions in the adjacent frame intervals of an object. Of course, these neighbour regions should reach the last region *R*2 stated in the query in order to return a valid result for the query. There can be more than one trajectory each forming a different path. This means that the object may have moved from *R*1 to *R*2

several times in the video. Thus, the result of the query is a set of lists of rectangles. Each list shows a trajectory from rectangle R1 to rectangle $R2$. These lists can be either linear or non-linear. Here is the algorithm for trajectory queries, assuming that we are given two regions $R1$ and $R2$ as starting and ending regions respectively.

- *Locate the object in the ObjectArray*
- *Retrieve interval-region list of the object*
- *Sort the list according to intervals*
- *For all interval-region pairs*
  if current region is similar to given starting region $R1$, *then start a new* ### *trajectory*
  else if current region is similar to given ending region $R2$, *then finish current* ### *trajectory*
  else if current interval is adjacent to previous interval and current region is a neighbour of previous region, then add current region into trajectory
  else the trajectory failed; clear current trajectory and start a new search starting from the current node of list

6.2.4. Compound and conjunctive fuzzy spatio-temporal queries

For instance "*Retrieve all intervals in which object A is at top-left corner of the screen and object B is at the right of object A with a threshold degree of 0.5*" is a good example for such compound queries. In fact, there are two separate queries in this string. First one is "Find all intervals, in which object A appears at top-left corner of the screen", and the second one is "*Find all intervals, in which object B appears at right of object A with a threshold degree of 0.5*". Both of these queries give frame-type query result. We take an intersection of both results and the obtained set of intervals is the final result of the query. This kind of combinations may be generated to form several compound and conjunctive queries and the model proposed and the prototype implemented are suitable for such extensions. One can easily implement compound and conjunctive fuzzy spatio-temporal queries in our system by adding a simple interface.

7. Conclusions and future work

In this paper, we presented a spatio-temporal video data model. Our model represents the semantic content of video streams and allows users to model any kind of semantic data involving objects, events, and activities in the video. The model presented here identifies spatial properties of objects with rectangular areas (regions) resembling MBRs (minimum-bounding rectangles). It is possible to compute spatial relationships between two rectangular areas, hence the objects covered by those rectangles. We can handle spatial relations *left*, *right*,

top, *bottom*, *top-left*, *top-right*, *bottom-left*, *bottom-right*, as directional relations, and *overlaps*, *equal*, *inside*, *contain*, *touch*, and *disjoint* as topological relations. We also support querying the trajectory of an object given the starting and ending regions.

A prototype of the proposed model is implemented. The prototype includes a data entry interface that allows users to do a detailed frame-by-frame investigation on the video stream. It is also possible to see the association map of the current video graphically on the screen. The application stores data for more than one video stream in its database. There is a query interface to generate spatial and non-spatial queries on the videos.

One advantage of this study is that, the proposed model is flexible in both its design and implementation, so that it is easy to extend it with new spatial relations. New query types on object trajectories can be developed without much effort. One disadvantage of the system is that data must be entered manually by an authorized user. The data provided by the user may be subjective, which may cause different definitions for the same object. However, this disadvantage can be overcome by developing some standards.

As future work, the prototype can be improved to be a complete video database system. Other topological relationships such as near, cover, far, very far can be included without changing the proposed data model. In this paper we did not specifically focus on the implementation of compound and conjunctive queries. However, we believe that we have developed a base for these query types, although further work is necessary for an efficient interface and algorithms. As another future work one can investigate ways of developing a semi-automatic data entry facility with special techniques in raw video processing and image recognition. For instance, once an object is introduced to the system manually, the upcoming appearances of the object can be extracted if one could make use of image recognition techniques on video frames.

References

[1] S. Adali, K.S. Candan, S. Chen, K. Erol, V.S. Subrahmanian, The advanced video information system: data structures and query processing, Multimedia Systems 4 (1996) 172–186.
[2] J.F. Allen, Maintaining knowledge about temporal intervals, Communications of ACM 26 (1983) 832–843.
[3] R.S. Aygün, A. Yazici, Modeling and management of fuzzy information in multimedia database applications, Multimedia Tools and Applications, in press.
[4] M. Cobb, F. Petry, Modeling spatial data within a fuzzy framework, Journal of American Society of Information Science 49 (1998) 253–266.
[5] G. Davenport, T.A. Smith, N. Pincever, Cinematic primitives for multimedia, IEEE Computer Graphics and Applications (1991) 67–74.
[6] M.E. Donderler, O. Ulusoy, U. Gudukbay, A rule-based video database system architecture, Information Sciences 143 (2002) 13–45.

[7] S. Gibbs, C. Breiteneder, D. Tsichritzis, Data modeling of time-based media, Proceedings of the ACM SIGMOD Conference on Management of Data, Minneapolis, Minnesota, June 1994, pp. 91–102.
[8] H. Jiang, A.K. Elmagarmid, WVTDB—a semantic content based videotext database system on the world wide web, IEEE Transactions on Knowledge and Data Engineering 10 (6) (1998) 947–966.
[9] H. Jiang, A.K. Elmagarmid, Spatial and temporal content-based access to hypervideo databases, The VLDB Journal 7 (1998) 226–238.
[10] M. Koprulu, A Spatio-temporal Video Data Model, M.Sc. Thesis, Middle East Technical University, Department of Computer Engineering, 2001.
[11] T.C.T. Kuo, A.L.P. Chen, Content-based query processing for video databases, IEEE Transactions on Multimedia 2 (1) (2000) 1–13.
[12] J.Z. Li, M.T. Özsu, D. Szafron, Modeling of moving objects in a video database, Proceedings of IEEE International Conference on Multimedia Computing and Systems, Ottawa, Canada, June 1997, pp. 336–343.
[13] E. Oomoto, K. Tanaka, OVID: design and implementations of a video-object database system, IEEE Transactions on Knowledge and Data Engineering 5 (4) (1993) 629–643.
[14] D. Papadias, Y. Theodoridis, Spatial relations, minimum bounding rectangles and spatial data structures, International Journal of Geographical Information Science 11 (1997) 111–138.
[15] D. Papadias, N. Karacapilidis, D. Arkoumnais, Processing fuzzy spatial queries: a configuration similarity approach, International Journal of Geographical Information Science 13 (1999) 93–128.
[16] Y. Theodoridis, M. Vazirgiannis, T. Sellis, Spatio-temporal indexing for large multimedia applications, in: Proceedings of the IEEE International Conference on Multimedia Computing Systems, 1996, Hiroshima, Japan, IEEE Computer Society Press, Los Alamitos, CA, 1996, pp. 441–448.
[17] M. Vazirgiannis, Uncertainity handling in spatial relationships, ACM-SAC 2000 Proceedings, Como, Italy, pp. 215–221.
[18] R. Weiss, A. Duda, D. Gifford, Content-based access to algebraic video, in: Proceedings of the First IEEE International Conference on Multimedia Computing and Systems, 1994, Boston, MA, IEEE Computer Society Press, Los Alamitos, CA, 1994, pp. 140–153.
[19] A. Yazici et al., Handling complex and uncertain information in the ExIFO and NF^2 data model, IEEE Transactions on Fuzzy Systems 7 (6) (1999).