

TECHNICAL RESEARCH REPORT

AVIS: An Advanced Video Information System

*by S. Adali, K.S. Candan, K. Erol,
V.S. Subrahmanian*

T.R. 97-44



*Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry*

AVIS: AN ADVANCED VIDEO INFORMATION SYSTEM *

Sibel Adalı

K. Selçuk Candan

Kutluhan Erol

V.S. Subrahmanian

Department of Computer Science,

Institute for Advanced Computer Studies &

Institute for Systems Research,

University of Maryland,

College Park, Maryland 20742.

Email: {sibel, candan, kutluhan, vs}@cs.umd.edu.

Su-Shing Chen

Information Technology & Organizations Program,

National Science Foundation, 4201 Wilson Blvd,

Arlington, VA 22230.

Email: schen@nsf.gov.

ABSTRACT

During the last few years, the advent of the CD-Rom, and the introduction of high bandwidth communications networks has caused a spectacular explosion in the availability of large video-libraries. While a great deal of effort has been invested in problems of how to effectively utilize bandwidth to communicate large bodies of data across the network, relatively little effort has gone into how to organize, and access, video databases. In this paper, we describe how video data may be organized and structured so as to facilitate queries. We develop a formal model of video data and show how spatial data structures, suitably modified, provide an elegant way of storing such data. We develop algorithms to process various kinds of video queries and show that in most cases, the complexity of these algorithms is linear. We develop algorithms to update these video databases. A prototype system called AVIS ("Advanced Video Information System") has been designed at the University of Maryland based on these concepts.

INTRODUCTION

*This research was supported by the Army Research Office under grant DAAH-04-95-10174, by the Air Force Office of Scientific Research under grant F49620-93-1-0065, by ARPA/Rome Labs contract Nr. F30602-93-C-0241 (Order Nr. A716), and by an NSF Young Investigator award IRI-93-57756.

The primary aim of this paper is to study methods of indexing video databases so as to efficiently store and retrieve video data in a number of diverse ways. Some example queries to be used for such retrievals may be:

(Query 1) find all video-frames in which John Wayne appears,

(Query 2) find all video-frames in which someone is being murdered, or more complicated ones such as

(Query 3) find all video-frames in which Dean Martin appears and Fred Astaire is dancing with Ginger Rogers,

(Query 4) find all people who appear in the frames where Gene Kelly and Ginger Rogers are getting married,

(Query 5) find all video-frames showing Ginger Rogers dancing with someone other than Fred Astaire when he is in the same room, (note that Fred Astaire may not be visible in all these frames, but the fact that he is in the room during that time may be known in advance)

(Query 6) find all video-frames in which Alfred Hitchcock appears after someone is killed.

The primary contributions of this paper are the following:

1. First of all, we will show that certain kinds of existing spatial data structures are adequate for efficiently storing certain kinds of information. In particular, we will show that the problem of storing which objects occur in which frames may be viewed as an equivalent problem of storing line segments.
2. Second, we will show how we may make use of the above “spatial database” intuition to efficiently index multimedia data so as to be able to handle all the kinds of queries listed above – in particular, we will show how a combination of spatial database technology and relational database technology can be “merged” together to efficiently solve all the kinds of queries above.
3. Third, we will describe how updates to information about video data may be efficiently implemented using the data structures defined above.
4. Finally, we will describe a prototype implementation, called AVIS (“Advanced Video Information System”) of the above data structures and algorithms. The implementation suggests that the theoretical algorithms work very well in practice.

VIDEO INFORMATION

If we look at the queries given in the introduction section, we will notice that two types of information are being queried. The first type is a set of *entities* – things of interest to us in the movie. The second type of information is the video-frames in which these entities are present. In this section, we will list the types of entities in a video that are of interest and later we will explain how the notion of the occurrence of an entity can be mapped onto the spatial domain.

- **Video Objects:** These are the entities that are present in the video frames.
- **Activity Types:** Informally speaking, an activity describes the subject of a given video frame-sequence.
- **Event:** An event may be thought of as an instantiation of an activity type.

- **Roles:** The events that are of the same activity type are differentiated from each other by the set of objects involved in them.
- **Teams:** A team is a set of objects/ descriptions that jointly describe an event, i.e. they are the instantiation of the roles in an activity type.

Frame Sequences: In the area of video-databases, we observe that video-data may be classified as follows: a body of *raw video data*, RVD, for short, is a set of entities called *frames*. In addition to the raw video data, there is a set, ENT, of “entities” – objects that are of interest as given above within the domain of a given RVD. Last but not least, there is an association map, λ , that specifies which objects occur in which video-frames.

We will assume that the set of frames, RVD, is the set $\{1, \dots, n\}$ for some arbitrarily chosen, but fixed integer n . Likewise, we will assume that the set of objects is also the set $\{1, \dots, k\}$ for some arbitrarily chosen, but fixed integer k . There is no loss of generality in making this assumption; it is convenient in providing a formal mathematical framework for reasoning about multimedia systems.

A *frame-sequence* is a pair $[i, j]$ where $1 \leq i \leq j \leq n$. $[i, j]$ represents the set of all frames between i (inclusive) and j (non-inclusive). In other words, $[i, j] = \{k \mid i \leq k < j\}$. i (resp. j) is said to be the *start* (resp. *end*) of the frame-sequence $[i, j]$.

We may define a partial ordering, \sqsubseteq , on the set of all frame sequences as follows: $[i_1, j_1] \sqsubseteq [i_2, j_2]$ iff $i_1 < j_1 \leq i_2 < j_2$. Intuitively, $[i_1, j_1] \sqsubseteq [i_2, j_2]$ means that the sequence of frames denoted by $[i_1, j_1]$ precedes the sequence of frames denoted by $[i_2, j_2]$. As usual, we use $[i_1, j_1] \sqsubset [i_2, j_2]$ to denote that $[i_1, j_1] \sqsubseteq [i_2, j_2]$ and $j_1 \neq i_2$.

A set, X , of frame-sequences is said to be *well-ordered* iff:

1. X is finite, i.e. $X = \{[i_1, j_1], \dots, [i_r, j_r]\}$ for some integer r , and
2. $[i_1, j_1] \sqsubseteq [i_2, j_2] \sqsubseteq \dots \sqsubseteq [i_r, j_r]$.

A set, X , of frame-sequences is said to be *solid* iff

1. X is well-ordered, and
2. there is no pair of frame-sequences in X of the form $[i_1, i_2)$ and $[i_2, i_3)$.

An *association-map* is a function, λ , such that for each entity $ent \in ENT$, $\lambda(ent)$ is a solid set of frame-sequences. Intuitively, $\lambda(ent) = \{[i_1, j_1), [i_2, j_2)\}$ means that entity ent occurs in all frames in the frame-sequences $[i_1, j_1)$ and $[i_2, j_2)$.

Association Maps and Segment Trees: Association maps correspond exactly to line segments on the x -axis of the Cartesian plane, and hence, association maps may be stored using any method to store such collinear line segments, such as the well known segment tree (cf. Samet [11]). The one difference between our segment trees, and the ordinary segment trees (cf. Samet[11]) is that a single object name may label multiple line segments. To differentiate between the two, we will refer to our data structure as a frame segment tree.

The observation that association maps correspond to line segments is not new – it has been made by several individuals in the multimedia community.

One of the key observations in our paper is that this object-frame map is just a set of line segments all of which are horizontal. Consequently, the frames in which a given object appears may be viewed as a set of line segments on the x -axis, obtained by projecting the shaded line segments shown in the figure onto the x -axis.

Hence, an appropriate data structure for this problem would contain:

- a segment tree representing the set of all line segments in RVD, and
- additional arrays, such as OBJECTARRAY, EVENTARRAY and ACTIVITYARRAY which provide additional indices to the nodes in the tree denoting where the corresponding data is located.

Frame Segment Tree Representation of Video-Frames:

1. Each node in the frame segment tree represents a *frame-sequence* $[x, y)$ starting at frame x and in-

cluding all frames up to, but not including, frame y .

2. The number inside each node may be viewed as a pointer to that node.
3. The set of numbers (in boldface) placed next to a node denotes the id-numbers of video objects and events that appear in the entire frame-sequence associated 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).

The OBJECTARRAY: The object array is an array whose i 'th element denotes video object number i (denoted by oi). Associated with any element of this array is an *ordered linked list* of pointers to nodes in the frame segment tree.

The EVENTARRAY: As in the case of the OBJECTARRAY the event array stores the frame sequences, in the form of links to the frame segment tree, for each different event. Events are uniformly numbered from $e1$ to eN . For each event, the following information is stored: the activity type, the team as a list of pairs of the form (role: player) and finally, a list of tree nodes that indicates the frame sequences in which this event occurs.

The ACTIVITYARRAY: This is another index which simply stores, for each activity type, the list of events of that type. This will facilitate queries about a certain activity type as opposed to a certain event of that type. Similarly, the ROLEARRAY simply lists the name of the roles, since the roles are also stored by their unique id number in the video database. In addition to these, we assume that we have hash tables for objects, activity types and roles that map their names to their unique id numbers. We will not give details of these tables, since they're only used to locate the id numbers.

Video Database: A video-database is a 9-tuple

$(RVD, OBJ, EVT, \lambda, ACT, \mathfrak{R}, \rho, ROLE, PLAYERS)$

where

- RVD is a set of integers $\{1, \dots, n\}$ for some n ;
- OBJ is a set whose members are called “objects”;
- EVT is a set whose elements are called *events*;
- λ is a that assigns to each entity $ent \in (\text{OBJ} \cup \text{EVT})$, a solid set of frame-sequences;
- ACT is a set whose elements are called *activity types*;
- \mathfrak{R} is a set whose elements are called *roles*;
- \wp is a map that assigns, to each event an activity type;
- ROLE is a map that takes as input, a frame-sequence $[i, j] \in \text{EVT} \cup \bigcup_{o \in \text{OBJ}} \lambda(o)$, an activity A in $\wp([i, j])$, and a role in $\mathfrak{R}(A)$, and assigns as output, a member of OBJ;
- PLAYERS is a map that takes as input an event and its activity type, as output returns a mapping from the roles of the activity to the entities in the database and to strings.

Elementary Object Queries: This is a query of the form: “Given the name of an object, find all video frames in which a given object occurs.”

Elementary Activity Type Queries: This is a query of the form: “Find all the video frames in which events of a given activity type occurs.”

Detailed Activity Queries – Event Queries: This is a query of the form: “Find all video frames in which one of a given set of events occurs, where the events are specified by the activity type and the roles of specific objects involved in this activity”

Object Occurrence Queries: This is a query of the form: “Find all objects that occur in a given set of frame sequences.”

Activity Type Occurrence Queries: This is a query of the form: “Find all the activities that occur in a given set of frame sequences.”

Conjunctive Queries: (1) Queries that take as input, a specification, and return as output, a set of frame sequences, (2) queries that take as input, a set of frame sequences, and return a set of objects, (3) same type of input as 2, but returns as output, a set of events, (4) there are some other queries that we have not discussed, which take the same types of input as 2 and 3, but they return as output return activity types, role of a specific object and etc.

Compound Queries: For example, suppose we consider the query “Find all the objects present in the video when a certain event (specified by the activity type and a team) occurs.” In general, answers to compound queries are found by directing the output of one function as input to another function.

RELATED WORK

Gibbs et. al. [2] study how stream-based temporal multimedia data may be modeled using object based methods. However, concepts such as roles and players, the distinction between activities and events, and the integration of such video systems with other traditional database systems are not addressed.

Hjelsvold and Midtstraum [5] develop a “generic” data model for capturing video content and structure. Their idea is that video should be included as a data type in relational databases, i.e. systems such as PARADOX, INGRES, etc. should be augmented to handle video data. In particular, they study temporal queries – something we do not do in this paper. In contrast, the innovation in our approach is the use of well studied spatial (rather than temporal)’ data structures, suitably modified, to query video data. We have used those intuitions to develop techniques for processing queries as well as for *updating* these video data structures – the first algorithms we know of for incorporating updates to video data.

Oomoto and Tanaka [9] have developed a language called VideoSQL for accessing video data. In contrast to their effort, in this paper we develop indexing structures and query processing algorithms for querying video data, and use a logical query language to query the data.

There has also been a good deal of work on picture re-

trieval systems [4, 3, 8] – however, these works deal with static images, whereas in contrast we have studied ways of organizing temporal sequences of such images (i.e. video).

CONCLUSION

Once an electronically accessible video library exists (and some such libraries exist even today, e.g. at the National Archives in College Park, Md.), the need to access these database “by content” assumes great importance. The primary aim of this paper is to develop techniques by which the content of these movies can be stored on-line so as to facilitate such accesses.

To accomplish this, we have presented a succinct theoretical description of video databases. We have then shown how such video databases may be stored electronically, and furthermore, we have designed polynomial-time query processing algorithms that traverse these data structures. In addition, as the “content” of these video databases may be updated from time to time, we have developed methods to implement such updates. Finally, we have developed an implemented system called AVIS (“Advanced Video Information System”) that implements the various video-based queries described here.

References

- [1] S. Adalı, K.S. Candan, Su-Shing Chen, K. Erol, and V.S. Subrahmanian. (1996) *Advanced Video Information System: Data Structures and Query Processing*. ACM-Springer Multimedia Systems Journal, Vol. 4, pp 172-186, August 96.
- [2] S. Gibbs, C. Breiteneder and D. Tschritzis. (1994) *Data Modeling of Time-Based Media*, Proc. 1994 ACM SIGMOD Conf. on Management of Data, pps 91-102.
- [3] V.N. Gudivada and V.V. Raghavan. (1993) *Design and Evaluation of Algorithms for Image Retrieval by Spatial Similarity*, to appear in: ACM Transactions on Information Systems.
- [4] V.N. Gudivada, V.V. Raghavan and K. Vanapipat. (1994) *A Unified Approach to Data Modeling and Retrieval for a Class of Image Database Applications*, to appear in: “Multimedia Database Sys-

tems” (eds. S. Jajodia and V.S. Subrahmanian), Springer.

- [5] R. Hjelsvold and R. Midtstraum. (1994) *Modeling and Querying Video Data*, Proc. 1994 Intl. Conf. on Very Large Databases, pps 686-694, Santiago, Chile.
- [6] J. Lu, A. Nerode, V.S. Subrahmanian, Hybrid Knowledge Bases, *IEEE Transactions on Knowledge and Data Engineering*, 1994. To appear. Available as technical report CS-TR-3037, University of Maryland.
- [7] S. Marcus, V.S. Subrahmanian, *Multimedia Database Systems*, 1994. Submitted for publication.
- [8] W. Niblack, et. al., The QBIC Project: Querying Images by Content Using Color, Texture and Shape, IBM Research Report, Feb. 1993.
- [9] E. Oomoto and K. Tanaka. (1993) *OVID: Design and Implementation of a Video-Object Database System*, IEEE Trans. on Knowledge and Data Engineering, 5, 4, pps 629-643.
- [10] L. Rowe and B.C. Smith. (1992) *A Continuous Media Player*, Proc., 3rd Intl. Workshop on Network and Operating Systems Support for Digital Audio and Video, pps 237-249, Nov. 1992.
- [11] H. Samet *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, 1989.
- [12] J.D. Ullman *Principles of Database and Knowledge-base Systems*, Computer Science Press, 1989.