A RULE-BASED VIDEO DATABASE SYSTEM ARCHITECTURE[*]

Mehmet Emin Dönderler, Özgür Ulusoy and Uğur Güdükbay
Department of Computer Engineering, Bilkent University
Bilkent, 06533 Ankara, Turkey
e-mail: {mdonder, oulusoy, gudukbay}@cs.bilkent.edu.tr

Abstract

We propose a novel architecture for a video database system incorporating both spatio-temporal and semantic (annotation, event/activity and category-based) query facilities. The originality of our approach lies in the fact that we intend to provide full support for spatio-temporal, relative object-motion and similarity-based object-trajectory queries by a rule-based system utilizing a knowledge-base while using an object-relational database to answer semantic-based queries. Our method of extracting and modeling spatio-temporal relations is also a unique one such that we segment video clips into shots using spatial relationships between objects in video frames rather than applying a traditional scene detection algorithm. The technique we use is simple, yet novel and powerful in terms of effectiveness and user query satisfaction: video clips are segmented into shots whenever the current set of relations between objects changes and the video frames, where these changes occur, are chosen as key frames. The directional, topological and third-dimension relations used for shots are those of the key frames selected to represent the shots and this information is kept, along with frame numbers of the key frames, in a knowledge-base as Prolog facts. The system has a comprehensive set of inference rules to reduce the number of facts stored in the knowledge-base because a considerable number of facts, which otherwise would have to be stored explicitly, can be derived by rules with some extra effort.

Keywords: *content-based retrieval, spatio-temporal relations, video databases, knowledge representation, rule-based video modeling, inference rules, information systems.*

1 Introduction

There is an increasing demand toward multimedia technology in recent years. Especially, first image and later video databases have attracted a great deal of attention. One common property of image and video databases is the existence of spatial relationships between salient objects. Besides, video data has also a time dimension, and consequently, objects change their locations and relative positions with respect to each other in time. Because of this, we talk about spatio-temporal relationships instead of spatial or temporal relationships alone for video data.

This paper proposes a novel architecture for a video database system having both spatio-temporal and semantic (annotation, event/activity and category-based) query facilities. The

[*]This work is supported by the Scientific and Research Council of Turkey (TÜBİTAK) under Project Code 199E025.

architecture is original in that we intend to provide full support for spatio-temporal, relative object-motion and similarity-based object-trajectory queries by a rule-based system utilizing a knowledge-base while using an object-relational database so as to answer semantic-based queries. Our method of extracting and modeling spatio-temporal relations is also a unique one: video clips are segmented into shots with respect to the spatial relationships between objects in video frames in comparison to a conventional scene detection algorithm. We believe that our approach to use spatial relations for segmenting video clips into shots yields an intuitive and simple representation of video data. Moreover, it also provides more effective and precise answers to the user queries that involve objects' relative spatial positions in time dimension since stored facts are generated based upon objects' relative positions and key frames are detected when a change on the set of spatial relations occurs.

There is a very limited number of proposals in the literature that consider both spatial and temporal features of video objects in an integrated manner. Detailed discussion of some of the proposals related to our work and their comparison to our approach are provided in Section 2.

The contributions of this paper can be shortly stated as follows:

Video modeling and querying The proposed system uses a *rule-based* approach for modeling and querying spatio-temporal relations. It keeps in its knowledge-base only a relatively small number of relations as facts deriving the rest using a rule-based inference mechanism provided by Prolog. The query processor extracts spatio-temporal and semantic parts from queries and sends them to proper system components. Intermediate answers returned from these components are integrated seamlessly by the query processor to form final query results.

Video segmentation A novel approach is proposed for the segmentation of video clips based on the spatial relationships between salient objects in video data. Video clips are segmented into shots whenever the current set of relations between video salient objects changes, thereby helping us to determine parts of videos where the spatial relationships do not change at all. Nevertheless, we do not make any shot detection or do any image processing for the segmentation of video data: minimum bounding rectangles (MBRs) of the salient objects are manually specified before the relationship information is extracted from video frames. Apart from MBR specification, fact-extraction process for the directional and topological relations is carried out automatically.

Retrieval Granularity To the best of our knowledge, all the systems proposed in the literature so far associate video features with scenes that are defined to be the smallest logical units of video clips. However, our data model supports a finer granularity for query processing, which is independent of semantic segmentation of video clips: it allows users to retrieve any segment of a video clip, in addition to semantic video units, as a result of a query.

Directional relations To determine which directional relation holds between two objects, the center points of the objects' MBRs are used. Thus, directional relations can also be defined for overlapping objects as opposed to other works that are based on Allen's temporal interval algebra [2, 6, 12, 13].

Third-Dimension (3D) Relations Some additional relations were also defined on the third-dimension (z-axis of three dimensional space) and rules were implemented for them. However, they were not experimented due to the fact that they have not yet been fully incorporated into our fact-extraction tool. The 3D relations defined in the system are *infrontof*, *behind*, *strictlyinfrontof*, *strictlybehind*, *touchfrombehind*, *touchedfrombehind* and *samelevel*. We present the rules for these relations together with the rest of the spatio-temporal rules in Appendix A.

Currently, our video database system can answer a broad range of spatio-temporal queries using its knowledge-base. We intend to provide full support for relative object-motion, similarity-based object-trajectory and semantic queries as well. These features are under development.

The organization of this paper is as follows: Section 2 presents a brief introduction of the research done in literature regarding spatio-temporal relationships and their representation for modeling and querying of video data. Overall architecture of the proposed video database system and the rule-based approach to represent spatio-temporal relations between video salient objects are introduced in Section 3. Section 4 gives some example queries based on an imaginary soccer game through which our rule-based approach is demonstrated. Section 5 presents the results of our performance tests regarding the efficiency of the proposed system in terms of space and time criteria, and its scalability with respect to the number of salient objects per frame and the total number of frames in video. We make our conclusions and state our future work in Sections 6 and 7, respectively. Finally, the list of our inference rules is provided in Appendix A.

2 Related Work

There are not many studies that have appeared in the literature regarding the spatio-temporal modeling of video data. However, spatio-temporal relations between video objects constitute an integral part of the query types to be expected from users. Consequently, it is necessary for a video database system to employ efficient and effective spatio-temporal modeling and indexing of video data.

As mentioned in [14], there is a very limited number of proposals in the literature that take into account both spatial and temporal properties of video salient objects in an integrated manner. Some of the proposed index structures are *MR-tree*s and *RT-tree*s [17], *3D R-tree*s [15] and *HR-tree*s [10]. These structures are some adaptations of the well-known R-tree family. There are also quadtree based indexing structures, such as *Overlapping Linear Quadtree*s [16], proposed for spatio-temporal indexing.

*3D R-tree*s consider time as an extra dimension to the original two-dimensional space. Thus, objects represented by two-dimensional MBRs are now captured by three-dimensional minimum bounding boxes (MBBs). However, if this approach were to be used for moving objects, a lot of empty space would be introduced within objects' MBBs since the movement of an object is captured by using only one MBB. Thus, it is not a proper representation mechanism for video data, where objects frequently change their positions in time.

*RT-tree*s are proposed to solve this dead space problem by incorporating the time information by means of time intervals inside the R-tree structure. However, whenever an object changes its position, a new entry with temporal information must be inserted to the structure. This causes the generation of many entries that makes the *RT-tree* grow considerably. Furthermore, time information stored with nodes plays a complementary role and *RT-tree*s are not able to answer temporal queries such as "*find all objects that exist in the database within a given interval*".

*MR-tree*s and *HR-tree*s use the concept of overlapping B-trees [7]. They have separate index structures for each time point where a change occurs in an object position within the video data. It is space-efficient if the number of objects changing their locations is low because index structures may have some common paths for those objects that have not moved. Nevertheless, if the number of moving objects is large, they become inefficient. Detailed discussion of all these index structures can be found in [14].

All these approaches incorporate the MBR representation of spatial information within index structures. Thus, to answer spatio-temporal queries, directional and topological relations should be computed and checked for query satisfaction, which is a costly operation when performed during query processing.

Our rule-based approach to model spatio-temporal relations in video data eliminates the need for the computation of relations at the time of query processing, thereby cutting down the query response time considerably. In our approach, a key frame represents some consecutive frames in a video with no change in the set of spatial relations between video objects in the frames. Computed spatial relations for each key frame are stored to model and query video data for spatio-temporal relations.

Li et al. describe an effort somewhat similar to our approach, where some spatial relations are computed by associated methods of objects while others may be derived using a set of inference rules [6]. Nonetheless, the system introduced in [5, 6] does not explicitly store a set of spatio-temporal relations from which a complete set of relations between all objects can be derived by rules as we do, and consequently, relations which cannot be derived by rules are computed during query processing. Our approach of pre-computing and storing a set of relations that cannot be derived by the set of inference rules a priori to querying reduces the computational cost of queries considerably since there is no need at all to compute any spatio-temporal relation using any coordinate information at the time of query processing. All the relations that are not stored explicitly in the fact-base can be easily derived by the inference rules.

In [5], a new video model is introduced. In this model, there is no restriction on how videos are segmented. After the segmentation, shots are grouped in a hierarchy on the basis of the common video objects they contain, developing an index structure, called *CVOT*. However, employed as a common practice by all the systems proposed in the literature to the best of our knowledge, video features are associated with scenes that are defined to be the smallest logical units of videos. In our approach, spatio-temporal relations between video objects and object-trajectories are represented as facts in a knowledge-base, and they are not explicitly related to semantic units of videos. It is because users may also wish to see only the parts of a video, where the conditions given in a query are satisfied, rather than the scenes that contain these segments. Thus, our system returns precise answers for spatio-temporal queries in terms of frame intervals whereas this functionality is not implemented in *CVOT*.

In [9], a unified framework for characterizing multimedia information systems, which is built on top of the implementations of individual media, is proposed. Some of user queries may not be answered efficiently using these data structures; therefore, for each media-instance, some feature constraints are stored as a logic program. Nonetheless, temporal aspects and relations are not taken into account in the model and complex queries involving aggregate operations as well as uncertainty and time in queries require further work to be done. In addition, though the framework incorporates some feature constraints as facts to extend its query range, it does not provide a complete deductive system as we do. The authors extend their work defining feature-subfeature relationships in [8]. When a query cannot be answered, it is relaxed by substituting a subfeature for a feature. This relaxation technique provides some support for reasoning with uncertainty. In [1], a special kind of segment tree called *frame segment tree* and a set of arrays to represent objects, events, activities and their associations are introduced. The proposed model is based on the generic multimedia model described in [9]. The additional concepts here are activities, events and their associations with objects, thereby relating them to the frame sequences. The proposed data model and the algorithms for handling different types of queries were implemented within a prototype, called Advanced Video Information System (AVIS). However, the semantics of the temporal queries, such as "Find all the events Tom is involved in after he finds out that he has been admitted to a college" and those more complex than this one, are not addressed, which we plan to support using temporal operators to query events.

Sistla et al. propose a graph and automata based approach to find the minimal set of spatial relations between objects in a picture given a set of relations that is a superset of the minimal set [13, 12]. They provide algorithms to find the minimal set from a superset as well as to deduce all the relations possible from the minimal set itself for a picture. However, the authors restrict the directional relations to be defined only for disjoint objects as opposed to our approach where overlapping objects may also have directional relations. Moreover, the set of inference rules considered in their implementation is rather small compared to ours. They do not incorporate any 3D relation, either. Furthermore, our fact-extraction algorithm is simpler and it extracts spatio-temporal, appearance and trajectory properties of objects from a video even though we do not claim that it produces the minimal set of spatial relations in a video frame as they do for a picture.

A content-based logic video query language, *CVQL*, is proposed in [4]. Users retrieve video data specifying spatial and temporal relationships for salient objects. An elimination-based preprocessing for filtering unqualified videos and a behavior-based approach for video function evaluation are also introduced. For video evaluation, an index structure, called

M-index, is proposed. Using this index structure, frame sequences satisfying a query predicate can be retrieved. Nonetheless, topological relations between salient objects are not supported since an object is represented by a point in 2D space. Hence, the system cannot answer topological queries. Our system provides full support for spatio-temporal querying of video data.

3 The System Architecture and the Rule-based Approach

3.1 Overall System Architecture

Figure 1 illustrates the overall architecture of our target Web-based video database system. The proposed system is built on a client-server architecture. Users access the video database system over the Internet through a Java client Applet.

Users may query the system with sketches. A visual query is formed by a collection of objects with different attributes including relative object-motion, object-trajectory with similarity measure, spatio-temporal ordering of objects in time, annotations and events. Motion is sketched as an arbitrary polygonal trajectory with relative speed information for each query object. Annotations may also be used to query the system based on keywords. Users are able to browse the video collection before posing complex and specific queries. A text-based SQL-like query language is also available for experienced users.

Web clients communicate user queries, transformed to SQL-like text-based query language expressions if visual queries are given, to the query processor. Query processor is responsible for retrieving and responding queries. It first separates the keyword, activity/event and category-based query conditions in a query from those that could be answered by the knowledge-base. The former type of conditions is organized and sent as regular SQL queries to an object-relational database whereas the latter part is reconstructed as Prolog queries. Intermediate results returned by these two system components are integrated by the query processor and final results are sent to Web clients.

Raw video data and video data features are stored separately. The feature database contains semantic properties of videos used for keyword, activity/event and category-based queries on video data. These features are generated and maintained by a video annotator tool being developed as a Java application. The knowledge-base is used to answer spatio-temporal, object-appearance and object-trajectory queries and the facts-base is populated by the fact-extractor tool, which is a Java application as well.

3.2 Knowledge-base Structure

Rules have been extensively used in knowledge representation and reasoning. The reason why we employed a rule-based approach to model and query spatio-temporal relations between salient objects is that it is very space efficient: only a relatively small number of facts needs to be stored in the knowledge-base and the rest can be derived by the inference rules, which yields a

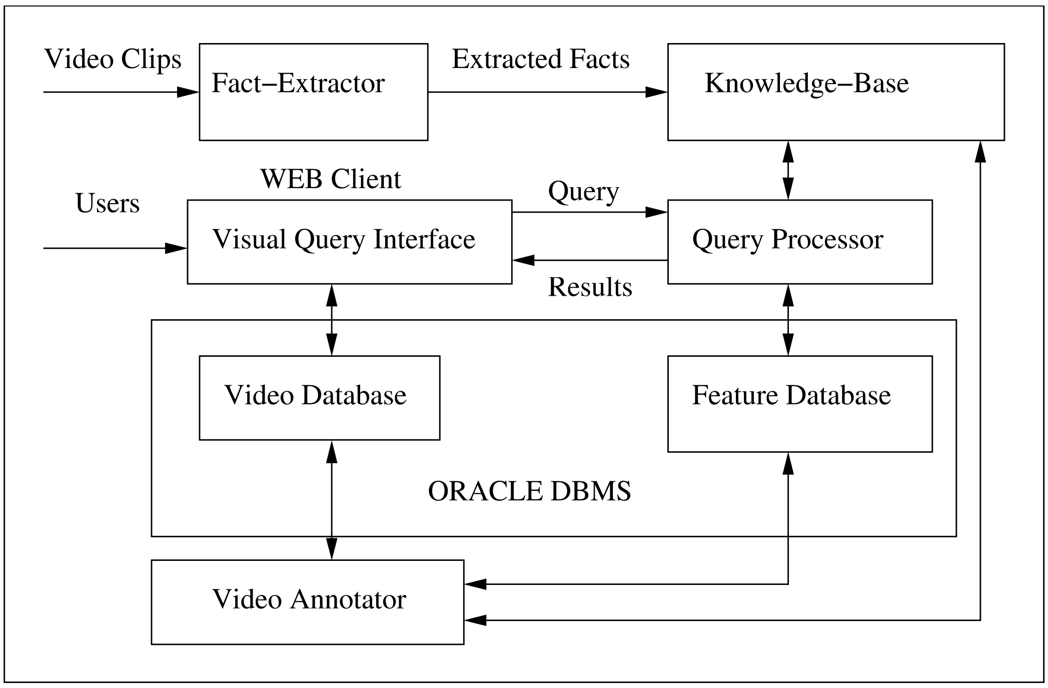


Figure 1: Overall Architecture of Our Target Video Database System

substantial improvement in storage space. Besides, our rule-based approach provides an easy-to-process and easy-to-understand structure for a video database system.

In the knowledge-base, each fact[1] has a single frame number, which is of a key frame. This representation scheme allows Prolog, our inference engine, to process spatio-temporal queries faster and easier than it would with frame intervals attached to the facts, because frame interval processing to form final query results can be carried out efficiently by some optimized code, written in C or C++, outside the Prolog environment. Therefore, the rules used for querying video data, which we call *query rules*, have frame-number variables as a component. A second set of rules that we call *extraction rules* was also created to work with frame intervals in order to extract spatio-temporal relations from video clips. Extracted spatio-temporal relations are converted to be stored as facts with frame numbers of the key frames attached in the knowledge-base and these facts are used by the query rules for query processing in the system. In short, spatio-temporal relations in video clips are stored as Prolog facts in the knowledge-base in a key-frame basis and the extraction rules are only used to extract the spatio-temporal relations from video data.

The reason of using a second set of rules with frame intervals to extract spatio-temporal relations is that it is much easier and more convenient to create the facts-base by first populating an initial facts-base with frame intervals and then converting this facts-base to the one with frame numbers of the key frames in comparison to directly creating the final facts-base in the process of fact-extraction. The main difficulty, if a second set of rules with frame intervals had not been used while extracting spatio-temporal relations, would be detecting the key frames of a

[1]Except for *appear* and *object-trajectory* facts, which have frame intervals as a component instead of frame numbers because of storage space, ease of processing and processing cost considerations.

video clip when processing it frame by frame at the same time. It is not a problem so far as the coding is concerned, but since the program creating the facts-base would perform this key frame detection operation for each frame, it would take whole a lot of time to process a video clip compared to our method.

In the knowledge-base, only are the basic facts stored, but not those that can be derived by rules according to our fact-extraction algorithm. Nonetheless, using a frame number instead of a frame interval introduces some space overhead because the number of facts increases due to the repetitions of some relations for each key frame over a frame interval. Nevertheless, it also greatly reduces the complexity of the rules and improves the overall query response time.

The algorithm developed for converting an initial facts-base to the one incorporated into the knowledge-base is very simple. It is based on the simple observation that key frames are the starting frames of the intervals associated with the facts in the initial facts-base. After the detection of the key frames, each fact with a frame interval is converted into a group of facts with frame numbers of the key frames. For example, if *west*(A, B, [*1*, *100*]) is a fact in the initial facts-base and 1, 10 and 50 are the key frames that fall into the frame interval range of [*1*, *100*], then, this fact is converted to the following facts in the knowledge-base: *west*(A, B, 1), *west*(A, B, 10) and *west*(A, B, 50).

In the system, facts are stored in terms of four directional relations, *west*, *south*, *south-west* and *north-west*, six topological relations, *cover*, *equal*, *inside*, *disjoint*, *touch* and *overlap*, and four 3D relations defined on the z-axis of three dimensional space, *infrontof*, *strictlyinfrontof*, *touchfrombehind* and *samelevel*, because the query rules are designed to work on these types of explicitly stored facts. However, there are also rules for *east*, *north*, *north-east*, *south-east*, *right*, *left*, *below*, *above*, *behind*, *strictlybehind*, *touchedfrombehind*, *contains* and *covered-by*. These rules do not work directly with the stored facts, but rather they are used to invoke related rules. For example, let's suppose that there is a relation stored as a fact for the pair of objects σ(A, B), such as *west*(A, B, 1), where A and B are object identifiers and 1 is the frame number of the relation. When a query "*east*(B, A, F)" is posed to the system, the rule *east* is used to call the rule *west* with the order of objects switched. That is, it is checked to see if *west*(A, B, F) can be satisfied. Since there is a fact *west*(A, B, 1) stored in the facts-base, the system returns 1 for F as the result of the query.

Above argument also holds for the extraction rules only this time for extracting relations from a video clip rather than working on stored facts. Therefore, the organization of the extraction rules is the same as that of the query rules.

Four types of inference rules, *strict directional*, *strict topological*, *heterogeneous directional and topological* and *3D relations*, were defined with respect to the relations' types in the rule body. For example, *directional rules* have only directional relations in their body whilst *heterogeneous rules* incorporate both directional and topological components. The complete listing of our inference rules is given in Appendix A.

In addition, some other facts, such as *object-trajectory* and *appear* facts, are also stored in the knowledge-base. These facts have frame intervals rather than frame numbers attached as

Relation	Inverse	Meaning
A infrontof B	B behind A	AAA BBB or AAABBB or AAA BBB and A overlap B
A strictlyinfrontof B	B strictlybehind A	AAA BBB or AAABBB or AAA BBB
A samelevel B	B samelevel A	AAA BBBBBB or AAAAAA BBB or AAA BBBBBB or AAA BBB
A touchfrombehind B	B touchedfrombehind A	BBBAAA

Table 1: Definitions of 3D relations

a component. *Appear* facts are used to derive some trivial facts, *equal*(A,A), *overlap*(A,A) and *samelevel*(A,A), as well as to answer object-appearance queries in video clips by rules. *Object-trajectory* facts are intended to be used for answering relative object-motion and similarity-based object-trajectory queries when the system is completed.

Table 1 presents semantic meanings of our 3D relations based on Allen's temporal interval algebra. The relations *behind*, *strictlybehind* and *touchedfrombehind* are inverses of *infrontof*, *strictlyinfrontof* and *touchfrombehind*, respectively. The inverse relation of *samelevel* is itself. The relations *infrontof* and *behind* require that objects overlap in 2D-space whereas *strictlyinfrontof* and *strictlybehind* do not imply this condition. Further information on directional and topological relations can be found in [3, 11].

3.3 Fact Extraction Algorithm

The algorithm for deciding what relations to store as facts in the knowledge-base is illustrated as a pseudo-code in Figure 2. In this algorithm, objects at each frame, κ, are ordered with respect to the center-point x-axis values of objects' MBRs. Index values of the objects are used as object labels after this sorting process. Then, objects are paired with respect to their labels starting with the object whose label is 0. The directional and topological relations are computed for each possible object pair whose first object's label is smaller than that of the second object and whose label distance is one. The *label distance* of an object pair is defined as the absolute numerical difference between the object labels. After exhausting all the pairs with the label distance one, the same operation is carried out for the pairs of objects whose label distance is two. This process is continued in the same manner and terminated when the distance reaches the number of objects in the frame.

Initially, the set of relations, η, is empty. All directional and topological relations are computed for each object pair as described above for the current frame being processed and the computed relations are put in the array λ in order. Then, for each relation in λ, starting with the first one indexed 0, it is checked to see if it is possible to derive the computed relation from the relations in η by the extraction rules. For example, for the first frame, if a relation cannot be derived from η using the rules, this relation is added to η with the frame interval [*1*, *1*]. Otherwise, it is ignored since it can be derived. For the consecutive frames, if a computed relation cannot be derived, an additional check is made to see whether there is such a relation in η that holds for a frame interval up to the current frame processed. If so, the frame interval of that relation is extended with the current frame by increasing the last component of its interval by one. Otherwise, the computed relation is added to η with the frame interval [*current frame*, *current frame*]. The set of relations obtained at the end contains the relations that must be stored as facts in the knowledge-base after conversion. The rest of the relations may be derived from these facts by rules.

For 3D relations, computation cannot be done automatically since 3D coordinates of the objects are unknown and cannot be extracted from video frames. Hence, these relations are entered manually for each object-pair of interest and those that can be derived by rules are eliminated automatically by the fact-extraction tool. The tool can perform an interactive conflict check for 3D relations and has some facilities to keep the existing set of 3D relations intact for the consecutive frames as well as to edit this set with error-and-conflict check on the current set for the purpose of easy generation of 3D relations. Generation of 3D relations is carried out for each frame of a video clip at the same time while the rest of the spatio-temporal relations are extracted. These 3D relations are then put in λ and they, along with the rest of the relations, are also used for key frame detection.

The initial fact-base, η, is also populated with the *appear* and *object-trajectory* facts. For each object, an *appear* fact is kept where it appears in video represented with a list of frame intervals. Furthermore, for each object, an *object-trajectory* fact is added for the entire video. These facts are copied to the final facts-base without any conversion. *Appear* facts are also used to detect key frames if an object appears when there is no object in the previous frame or if an object disappears while it is the only object in the previous frame.

Our approach greatly reduces the number of relations to be stored as facts in the knowledge-base, which also depends on some other factors as well, such as the number of salient objects, the frequency of change in spatial relations, and the relative spatial locations of the objects with respect to each other. Nevertheless, it is not claimed that the set of relations stored in the

```
1. Start with an empty set of facts, $\eta$.
2. Set $m$ to the number of frames in video
3. For ($currentFrame = 0$; $currentFrame < m$; $currentFrame++$)
4.     Begin
5.         Set $\kappa$ to be the object array of the current frame
6.         Sort $\kappa$ in ascending order on x-axis coordinates of object MBR center points
           (* Use object index values in the sorted array as object labels *)
7.         Set $n = |\kappa|$ (Number of objects in the frame)
8.         For ($i = 0$; $i < n$; $i++$)
9.             Begin
10.                If (there exist an object-appearance fact for $\kappa[i]$ in $\eta$)
11.                    Update this fact accordingly for currentFrame
12.                Else
13.                    Put a new object-appearance fact for $\kappa[i]$ in $\eta$
14.                If (there exist an object-trajectory fact for $\kappa[i]$ in $\eta$)
15.                    Update this fact accordingly for currentFrame
16.                Else
17.                    Put a new object-trajectory fact for $\kappa[i]$ in $\eta$
18.            EndFor
19.        Set $\lambda$ to be an empty array
20.        Set index to 0
21.        For ($labelDistance = 1$; $labelDistance < n$; $labelDistance++$)
22.            Begin
23.                For ($index1 = 1$; $index1 < n - labelDistance$; $index1++$)
24.                Begin
25.                    $index2 = index1 + labelDistance$
26.                    Find dirRelation($\kappa[index1]$, $\kappa[index2]$) and put it in $\lambda[index]$
27.                    Increment index by 1
28.                    Find topRelation($\kappa[index1]$, $\kappa[index2]$) and put it in $\lambda[index]$
29.                    Increment index by 1
30.                EndFor
31.            EndFor
32.        Put 3D relations in $\lambda$ incrementing index by 1 at each step
33.        Reorder $\lambda$ with respect to the dependency criteria among relations as follows:
                 * A relation with a smaller index value is placed before a relation of the same
                   type with a bigger index value
                 * The order of placement is (a), (b), (c), (d), (e), (f), (g) and (h)
                 a) {equal}, b) directional relations, c) {cover, touch}
                 d) {inside}, e) {overlap, disjoint}, f) {samelevel, touchfrombehind}
                 g) {infrontof}, h) {strictlyinfrontof}
34.        Update $\eta$ as follows: (* Facts-base population *)
35.              For ($i = 0$; $i < index$; $i++$)
36.              Begin
37.                  If ($\lambda[i]$ can be derived by extraction rules using the relations in $\eta$)
38.                      Skip and ignore the relation
39.                  Else
40.                      If ($\exists\beta \in \eta$ such that $\beta$ is the same as $\lambda[i]$ except for its frame
                            interval, whose ending frame is $currentFrame - 1$)
41.                          Extend the frame interval of $\beta$ by 1 to currentFrame
42.                      Else
43.                          Put $\lambda[i]$ in $\eta$ with the frame interval [currentFrame, currentFrame]
44.              EndFor
45.    EndFor
```

Figure 2: Fact-Extraction Algorithm

Order	Relation	Dependencies
1	equal	None
2	Directional Relations	equal
3	cover, touch	equal
4	inside	equal, cover
5	overlap, disjoint	equal, cover, touch, inside
6	samelevel, touchfrombehind	None
7	infrontof	touchfrombehind
8	strictlyinfrontof	samelevel, touchfrombehind, infrontof

Table 2: Dependencies Among Rules

knowledge-base is the minimal set of facts that must be stored because the number of facts to be stored depends on the labeling order of objects in our method and we use the x-axis ordering to reduce this number. Our heuristic in this approach is that if it is started with the pairs of objects whose label distance is smaller, most of the relations may not need to be stored as facts for the pairs of objects with a bigger label distance. The reason is that these relations might be derived from those already considered to be stored in the knowledge-base. In addition, since the spatial relations are ordered according to the dependency criteria given in Table 2 before deciding which relations to store in the facts-base, no dependent relation is stored just because a relation of different type it depends on has not been processed yet.

The fact-extraction process is semi-automatic: objects' MBRs are specified manually and 3D relations are entered by the user through graphical components. Users do not have to draw each MBR for consecutive frames because MBR resizing, moving and deletion facilities are provided for convenience. Moreover, the tool performs 3D-relation conflict check and eliminates the derivable 3D relations from the set as they are entered by the user. The set for 3D relations is also kept intact for subsequent frames so that the user can update it without having to reenter any relation that already exists in the set. Nevertheless, with this user intervention involved, it is not possible to make a complete complexity analysis of the algorithm. During our experience with the tool, it is observed that the time to populate a facts-base for a given video is dominated by the time spent interacting with the tool. However, since the fact extraction process is carried out offline, it does not have any influence on the system's performance. When the user intervention part is ignored, the complexity of our algorithm can be roughly stated as $O(mn^2)$ where m is the number of frames processed and n is the average number of objects per frame. It is a rough estimation because the facts-base is populated as frames are processed and it is not possible to guess the size of the facts-base or the number of each relation put in the set by type at any time during the process.

3.4 Directional Relation Computation

According to our definition, overlapping objects can also have directional relations associated except for the pairs of objects that are *equal* to each other, as opposed to the case where Allen's temporal interval algebra is used to define the directional relations.

In order to determine which directional relation holds between two objects, the center points of the objects' MBRs are used. Obviously, if the center points of the objects' MBRs are the same, then there is no directional relation between the two objects. Otherwise, the most intuitive directional relation is chosen with respect to the closeness of the line segment between the center points of the objects' MBRs to the eight directional line segments. For that, the origin of the directional system is placed at the center of the MBR of the object for which the relation is defined. In the example given in Figure 3, object A is to the *west* of object B because the center of object B's MBR is closer to the directional line segment *east* than the one for *south-east*. Moreover, these two objects overlap with each other, but a directional relation can still be defined for them. As a special case, if the center points of objects' MBRs fall exactly onto the middle of two directional segments, which one to be considered is decided as follows: the absolute distance of the objects' MBRs is computed on x and y axes with respect to the far-most vertex coordinates on the region, where the two directional line segments in question reside. If the distance in x-axis is greater, then the line segment that is closer to the x-axis is selected. Otherwise, the other one is chosen. Here, the objects' relative sizes and positions in 2D coordinate system implicitly play an important role in making the decision. Our approach to find the directional relations between two salient objects can be formally expressed as in Definitions 1 and 2.

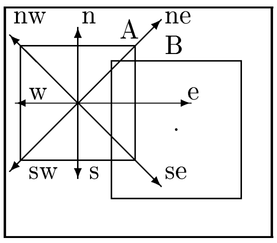
Definition 1 *The directional relation $\beta(A,B)$ is defined to be in the opposite direction to the directional line segment which originates from the center of object A's MBR, and is the closest to the center of object B's MBR.*

Definition 2 *The inverse of a directional relation $\beta(A,B)$, $\beta^{-1}(B,A)$, is the directional relation defined in the opposite direction.*

According to Definition 1, given two objects A and B, if the center of object B's MBR is closer to the directional line segment *east* in comparison to the others when the directional system's origin is at the center of object A's MBR, then the directional relation between objects A and B is *west*(A, B), where object A is the one for which the relation is defined. Thus, object A is to the *west* of object B. Using Description 2, it can be concluded that object B is to the *east* of object A. The rest of the directional relations can be determined in the same way.

4 Query Examples

This section provides some query examples based on an imaginary soccer game fragment between England's two teams *Arsenal* and *Liverpool*. These queries do not have any 3D-relation



west(A,B), east(B,A)

Figure 3: Directional Relation Computation

component. Nor do they contain any relative object-motion or similarity-based object-trajectory components because algorithms to process such queries are still under development. In the examples, the word "player(s)" is used for the member(s) of a soccer team except for the goalkeeper. Prolog query predicates and query results are only provided for the first query, Query 1.

Query 1 "Give the number of passes for each player of *Arsenal*".

Query: `pass X Y arsenal`, where X and Y are variables that stand for the players of *Arsenal* who give and take the passes, respectively.

Query Predicates:

```
pass(X, Y, T) :- fmember(X, T), fmember(Y, T), X \= Y,
           p-touch(X, ball, F1), p-inside(ball, field, F1),
           noother(X, ball, F1), p-touch(Y, ball, F2), F2 > F1,
           p-inside(ball, field, F2), noother(Y, ball, F2),
           fkframe(L, F1, F2), checklist(p-inside(ball, field), L),
           checklist(notouch(ball), L).

fmember(X, T) :- (getmembers(L, T), member(X, L), not(goalkeeper(X, T))).

noother(X, Y, F) :- findall(Z, p-touch(Z, Y, F), L),
                    forall(member(Z, L), Z = X).

fkframe(L, F1, F2) :- keyframes(K), findall(X, kframes(X, K, F1, F2), L).

keyframes([1, 10, 21, 25, 31, 35, 55, 61, 80, 91, 95, 101, 105, 111, 115,
           121, 125, 131, 135, 141, 150, 161, 165, 171, 172, 175, 181]).

kframes(X, L, F1, F2) :- member(X, L), X > F1, X < F2.

notouch(X, F) :- not(p-touch(Z, X, F)).

goalkeeper(X, T) :- getmembers(Y, T), last(X, Y).

getmembers(X, T) :- (T = arsenal, X = [dixon, keown, adams, winterburn,
           ljunberg, petit, vieira, overmars, kanu, bergkamp, seaman]);
                    (T = liverpool, X = [staunton, henchoz, hyypia, heggem,
           carragher, redknapp, hamann, smicer, owen, camara, westerveld]).
```

Explanation: It is assumed that if a player touches the ball alone, it is in his control. Consequently, if a player of *Arsenal* touches the ball for some time and then transfers the control of it to another player of his team, this event is considered as a pass from that player to another one in his team. Moreover, the ball should not be played (touched) by anyone else and it should also stay inside the field during this event.

The result of this query is:

```
Player:keown Passes(given):1
Player:adams Passes(given):2
Player:kanu Passes(given):1
Player:bergkamp Passes(given):1

Team:arsenal Total Passes:5
```

Query 2 "Give the number of shoots to the goalkeeper of the opponent team for each player of *Arsenal*".

Query: `shoot X arsenal`, where X is a variable that stands for the players of *Arsenal* who shoot.

Explanation: In this query, we are interested in finding the number of shoots to the goalkeeper of *Liverpool* by each player of *Arsenal*. In order to answer this query, the facts of *touch* to the ball are found for each player of *Arsenal*. For each fact found, it is also checked if there is a fact of *touch* to the ball for the opponent team's goalkeeper, whose frame number is bigger. Then, a check is made to see if there is no other *touch* to the ball between these two events and also if the ball is inside the field during the entire period. If all above conditions are satisfied, this is considered a shoot to the goalkeeper. Then, all such occasions are counted to find the number of shoots to the goalkeeper by each player of *Arsenal*.

Query 3 "Give the average ball control (play) time in frames for each player of *Arsenal*".

Query: `hold X arsenal`, where X is a variable that stands for the players of *Arsenal* who play with the ball.

Explanation: As it is assumed that when a player touches the ball alone, it is in his control, the ball control time for a player is computed with respect to the frame intervals during which he is in touch with the ball. Therefore, the following operation is performed for each player of *Arsenal* so as to answer this query: frame intervals during which a player touches the ball are found and the number of frames in the intervals are summed up. Divided by the number of frame intervals found, this gives for the player the average ball control time in frames. Since in a soccer game, a player may touch the ball outside the field as well, only are the frame intervals when the ball is inside the field considered. It is also possible to give the time information in seconds provided that the frame rate of the video is known.

Original Video	Total # of Frames	Total # of Objects	Max. # of Objects in a Frame
Jornal.mpg	5254	21	4
Smurfs.avi	4185	13	6

Table 3: Specifications of the movie fragments

Query 4 "Give the number of ball losses to the opponent team's players for *Adams* of *Arsenal*".

Query: `loss adams arsenal`.

Explanation: if *Adams* of *Arsenal* touches the ball for some time and then the control of the ball goes to a player of the opponent team, this event is considered as a ball loss from *Adams* to an opponent player. Furthermore, the ball should not be played (touched) by anyone else and it should stay inside the field during this event.

Query 5 "Give the number of kicks to outside field for *Adams* of *Arsenal*".

Query: `outside adams arsenal`.

Explanation: First, the key frames when *Adams* of *Arsenal* is in touch with the ball while the ball is inside the field are found. Then, for each key frame found, a fact with a bigger frame number, representing the ball being outside the field, is searched. If there is no *touch* to the ball between these two events, then this is a kick outside the field. All such occasions are counted to find the number of kicks outside the field by *Adams*.

5 Performance and Scalability Experiments

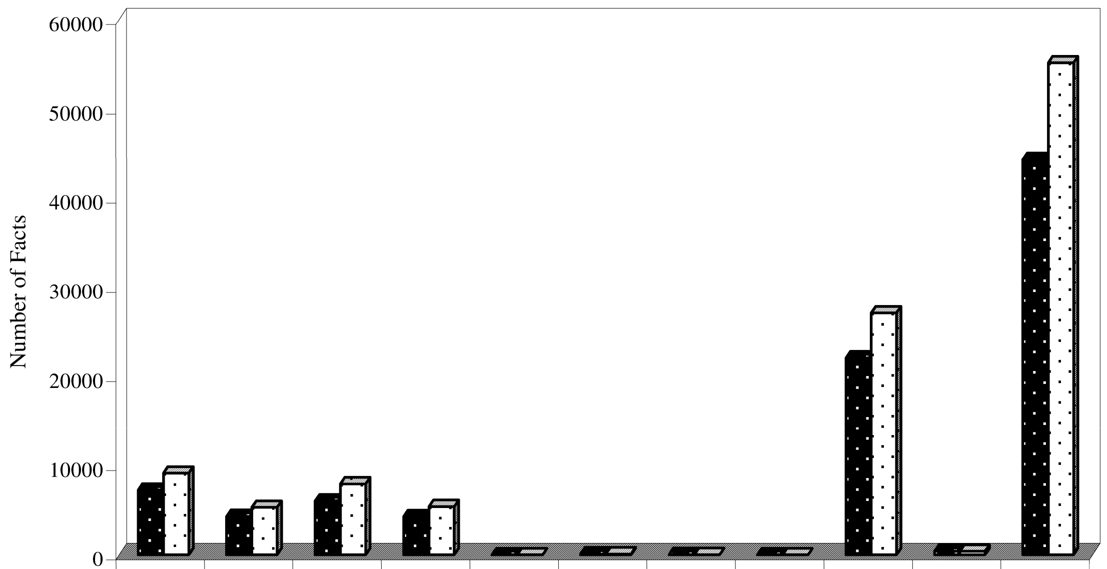
In order to show that our video database system is scalable in terms of the number of salient objects per frame and the total number of frames in a video clip as well as to demonstrate the space savings due to our rule-based approach, some program-generated synthetic video data was used. These tests constitute the first part of the overall tests. In the second part, the system's performance was tested on some real video clip fragments with the consideration of space and time efficiency criteria to show its applicability in real-life applications. The real video clip fragments were extracted from *jornal.mpg*, MPEG-7 Test Data set CD-14, Port. news, and a *Smurfs* cartoon episode named *Bigmouth*'s *Friend*. Table 3 presents some information about these video fragments.

In order to make a judgement on how successful our fact-extraction algorithm is in eliminating the redundant facts for both synthetic and real video data, two facts-bases were created. The first facts-base consists only of the basic facts extracted by the fact-extraction algorithm while the second one comprises all the facts computed again with this algorithm, but this time with its fact-reduction feature turned off. Since the two facts-bases were created using the same video data for synthetic and real video separately, the sizes of the resultant facts-bases give us an idea about how well our fact-reduction feature works as well as how efficient our approach is for space considerations.

5.1 Tests with Program-Generated Video Data

For the space efficiency tests, the number of objects per frame was selected as 8, 15 and 25 while the total number of frames was fixed to 100. To show the system's scalability in terms of the number of objects per frame, the total number of frames was chosen to be 100 and the number of objects per frame was varied from 4 to 25. For the scalability test with respect to the total number of frames, the number of objects was fixed to 8 whilst the total number of frames was varied from 100 to 1000.

Figures 4-6 give the space efficiency test results as bar charts for 8, 15 and 25 objects per frame for a 1000-frame synthetic video data. In these figures, *Facts-base 1* is the facts-base with redundant facts eliminated whereas *Facts-base 2* is the other facts-base that contains all the relations computed by the fact-extraction algorithm with its fact-reduction feature turned off. The numbers corresponding to these fields present the number of facts stored for each relation separately and in total for all relations in respective facts-bases.

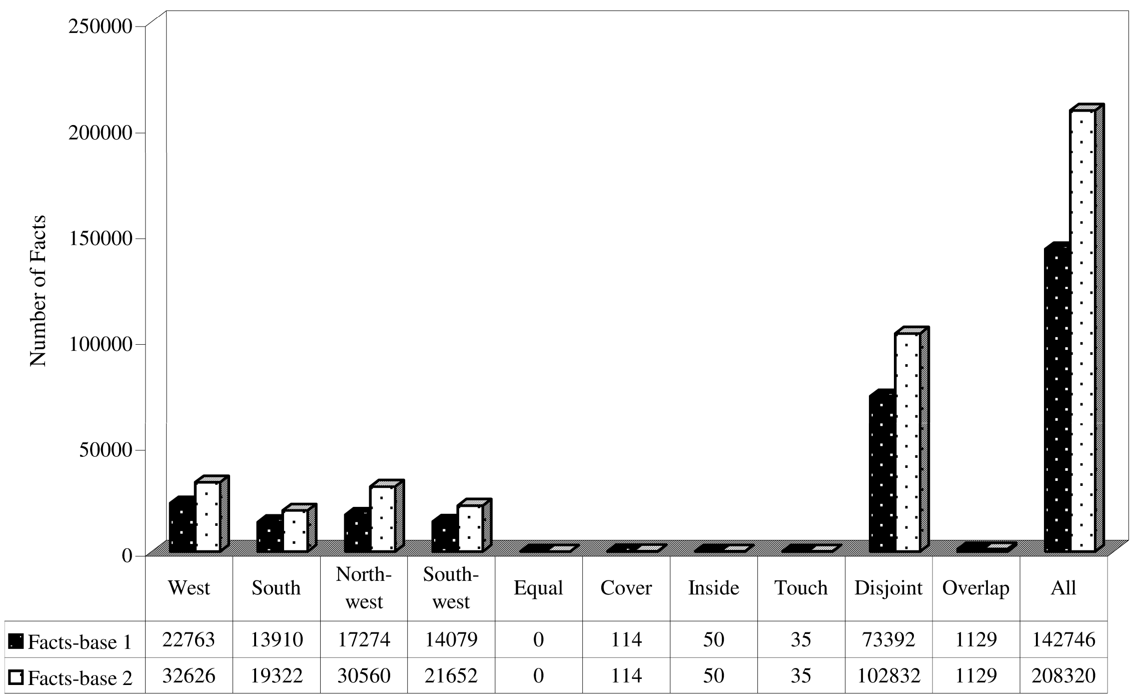


	West	South	North-west	South-west	Equal	Cover	Inside	Touch	Disjoint	Overlap	All
■ Facts-base 1	7240	4290	6018	4257	0	79	16	1	22068	383	44352
□ Facts-base 2	9141	5300	7944	5391	0	79	16	1	27101	383	55160

Figure 4: Space Efficiency Test Results (8 Objects and 1000 Frames)

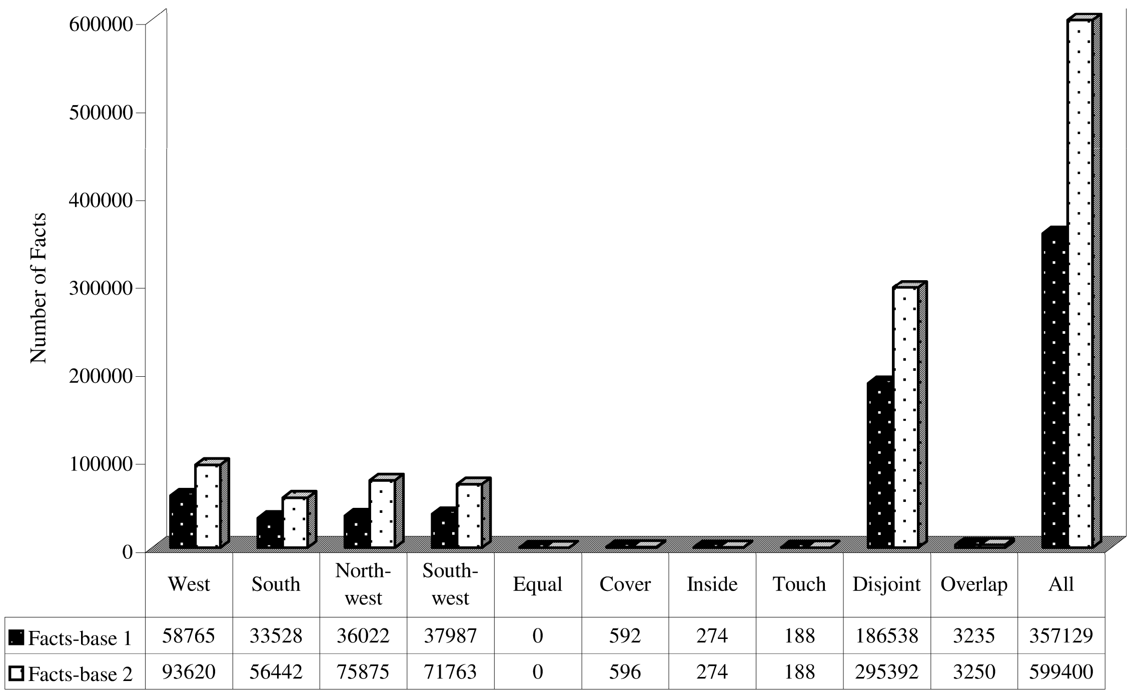
Four different types of queries were used for the scalability tests by taking four possible combinations of object-variable unifications that can be used to query the system. The queries are based on the *west* and *disjoint* relations and they are given in Table 4.

In the first part of the tests, where the system's scalability in terms of the number of objects per frame was checked, 1 and 7 were used in queries as object identifiers while for the second part, in which the system was tested for its scalability on the total number of frames, 1 and 0 were selected as object identifiers. In our test data, positive integer identifiers were used



	West	South	North-west	South-west	Equal	Cover	Inside	Touch	Disjoint	Overlap	All
■ Facts-base 1	22763	13910	17274	14079	0	114	50	35	73392	1129	142746
□ Facts-base 2	32626	19322	30560	21652	0	114	50	35	102832	1129	208320

Figure 5: Space Efficiency Test Results (15 Objects and 1000 Frames)



	West	South	North-west	South-west	Equal	Cover	Inside	Touch	Disjoint	Overlap	All
■ Facts-base 1	58765	33528	36022	37987	0	592	274	188	186538	3235	357129
□ Facts-base 2	93620	56442	75875	71763	0	596	274	188	295392	3250	599400

Figure 6: Space Efficiency Test Results (25 Objects and 1000 Frames)

X	Y	Query Format
Not Unified	Not Unified	west(X, Y, F), disjoint(X, Y, F)
Unified	Not Unified	west(1, Y, F), disjoint(1, Y, F)
Not Unified	Unified	west(X, 7/0, F), disjoint(X, 7/0, F)
Unified	Unified	west(1, 7/0, F), disjoint(1, 7/0, F)

Table 4: Queries for the Scalability Tests

for each object for the sake of simplicity, but in real video, salient objects are annotated by some meaningful textual names they can be remembered with. In our tests, each query returns non-empty results. Figures 7-14 provide the graphs obtained from the tests.

5.2 Tests with Real Video Data

We present our space efficiency test results as bar charts in Figures 15 and 16 for the video fragments taken from *jornal.mpg* and *smurfs.avi*, respectively. For the time efficiency tests, four queries were used for each of the video fragments. The queries used on the news report video fragment are as follows:

Query 1: Show the fragments of the clip where *priest*, *interviewee2* and *interviewee3* appear together, and also *interviewee2* is to the left of *interviewee3*.

Query 2: Show the fragments of the clip where *reporter1* and *reporter2* appear together with *priest* who is in his *car*.

Query 3: Show the fragments of the clip where *man3* is to the west of *man4* who is to the east of *woman2*.

Query 4: Show the longest possible fragments of the clip where *man6* is first to the left of *man5*, and later he becomes to the right of *man5*.

The first query is a directional-appearance query on salient objects *priest*, *interviewee2* and *interviewee3*. Query 2 is a topological-appearance query and Query 3 is a directional query. The last query, Query 4, is a motion query based on directional relations between salient objects *man5* and *man6*.

The second query assumes that if a person is inside a car or covered-by a car, then he/she is in that car. This assumption may not be correct depending on the camera view, but yet, it could be handled easily using the 3D relation *samelevel*. Nonetheless, since our tests are based on 2D spatial relations, no 3D relation is considered even though the system has a set of 3D inference rules as well. The results obtained for the queries are given in Table 5. The queries posed on the *Smurfs* video fragment are as follows:

Query 1: Give the parts of the video clip where *bigmouth* is below *robotsmurf* while *robotsmurf* starts moving from *bigmouth*'s left to his right and then goes from his right to his left

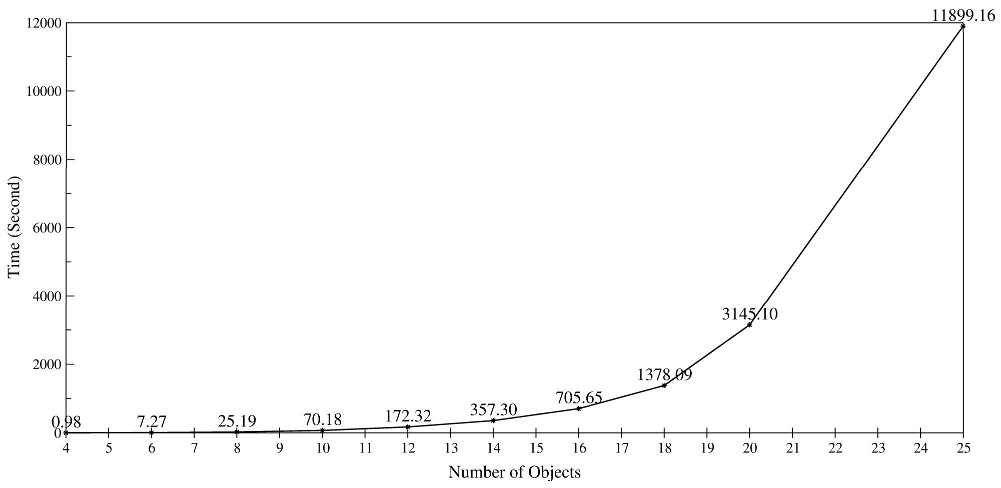


Figure 7: Query 1: west(X, Y, F), disjoint(X, Y, F) (100 Frames)

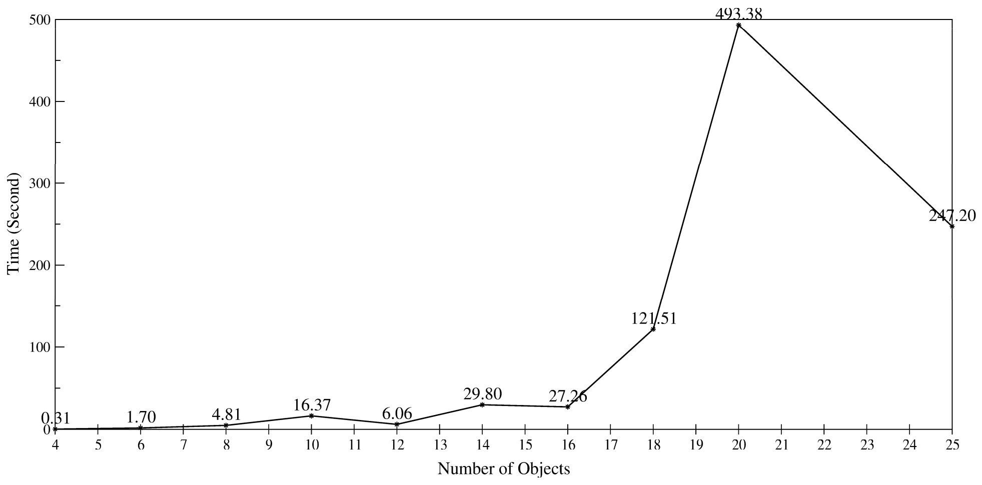


Figure 8: Query 2: west(1, Y, F), disjoint(1, Y, F) (100 Frames)

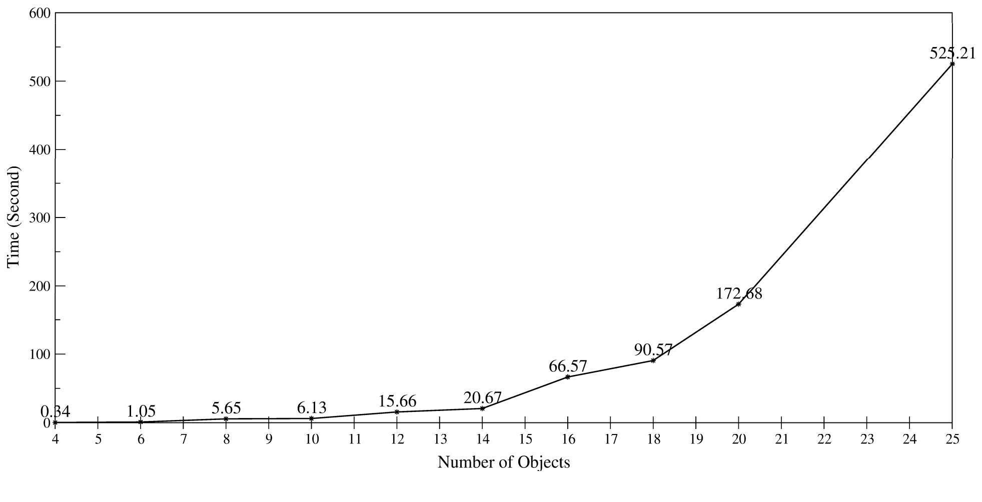


Figure 9: Query 3: west(X, 7, F), disjoint(X, 7, F) (100 Frames)

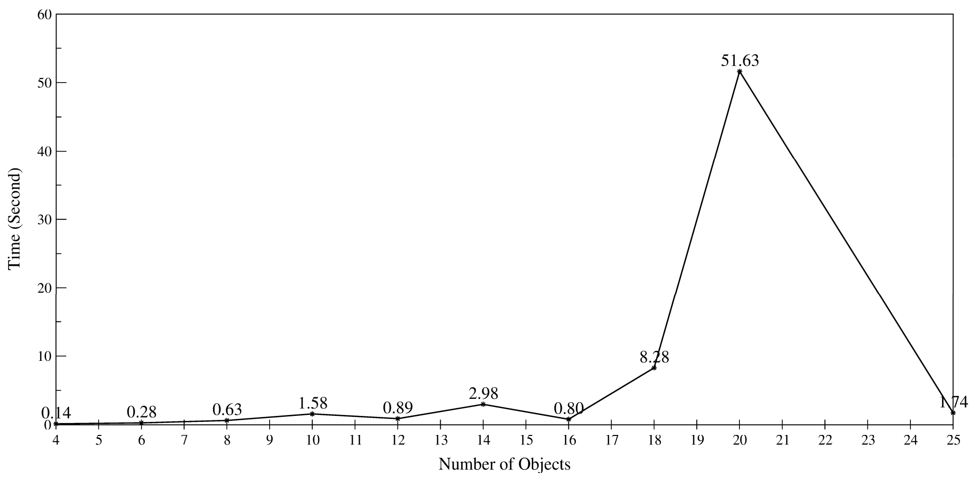


Figure 10: Query 4: west(1, 7, F), disjoint(1, 7, F) (100 Frames)

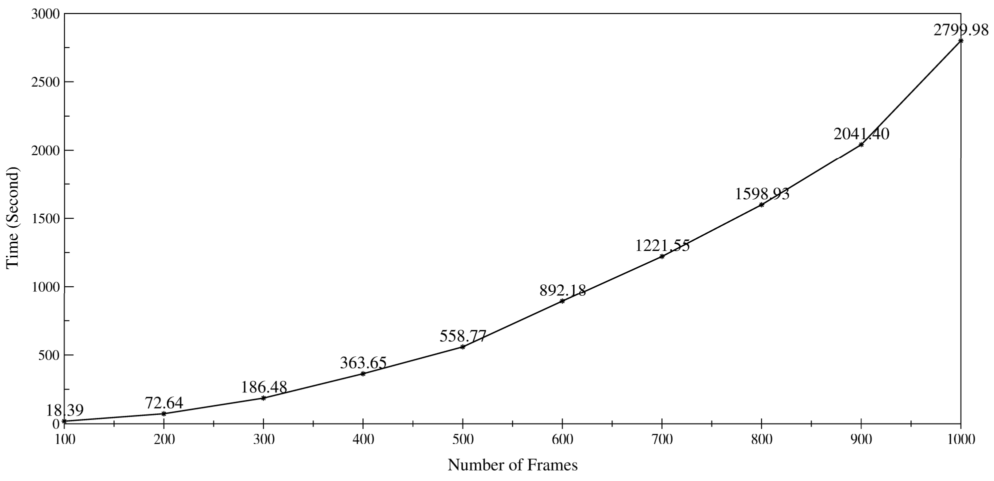


Figure 11: Query 5: west(X, Y, F), disjoint(X, Y, F) (8 Objects)

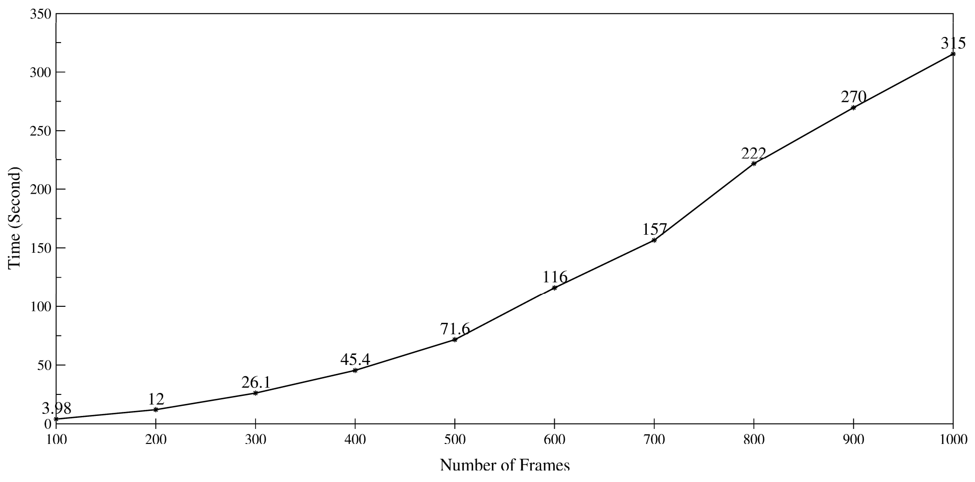


Figure 12: Query 6: west(1, Y, F), disjoint(1, Y, F) (8 Objects)

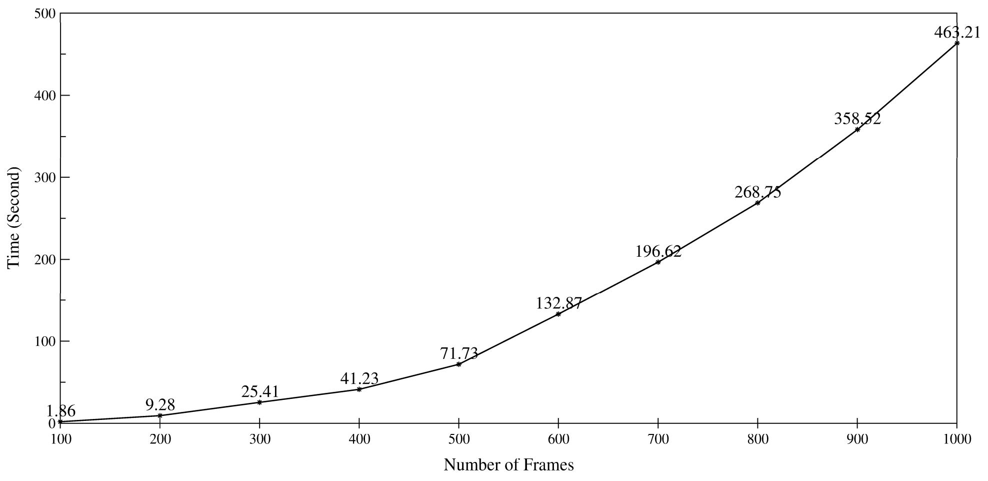


Figure 13: Query 7: west(X, 0, F), disjoint(X, 0, F) (8 Objects)

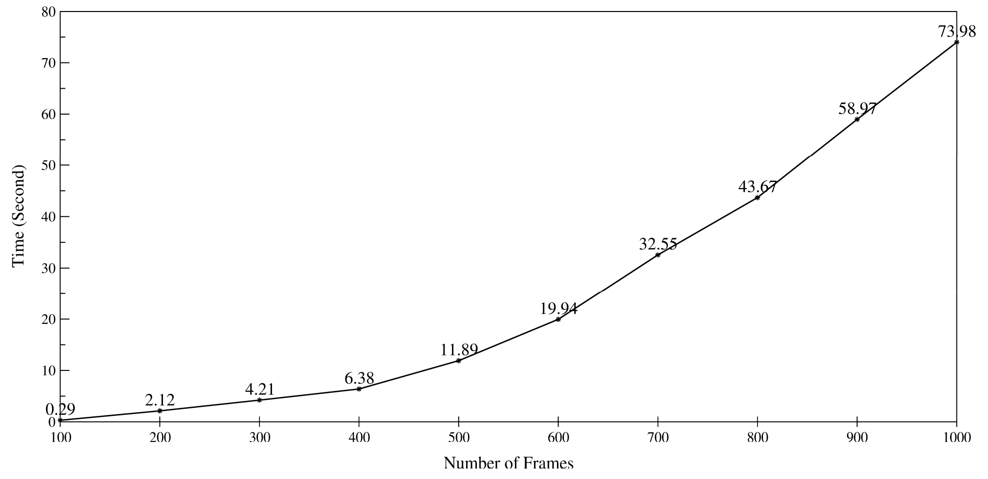
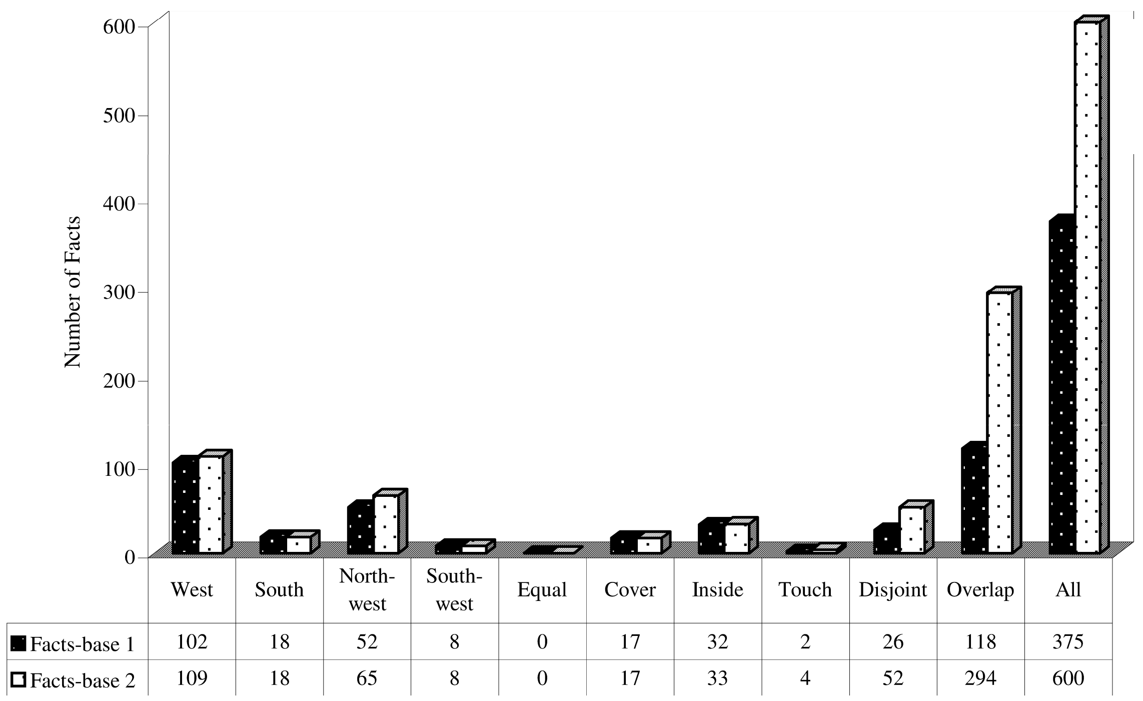
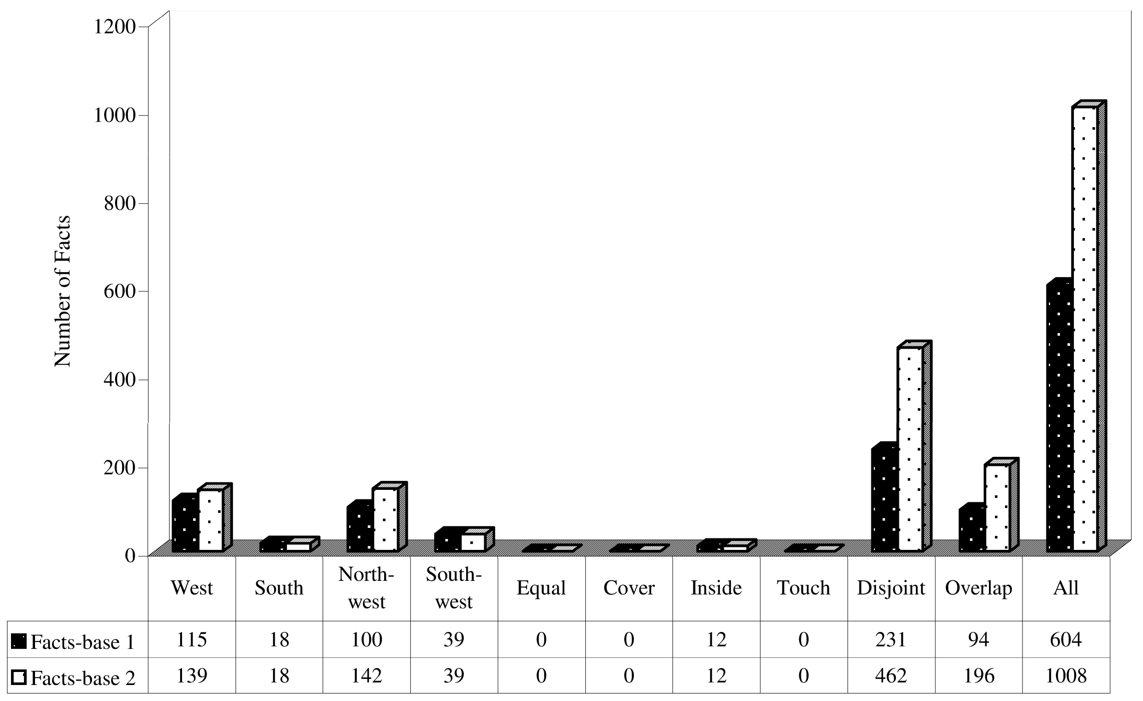


Figure 14: Query 8: west(1, 0, F), disjoint(1, 0, F) (8 Objects)



	West	South	North-west	South-west	Equal	Cover	Inside	Touch	Disjoint	Overlap	All
■ Facts-base 1	102	18	52	8	0	17	32	2	26	118	375
□ Facts-base 2	109	18	65	8	0	17	33	4	52	294	600

Figure 15: Space Efficiency Test Results for jornal.mpg



	West	South	North-west	South-west	Equal	Cover	Inside	Touch	Disjoint	Overlap	All
■ Facts-base 1	115	18	100	39	0	0	12	0	231	94	604
□ Facts-base 2	139	18	142	39	0	0	12	0	462	196	1008

Figure 16: Space Efficiency Test Results for smurfs.avi

Query #	Reduced Set (Sec.)
Query1	0.04
Query2	0.03
Query3	0.01
Query4	0.02

Table 5: Time Efficiency Test Results for jornal.mpg

Query #	Reduced Set (Sec.)
Query1	0.13
Query2	0.03
Query3	0.01
Query4	0.03

Table 6: Time Efficiency Test Results for smurfs.avi

repeating this as many times as it happens in the video fragment.

Query 2: Give the parts of the video clip where *Gargamel* is to the southwest of his *father* and *boyking*, who is between *soldier1* and *soldier2* (to his left and his right) and is in some distance with *Gargamel* and his *father*.

Query 3: Give the parts of the video clip where *lazysmurf*, *farmersmurf*, *grouchysmurf*, *smurfette* and *handysmurf* all appear together such that *lazysmurf* is to the west of *handysmurf* and *smurfette* is to the east of *farmersmurf*.

Query 4: Give the parts of the video clip where *robotsmurf* and *bigmouth* are close to each other (not disjoint) and *robotsmurf* is to the right of *bigmouth*, and there is no other object of interest that appears.

Query 1 is a directional-motion query while Query 2 is a directional-topological query. The third query is a directional-appearance query and the last one is a directional-topological-appearance query. In Query 1, we are interested in finding the largest sequences of frames in the fragment repeating the motion condition stated as many times as possible sequentially in the video. Thus, the final answer returned by the system is the set of video frame intervals where this motion is repeated as many times as possible sequentially in the video. For Query 2, it is concluded that two objects are in some distance and not close to each other if their MBRs are disjoint. If the objects are close to each other, then it is decided that their MBRs are not disjoint as in Query 4. These assumptions are only our semantic definitions of being two objects close to each other or in a distance. Therefore, these queries are partially based on these semantic definitions. Table 5 shows the results obtained for the queries.

In the tests conducted with program-generated video data, there is a 19.59% savings from the space for the sample data of 8 objects and 1000 frames. The space savings for the sample video of 15 objects and 1000 frames is 31.47% while it is 40.42% for 25 objects and 1000 frames. With real data, for the first video fragment *jornal.mpg*, our rule-based approach provides a savings of 37.5% from the space. The space savings for the other fragment, *smurfs.avi*, is 40%.

The space savings obtained from the program-generated video data is relatively low compared to that obtained from the real video fragments. We believe that the reason behind such a behavior is due to the random simulation of the motion of objects in our synthetic test data: while creating the synthetic video data, the motion pattern of objects was simulated randomly changing objects' MBR coordinates by choosing only one object to move at each frame. However, in real video, objects generally move slower causing the set of spatial relations to change over a longer period of frames. During the tests with the synthetic video data, it is also observed that space savings do not change when the number of frames is increased while the number of objects of interest per frame is fixed. The test results obtained for the synthetic data comply with those obtained for the real video. Some differences seen in the results are due to the fact that synthetic data was produced by a program, therefore not being able to perfectly simulate a real-life scenario.

The results plotted in Figures 7-14 show that the system is scalable in terms of the number of objects and the number of frames when either of these numbers is increased while the other is fixed. The time value deviations in some graphs are due to the data sets that had to be created separately for each object set, thereby each set having possibly different facts.

The results obtained from the time efficiency tests on real video data show that the system has a reasonable response time, which is a small fraction of a second. Therefore, we can claim that our system is reasonably fast enough for spatio-temporal user queries.

6 Conclusions

A novel architecture has been proposed for a video database system incorporating both spatio-temporal and semantic (annotation, event/activity and category-based) query facilities. The proposed system handles spatio-temporal queries using a knowledge-base, which consists of a fact-base and a comprehensive set of rules implemented in Prolog, while the semantic part is handled by an object-relational database. Intermediate query results returned from these two system components are integrated seamlessly by the query processor and sent to the Web clients.

Our approach to represent spatio-temporal relations in video data is unique in that we segment video clips into shots using spatial relationships between objects in video frames rather than applying a traditional scene detection algorithm. This technique is simple, yet novel and powerful in terms of effectiveness and user query satisfaction: video clips are segmented into shots whenever the current set of relations between objects changes, and the video frames, where these changes occur, are chosen as key frames. The directional, topological and 3D relations used for shots are those of the key frames that have been selected to represent the shots, and this information is kept, along with frame numbers of the key frames, in a knowledge-base as Prolog facts. The set of rules in the knowledge-base considerably reduces the number of facts that need to be stored for spatio-temporal querying of video data while also keeping the system's response time in a reasonable value as the test results show.

7 Future Work

We have been working on designing an SQL-like text-based video query language as well as a graphical user interface to support visual query facilities in the system. Currently, queries are given to the system as Prolog predicates. Users will be able to formulate spatial, temporal, spatio-temporal, relative object-motion, similarity-based object-trajectory and object-appearance queries as well as keyword, activity/event and category-based queries on video data using the visual query interface or the text-based video query language. A query will first be processed by an interpreter, which will separate the keyword, activity/event and category-based query conditions from those that could be answered using the knowledge-base. The former type of conditions will be organized and sent as regular SQL queries to an object-relational database and the latter part will be reformulated as Prolog queries. Intermediate results returned by the two system components will be integrated and final results will be transferred to the Web clients.

We have also been working on enhancing the system with capabilities to provide full support for relative object-motion and similarity-based object-trajectory queries. The system, when completed, will be a Web-based video database system. Users will be able to query the system using animated sketches. A query scene will be formed as a collection of objects with different attributes. Attributes will include relative object-motion, object-trajectory with similarity measure, spatio-temporal ordering of objects in time, annotations and event specification. Motion will be specified as an arbitrary polygonal trajectory with relative speed information for each query object. Annotations will be used to query the system based on keywords. There will also be a category grouping of video clips in the database so that a user is able to browse the video collection before actually posing a query.

References

[1] S. Adalı, K.S. Candan, S. Chen, K. Erol, and V.S. Subrahmanian. Advanced video information system: Data structures and query processing. *ACM-Springer Multimedia Systems Journal*, 4:172–186, August 1996.

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

[3] M. Egenhofer and R. Franzosa. Point-set spatial relations. *Int'l Journal of Geographical Information Systems*, 5(2):161–174, 1991.

[4] T.C.T. Kuo and A.L.P. Chen. Content-based query processing for video databases. *IEEE Transactions on Multimedia*, 1(2), March 2000.

[5] J.Z. Li, M.T. Özsu, and D. Szafron. Modeling of video temporal relationships in an object database management system. In *Proc. of Multimedia Computing and Networking*, pages 80–91, San Jose, CA, USA, February 1996.

[6] J.Z. Li, M.T. Özsu, and D. Szafron. Spatial reasoning rules in multimedia management systems. In *Proc. of International Conference on Multimedia Modeling*, pages 119–133, Toulouse, France, November 1996.

[7] Y. Manolopoulos and G. Kapetanakis. Overlapping B+ trees for temporal data. In *Proceedings of the 5th Jerusalem Conference on Information Technology (JCIT)*, 1990.

[8] S. Marcus and V.S. Subrahmanian. Foundations of multimedia information systems. *JACM*, 43(3), May 1996.

[9] S. Markus and V.S. Subrahmanian. *Multimedia Database System: Issues and Research Directions (eds. V.S. Subrahmanian and S. Jajodia)*, chapter Towards a Theory of Multimedia Database Systems, pages 1–35. Springer-Verlag, 1996.

[10] M.A. Nascimento and J.R.O. Silva. Towards historical R-trees. In *Proceedings of ACM Symposium on Applied Computing (ACM-SAC)*, 1998.

[11] D. Papadias, Y. Theodoridis, T. Sellis, and M. Egenhofer. Topological relations in the world of minimum bounding rectangles: A study with R-trees. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 92–103, San Jose, CA, USA, 1996.

[12] A.P. Sistla and C. Yu. Retrieval of pictures using a similarity-based approach employing indices on spatial relationships. In *Proc. of the 21st VLDB Conference*, Zurich, Switzerland, September 1995.

[13] A.P. Sistla and C. Yu. Reasoning about qualitative spatial relationships. *Journal of Automated Reasoning*, 25(4):291–328, November 2000.

[14] Y. Theodoridis, J.R.O. Silva, and M.A. Nascimento. On the generation of spatio-temporal datasets. In *Proceedings of the 6th Int'l Symposium on Large Spatial Databases (SSD)*, LNCS Series, Hong Kong, China, July 1999. Springer-Verlag.

[15] Y. Theodoridis, M. Vazirgiannis, and T. Sellis. Spatio-temporal indexing for large multimedia applications. In *Proceedings of the 3rd IEEE Conference on Multimedia Computing and Systems (ICMCS)*, 1996.

[16] T. Tzouramanis, M. Vassilakopoulos, and Y. Manolopoulos. Overlapping linear quadtrees: A spatio-temporal access method. In *Proceedings of the 6th Int'l ACM Workshop on Geographical Information Systems (ACM-GIS)*, 1998.

[17] X. Xu, J. Han, and W. Lu. RT-tree: An improved R-tree index structure for spatio-temporal databases. In *Proceedings of the 4th International Symposium on Spatial Data Handling (SDH)*, 1990.

A List of Inference Rules

In this appendix, we give our set of *strict directional*, *strict topological*, *heterogeneous directional and topological* and *3D* rules. In defining the rules, the following terminology has been adopted: if the relation r_1 implies the relation r_2, $r_1 \Longrightarrow r_2$ is used. Moreover, if $r_1 \Longrightarrow r_2$ and $r_2 \Longrightarrow r_1$, it is demonstrated by $r_1 \Longleftrightarrow r_2$. In addition, there is also a rule set for *appear*, which is used to derive trivial facts, *equal*(A,A), *overlap*(A,A) and *samelevel*(A,A) , as well as to answer object-appearance queries in video clips. This set is given below:

appear(A) $\Longleftrightarrow$ *equal*(A,A)
appear(A) $\Longleftrightarrow$ *overlap*(A,A)
appear(A) $\Longleftrightarrow$ *samelevel*(A,A)

A.1 Strict Directional Rules

Rule Set 1.1 (Inverse Property) The relations *west*, *north*, *north-west*, *north-east*, *right* and *above* are inverses of *east*, *south*, *south-east*, *south-west*, *left* and *below*, respectively.

a) *west*(A,B) $\Longleftrightarrow$ *east*(B,A)
b) *north*(A,B) $\Longleftrightarrow$ *south*(B,A)
c) *north-west*(A,B) $\Longleftrightarrow$ *south-east*(B,A)
d) *north-east*(A,B) $\Longleftrightarrow$ *south-west*(B,A)
e) *right*(A,B) $\Longleftrightarrow$ *left*(B,A)
f) *above*(A,B) $\Longleftrightarrow$ *below*(B,A)

Rule Set 1.2 (Transitivity) If $\beta \in$ S, where S is the set of directional relations, then
β(A,B) $\wedge$ β(B,C) $\Longrightarrow$ β(A,C).

Rule Set 1.3 The relations *right*, *left*, *above* and *below* can be expressed by other directional rules.

a) *east*(A,B) $\vee$ *north-east*(A,B) $\vee$ *south-east*(A,B) $\Longleftrightarrow$ *right*(A,B)
b) *west*(A,B) $\vee$ *north-west*(A,B) $\vee$ *south-west*(A,B) $\Longleftrightarrow$ *left*(A,B)
c) *north*(A,B) $\vee$ *north-east*(A,B) $\vee$ *north-west*(A,B) $\Longleftrightarrow$ *above*(A,B)
d) *south*(A,B) $\vee$ *south-east*(A,B) $\vee$ *south-west*(A,B) $\Longleftrightarrow$ *below*(A,B)

A.2 Strict Topological Rules

Rule Set 2.1 (Inverse Property) The relations *inside* and *cover* are inverses of *contains* and *covered-by*, respectively.

a) *inside*(A,B) $\Longleftrightarrow$ *contains*(B,A)
b) *cover*(A,B) $\Longleftrightarrow$ *covered-by*(B,A)

Rule Set 2.2 (Reflexivity) The relations *equal* and *overlap* are reflexive.

a) *equal*(A,A)
b) *overlap*(A,A)

Rule Set 2.3 (Symmetry) The relations *equal*, *overlap*, *disjoint* and *touch* are symmetric.

a) *equal*(A,B) $\iff$ *equal*(B,A)
b) *overlap*(A,B) $\iff$ *overlap*(B,A)
c) *disjoint*(A,B) $\iff$ *disjoint*(B,A)
d) *touch*(A,B) $\iff$ *touch*(B,A)

Rule Set 2.4 (Transitivity) The relations *inside* and *equal* are transitive.

a) *inside*(A,B) $\wedge$ *inside*(B,C) $\implies$ *inside*(A,C)
b) *equal*(A,B) $\wedge$ *equal*(B,C) $\implies$ *equal*(A,C)

Rule Set 2.5 The relations *inside*, *equal* and *cover* imply the relation *overlap*.

a) *inside*(A,B) $\implies$ *overlap*(A,B)
b) *equal*(A,B) $\implies$ *overlap*(A,B)
c) *cover*(A,B) $\implies$ *overlap*(A,B)

Rule Set 2.6 The relationships between *equal* and {*cover*, *inside*, *disjoint*, *touch*, *overlap*} are as follows:

a) *equal*(A,B) $\wedge$ *cover*(B,C) $\implies$ *cover*(A,C)
b) *equal*(A,B) $\wedge$ *cover*(C,B) $\implies$ *cover*(C,A)
c) *cover*(A,B) $\wedge$ *equal*(A,C) $\implies$ *cover*(C,B)
d) *cover*(A,B) $\wedge$ *equal*(B,C) $\implies$ *cover*(A,C)
e) *equal*(A,B) $\wedge$ *inside*(B,C) $\implies$ *inside*(A,C)
f) *equal*(A,B) $\wedge$ *inside*(C,B) $\implies$ *inside*(C,A)
g) *inside*(A,B) $\wedge$ *equal*(A,C) $\implies$ *inside*(C,B)
h) *inside*(A,B) $\wedge$ *equal*(B,C) $\implies$ *inside*(A,C)
i) *equal*(A,B) $\wedge$ *disjoint*(B,C) $\implies$ *disjoint*(A,C)
j) *disjoint*(A,B) $\wedge$ *equal*(B,C) $\implies$ *disjoint*(A,C)
k) *equal*(A,B) $\wedge$ *overlap*(B,C) $\implies$ *overlap*(A,C)
l) *overlap*(A,B) $\wedge$ *equal*(B,C) $\implies$ *overlap*(A,C)
m) *equal*(A,B) $\wedge$ *touch*(B,C) $\implies$ *touch*(A,C)
n) *touch*(A,B) $\wedge$ *equal*(B,C) $\implies$ *touch*(A,C)

Rule Set 2.7 The relationships between *disjoint* and {*inside*, *touch*, *cover*} are as follows:

a) *inside*(A,B) $\wedge$ *disjoint*(B,C) $\Longrightarrow$ *disjoint*(A,C)
b) *disjoint*(A,B) $\wedge$ *inside*(C,B) $\Longrightarrow$ *disjoint*(A,C)
c) *inside*(A,B) $\wedge$ *touch*(B,C) $\Longrightarrow$ *disjoint*(A,C)
d) *touch*(A,B) $\wedge$ *inside*(C,B) $\Longrightarrow$ *disjoint*(A,C)
e) *cover*(A,B) $\wedge$ *disjoint*(C,A) $\Longrightarrow$ *disjoint*(C,B)
f) *disjoint*(A,B) $\wedge$ *cover*(A,C) $\Longrightarrow$ *disjoint*(C,B)

Rule Set 2.8 The relationships between *overlap* and {*inside*, *cover*} are as follows (excluding those given by Rule Set 2.5):

a) *inside*(A,B) $\wedge$ *overlap*(C,A) $\Longrightarrow$ *overlap*(B,C)
b) *overlap*(A,B) $\wedge$ *inside*(B,C) $\Longrightarrow$ *overlap*(A,C)
c) *cover*(A,B) $\wedge$ *overlap*(B,C) $\Longrightarrow$ *overlap*(A,C)
d) *overlap*(A,B) $\wedge$ *cover*(C,B) $\Longrightarrow$ *overlap*(A,C)

Rule Set 2.9 The relationships between *inside* and *cover* are as follows:

a) *inside*(A,B) $\wedge$ *cover*(C,B) $\Longrightarrow$ *inside*(A,C)
b) *inside*(A,C) $\wedge$ *cover*(A,B) $\Longrightarrow$ *inside*(B,C)
c) *cover*(A,B) $\wedge$ *cover*(B,C) $\wedge$ not(*inside*(C,A)) $\Longrightarrow$ *cover*(A,C)

A.3 Heterogeneous Directional and Topological Rules

Rule Set 3.1 If $\beta \in$ S, where S is the set of directional relations, then

a) *equal*(A,B) $\wedge$ β(B,C) $\Longrightarrow$ β(A,C)
b) β(A,B) $\wedge$ *equal*(B,C) $\Longrightarrow$ β(A,C)

A.4 3D Rules

Rule Set 4.1 (Reflexivity) The relation *samelevel* is reflexive.

samelevel(A,A)

Rule Set 4.2 (Symmetry) The relation *samelevel* is symmetric.

samelevel(A,B) $\Longrightarrow$ *samelevel*(B,A)

Rule Set 4.3 (Inverse Property) The relations *infrontof*, *strictlyinfrontof* and *touchfrombehind* are inverses of *behind*, *strictlybehind* and *touchedfrombehind*, respectively.

a) *infrontof*(A,B) $\Longleftrightarrow$ *behind*(B,A)
b) *strictlyinfrontof*(A,B) $\Longleftrightarrow$ *strictlybehind*(B,A)
c) *touchfrombehind*(A,B) $\Longleftrightarrow$ *touchedfrombehind*(B,A)

Rule Set 4.4 (Transitivity) The relations *strictlyinfrontof* and *samelevel* are transitive.

a) *strictlyinfrontof*(A,B) $\wedge$ *strictlyinfrontof*(B,C) $\Longrightarrow$ *strictlyinfrontof*(A,C)
b) *samelevel*(A,B) $\wedge$ *samelevel*(B,C) $\Longrightarrow$ *samelevel*(A,C)

Rule Set 4.5 The relationship between *touchfrombehind* and *infrontof* is as follows:

touchfrombehind(A,B) $\Longrightarrow$ *infrontof*(B,A)

Rule Set 4.6 The relationships between *strictlyinfrontof* and *samelevel* are as follows:

a) *samelevel*(A,B) $\wedge$ *strictlyinfrontof*(B,C) $\Longrightarrow$ *strictlyinfrontof*(A,C)
b) *strictlyinfrontof*(A,B) $\wedge$ *samelevel*(B,C) $\Longrightarrow$ *strictlyinfrontof*(A,C)

Rule Set 4.7 The relationship between *infrontof* and *strictlyinfrontof* is as follows:

infrontof(A,B) $\Longrightarrow$ *strictlyinfrontof*(A,B)