Modeling Video Temporal Relationships in an Object Database Management System

John Z. Li, Iqbal A. Goralwalla, M. Tamer Özsu, and Duane Szafron
Department of Computing Science, University of Alberta
Edmonton, Canada T6G 2H1 {zhong,iqbal,ozsu,duane}@cs.ualberta.ca

ABSTRACT

One of the key aspects of videos is the temporal relationship between video frames. In this paper we propose a tree-based model for specifying the temporal semantics of video data. We present a unique way of integrating our video model into an object database management system which has rich multimedia temporal operations. We further show how temporal histories are used to model video data, explore the video objectbase using object-oriented techniques. Such a seamless integration gives a uniform interface to end users. The integrated video objectbase management system supports a broad range of temporal queries.

Keywords: multimedia, temporal, object-oriented, database, video model, query, clips

1 INTRODUCTION

Management of multimedia data poses special requirements on database management systems. In this paper, we concentrate on the video data type, especially *video modeling*, which is the process of translating raw video data into an efficient internal representation for capturing video semantics. A video model is an essential part of an abstract multimedia information system which can be used as a basis for declarative querying. The abstract model has to be mapped to a concrete one. Object-oriented technology is generally accepted as a promising tool for modeling multimedia data.[1–3] Most of the video models either employ image processing techniques for indexing video data or use traditional database approaches based on keywords or annotated textual descriptions[4–6] to represent video semantics. In most cases, the annotated description of the video contents is created manually. This is a time consuming process. This paper proposes a video model called the Common Video Object Tree (CVOT). The model has the capability of automatic video segmentation and incorporates both spatial and temporal relationships among video objects. This allows native support for a rich set of temporal multimedia operations. We focus on temporal relationships in this paper.

We seamlessly integrate the abstract CVOT model with a powerful temporal object model to provide concrete object database management system (ODBMS) support for video data. The system that we use in this work is TIGUKAT,[7] which is an experimental system under development at the University of Alberta. We exploit the behaviorality and uniformity of the TIGUKAT object model in incorporating the CVOT model uniformly.

Figure 1 shows the proposed process and the shaded boxes are the focuses of this paper. Raw video data is processed by a *Video Analyzer* which uses video and image processing techniques via a *Video Tool Library* to recognize salient (physical) objects, video shots, etc. The main function of the *Video Analyzer* is to make feature extraction. As for what type of features are extracted depends on applications. Restricted by the current technology, such an analysis process has to be domain specific. The salient objects are encoded in the CVOT model by their properties, such as size, location, moving direction, etc. Then a CVOT tree is generated by the *CVOT Model*. The *Video Structuring* module builds the necessary indexes from the output of both the *Video Analyzer* and the *CVOT*

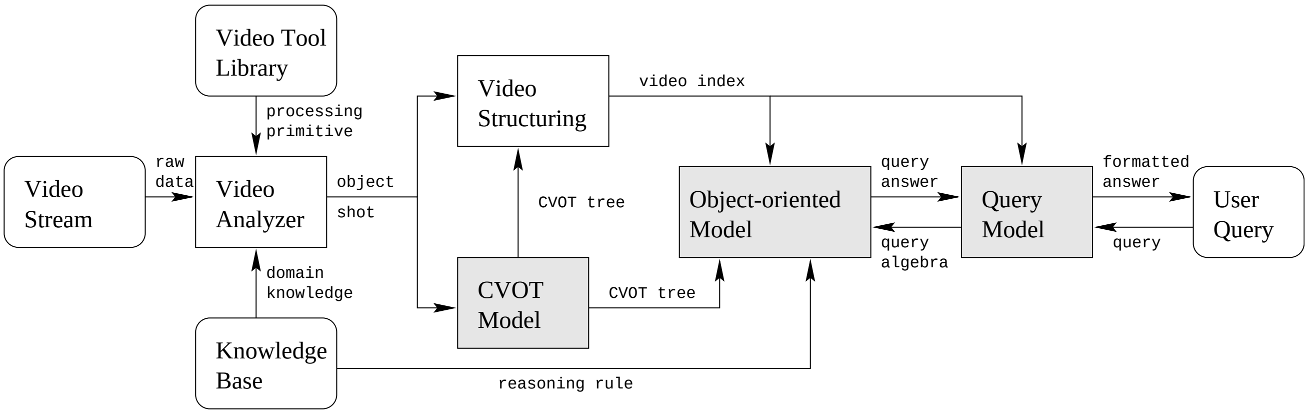

Figure 1: CVOT System Architecture

Model to provide efficient access. A unified model is necessary for users to query the system. An object-oriented model is used because of its powerful representation of the users' view and its suitability for multimedia data. Users post queries using a query language based on a *Query Model*. Then, the *Query Model* defines translation of the queries into an internal query algebra which can be executed by the system. In this procedure some query optimization can be done and therefore the *Query Model* needs to account for the specifics of video structuring. During the process of executing user queries, the system may not have all the essential information, but it can make temporal or spatial inference from the existing relations using the reasoning rules provided by the *Knowledge Base*. The answer to the queries is returned to the end user. Multimedia data require a very complicated presentation facility, such as synchronization of different media.

The major contributions of this paper are a new model for organizing video clips based on a common object tree, a unique way of integrating the CVOT model into an object database management system with rich temporal operations, a uniform approach of modeling video using temporal histories, and a integrated video object database supporting qualitative temporal operations on video data. The rest of the paper is organized as follows. The Common Video Object Tree model is introduced in Section 2. A brief discussion of the TIGUKAT system and its temporal extension is presented in Section 3. Section 4 describes how the CVOT model can be seamlessly integrated into the TIGUKAT. There are some query examples of showing the expressiveness of the integrated system in Section 5. The conclusions are given in Section 6.

2 THE COMMON VIDEO TREE MODEL

Each *video* consists of a number of *clips*. A *clip* is a consecutive sequence of *frames*, which are the smallest units of video data. The information about the semantics of a video must be structured so that indexes can be built for efficient data retrieval from a video object database. The functionality of a video object database depends on its model of time.

The main idea behind the Common Video Object Tree (CVOT) model is to find all the common objects among clips and to group clips according to these objects. We use a tree structure to represent such a clip group. In this section we give a formal definition of the model and then give an algorithm for constructing the tree. The proof of the theorems can be found in reference.[8]

2.1 Video Clip Sets

A clip is associated with a time interval $[t_s, t_f]$. More specifically, a clip is a set of consecutive frames between a start time t_s and a finish time t_f: $[t_s, t_f] = \{t | t_s \leq t \leq t_f\}$ where t_s and t_f are the relative (discrete) time instants in a given video and $t_s \leq t_f$. Since all clips have a start and finish time, a partial order could be defined over clips. To simplify notation, we use $C_i = [t_{s_i}, t_{f_i}]$ to mean that clip C_i is associated with a time interval $[t_{s_i}, t_{f_i}]$. Semantically, C_i is the set of all the frames within this interval.

DEFINITION 1. Let $C_i = [t_{s_i}, t_{f_i}]$ and $C_j = [t_{s_j}, t_{f_j}]$ be two clips. Then $\preceq$ is defined as the partial order over clips with $C_i \preceq C_j$ iff $t_{s_i} \leq t_{s_j}$ and $t_{f_i} \leq t_{f_j}$. Also, $C_i \prec C_j$ iff $t_{f_i} < t_{s_j}$.

DEFINITION 2. A set of clips C is said to be *ordered* iff C is finite, i.e., $C = \{C_1, \ldots, C_m\}$ and there exists a partial order such that $C_1 \preceq C_2 \preceq \ldots \preceq C_m$. A set of clips $C = \{C_1, C_2, \ldots, C_m\}$ is said to be *strongly ordered* iff C is ordered and $C_1 \prec C_2 \prec \ldots \prec C_m$. A set of clips $C = \{C_1 \ldots, C_m\}$ is said to be *perfectly ordered* iff C is ordered and for some two neighboring clips $C_i = [t_{s_i}, t_{f_i}]$ and $C_{i+1} = [t_{s_{i+1}}, t_{f_{i+1}}]$, we have $t_{s_{i+1}} = t_{f_i} + 1$ $(\forall i = 1, 2, \ldots, m-1)$. A set of clips $C = \{C_1 \ldots, C_m\}$ is said to be *weakly ordered* iff C is ordered and for any two neighboring clips $C_i = [t_{s_i}, t_{f_i}]$ and $C_{i+1} = [t_{s_{i+1}}, t_{f_{i+1}}]$, we have $t_{f_i} \geq t_{s_{i+1}}$ $(\exists i = 1, 2, \ldots, m-1)$.

THEOREM 1. All perfectly ordered clip sets are also strongly ordered.

2.2 Salient Objects

A *salient object* is an interesting physical object in a video frame. Each video frame has many salient objects, e.g. persons, houses, cars, etc. We assume there is always a finite set (possibly empty) of salient objects $SO = \{SO_1, SO_2, \ldots, SO_n\}$ for a given video. In a given frame, the spatial property of an SO_i is defined by a minimum bounding rectangle (X_i, Y_i), where $X_i = [x_{s_i}, x_{f_i}], Y_i = [y_{s_i}, y_{f_i}]$. x_{s_i} and x_{f_i} are salient object SO_i's projection on X axis and similarly for y_{s_i}, y_{f_i}. Hence, a salient object's spatial property can be represented by (X_i, Y_i).

Let $\mathcal{SO}$ be the collection of all salient object sets and $\mathcal{C}$ be the collection of all clip sets. We introduce two functions. One is the function $\mathcal{F}$: $SO \rightarrow \mathcal{C}$ which maps a salient object from $SO \in \mathcal{SO}$ into an ordered clip set $C \in \mathcal{C}$. The other is the function $\mathcal{F}^*$: $C \rightarrow \mathcal{SO}$ which maps a clip from $C \in \mathcal{C}$ into a salient object set $SO \in \mathcal{SO}$. Intuitively, function $\mathcal{F}$ returns a set of clips which contains a particular salient object while the reverse function $\mathcal{F}^*$ returns a set of salient objects which belong to a particular clip. We define the *common salient objects* for a given clip set as those salient objects which appear in every clip within the set. Some salient objects may appear in many different clips, but others may not. Hence, the number of common salient objects between clips are different. In order to quantify such a difference we introduce clip *affinity*.

DEFINITION 3. The *affinity* of m clips $\{C_1, \ldots, C_m\}$ is defined as

$$aff(C_1, \ldots, C_m) = |\mathcal{F}^*(C_1) \cap \mathcal{F}^*(C_2) \cap \ldots \cap \mathcal{F}^*(C_m)|$$

where $\{C_1, \ldots, C_m\}$ is an ordered clip set, $m \geq 2$, $|X|$ is the cardinality of set X, and $\cap$ is set intersection.

EXAMPLE 1. Figure 2 shows a video in which John using bat bat1 and Ken using bat bat2 are playing table tennis while Mary is watching. After playing, John drives home in his car. Let us assume that the salient objects are $SO = \{\mathsf{john}, \mathsf{ken}, \mathsf{mary}, \mathsf{ball}, \mathsf{bat1}, \mathsf{bat2}, \mathsf{car}\}$. If the video is segmented as in Figure 2 (it might be segmented into different clip sets by different approaches), then $C = \{C_1, C_2, C_3, C_4, C_5\}$ is a perfectly ordered clip set. Furthermore, john, ball, and bat1 are in C_1; john, mary, ball, bat1 are in C_2; ken, ball, bat2 are in C_3; ken, ball, bat2 are in C_4; and john, car are in C_5. Then,

$\mathcal{F}(\mathsf{john}) = \{C_1, C_2, C_5\}$ $\qquad$ $\mathcal{F}^*(C_1) = \{\mathsf{john}, \mathsf{ball}, \mathsf{bat1}\}$
$\mathcal{F}(\mathsf{ken}) = \{C_3, C_4\}$ $\qquad$ $\mathcal{F}^*(C_2) = \{\mathsf{john}, \mathsf{ball}, \mathsf{bat1}, \mathsf{mary}\}$
$\mathcal{F}(\mathsf{mary}) = \{C_2\}$ $\qquad$ $\mathcal{F}^*(C_3) = \{\mathsf{ken}, \mathsf{ball}, \mathsf{bat2}\}$
$\mathcal{F}(\mathsf{ball}) = \{C_1, C_2, C_3, C_4\}$ $\qquad$ $\mathcal{F}^*(C_4) = \{\mathsf{ken}, \mathsf{ball}, \mathsf{bat2}\}$
$\mathcal{F}(\mathsf{bat1}) = \{C_1, C_2\}$ $\qquad$ $\mathcal{F}^*(C_5) = \{\mathsf{john}, \mathsf{car}\}$
$\mathcal{F}(\mathsf{bat2}) = \{C_3, C_4\}$ $\qquad$ $\mathcal{F}(\mathsf{car}) = \{C_5\}$.

Now, the affinity of C_1 and C_2 is

$aff(C_1, C_2) = |\mathcal{F}^*(C_1) \cap \mathcal{F}^*(C_2)| = |\{\mathsf{john, ball, bat1}\} \cap \{\mathsf{john, ball, bat1, mary}\} = |\{\mathsf{john, ball, bat1}\}| = 3.$

Similarly, $aff(C_2, C_3) = 1$, $aff(C_1, C_2, C_3) = 1$, etc.

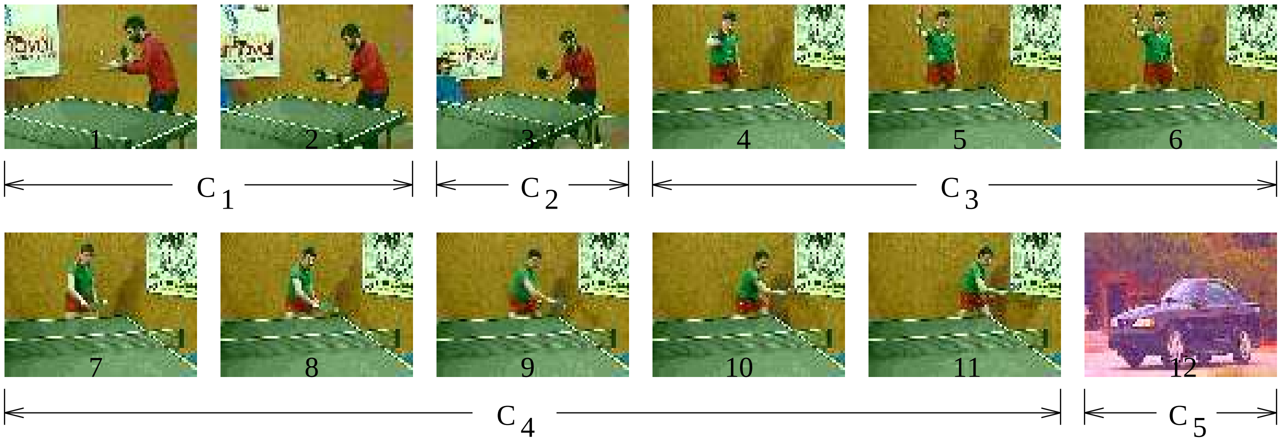

Figure 2: Salient Objects and Clips

Theorem 2. The affinity function is *monotonically non-increasing*. That is, if $\{C_1, \ldots, C_m\}$ is an ordered clip set, then $aff(C_1, \ldots, C_k) \geq aff(C_1, \ldots, C_k, C_{k+1})$ where $k = 2, 3, \ldots, m-1$.

2.3 The Common Video Object Tree

Clustering clips is an important issue as it affects both the effectiveness and efficiency of query retrievals. A clustering scheme should also maintain any existing temporal relationships among frames. We propose a tree-based model, called the *Common Video Object Tree* (CVOT), which builds a tree based on the common salient objects in a set of clips. Trees provide an easy and efficient way of clustering clips with less complexity than graphs. For any given weakly or perfectly ordered clip set C, each leaf node in a CVOT tree is an element of C. All the leaf nodes are ordered from left to right by their time intervals. An internal node represents a set of common salient objects, which appear in all its child nodes. The only node that can have an empty common salient object set is the root node or an node of the clip with an empty salient object set. Every node (including internal, leaf, and root node) has a time interval and a set of salient objects which appear during this time interval. Figure 3 shows an example of a CVOT tree which is built from Example 2. As seen in Figure 3, the cardinality of the common object set shrinks as we traverse the tree from the leaf nodes to the root. This is in conformance with the monotonically non-increasing nature of clip affinity stated in Theorem 2. The figure also shows how the time intervals are propagated up from the leaf nodes. For example, the internal node N_2 has the interval $[1, 3]$ which is composed of its two child leaf nodes C_1 and C_2. The root always spans all of the time intervals in the whole clip set.

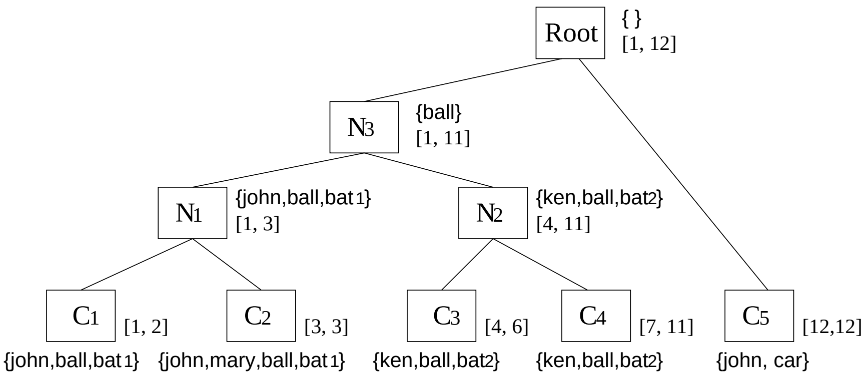

Figure 3: A Common Video Object Tree Built by GMCO Algorithm

We have developed a greedy algorithm, called *Greedy Maximum Common Objects* (GMCO), to build the CVOT tree.[8] Given a set of clips C, the GMCO algorithm builds the CVOT in a bottom-up fashion. The idea is to find the largest set of neighboring clips of the given set without reducing its affinity. An internal node is then created for this new set of clips. This process continues until either the common object set is empty or all nodes are merged into one node (root). If the common object set is empty, this node is attached to the root of the CVOT.

An array can be used to further index CVOT nodes for efficient query processing. Such an array is called CVOTArray. Each element of a CVOTArray is a salient object and consists of a linked list which links all the nodes where the salient objects appear. However, in any linked list, if there is a CVOT internal node, all its children will not be in this list so redundancy is eliminated. The cost of accessing such an array can be done by a hash function which is a constant. Hence the cost of searching for a salient object in a clip or in all clips is less than the number of its appearances.

3 THE ODBMS SUPPORT

CVOT is an abstract model. To have proper database management support for continuous media, this model needs to be integrated into a data model. We work within the framework of a uniform, behavioral object model such as the one supported by the TIGUKAT system.[7] The important characteristics of the model from the perspective of this paper are its *behaviorality* and its *uniformity*. The model is *behavioral* in the sense that all access and manipulation of objects is based on the application of behaviors to objects. The model is *uniform* in that every component of information, including its semantics, is modeled as a *first-class object* with well-defined behavior. The typical object-oriented features such as strong object identity, abstract types, strong typing, complex objects, full encapsulation, multiple inheritance, and parametric types are also supported.

The primitive objects of the model include: *atomic entities* (reals, integers, strings, etc.); *types* for defining common features of objects; *behaviors* for specifying the semantics of operations that may be performed on objects; *functions* for specifying implementations of behaviors over types; *classes* for automatic classification of objects based on type; and *collections* for supporting general heterogeneous groupings of objects.

In this paper, a reference prefixed by "T_" refers to a type, "**C_**" to a class, "*B_*" to a behavior, and "T_X< T_Y >" to the type T_X parameterized by the type T_Y. For example, T_person refers to a type, **C_person** to its class, *B_age* to one of its behaviors and T_collection< T_person > to the type of collections of persons. A reference such as David, without a prefix, denotes some other application specific reference. Consequently, the model separates the definition of object characteristics (a *type*) from the mechanism for maintaining instances of a particular type (a *class*). Temporality has been added to the TIGUKAT model[9] as type and behavior extensions of its type system. The following is a brief overview of the temporal ontology and temporal history features of this model. These features are relevant to the integration of the CVOT model into this particular temporal object model. Figure 4 gives part of the time type hierarchy that includes the temporal ontology and temporal history features of the temporal model.

A *time interval* is identified as the basic anchored specification of time and a wide range of operations on time intervals is provided. Unary operators which return the lower bound, upper bound and length of the time interval are defined. The model supports a rich set of ordering operations among intervals,[10] e.g., *precedes*, *overlaps*, *during*, etc., as well as set-theoretic operations e.g., *union*, *intersection* and *difference*.

A *time instant* (*moment*, *chronon*, etc.) is a specific anchored moment in time. A time instant is modeled as a special case of a (closed) time interval which has the same lower and upper bound, e.g., $Jan\ 24, 1996 = [Jan\ 24, 1996, Jan\ 24, 1996]$. A wide range of operations can be performed on time instants. A time instant can be compared with another time instant with the transitive comparison operators $<$ and $>$. A *time span* is an unanchored relative duration of time. A time span is basically an atomic cardinal quantity, independent of any time instant or time interval. A time span can be added to or subtracted from a time instant to return another time instant. A time instant can be compared with a time interval to check if it falls before, within or after the time interval. Time spans have a number of operations defined on them. A time span can be compared with another time span using the transitive comparison operators $<$ and $>$. A time span can be subtracted from or added to another time span to return a third time span. The detailed behavior signatures corresponding to the operations on time intervals, time

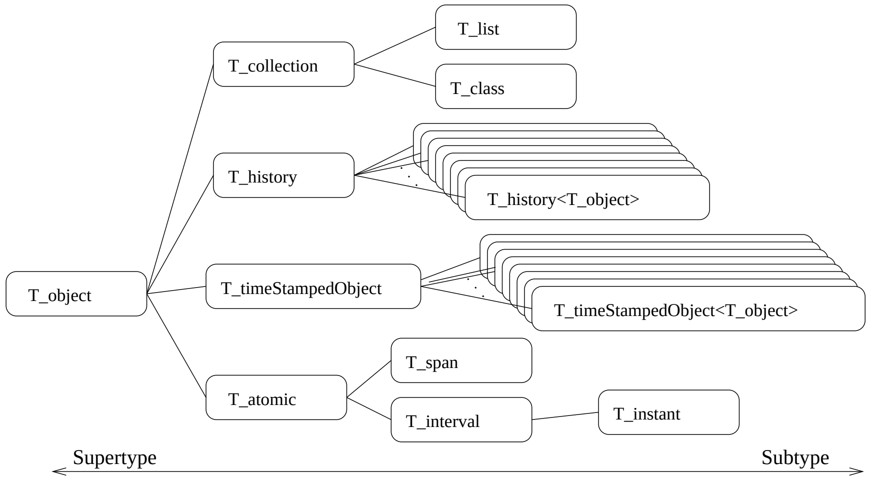

Figure 4: The Basic Time Type Hierarchy

instants, and time spans are given in the Table 1.

One requirement of a temporal model is an ability to adequately represent and manage histories of objects and real-world activities. Our model represents the temporal histories of objects whose type is `T_X` as objects of the `T_history<T_X>` type, as shown in Figure 4. A temporal history consists of objects and their associated timestamps (time intervals or time instants). A *timestamped object* knows its timestamp and its associated object (value) at (during) the timestamp. A temporal history is made up of such objects. Table 1 gives the behaviors defined on histories and timestamped objects. Behavior *B_history* defined on `T_history<T_X>` returns the set (collection) of all timestamped objects that comprise the history. Another behavior defined on history objects, *B_insert*, timestamps and inserts an object in the history. The *B_validObjects* behavior allows the user to get the objects in the history that were valid at (during) the given time.

Each timestamped object is an instance of the `T_timeStampedObject<T_X>` type. This type represents objects and their corresponding timestamps. Behaviors *B_value* and *B_timeStamp* defined on `T_timeStampedObject` return the value and the timestamp of a timestamped object, respectively.

4 SYSTEM INTEGRATION

Integrated multimedia systems can result in a uniform object model, simplified system support and possibly better performance. In such a system, the multimedia component can directly use many functions provided by the ODBMS, such as concurrency control, data recovery, access control etc. In this section we discuss the integration of the CVOT model into an ODBMS, with the temporal object model described in the previous section, as well as the type hierarchy and behavior definitions of video data. We explain why temporal histories are used to model the various features of the CVOT model and the contents of a video. Figure 5 shows our proposed video type system. The types that are in a grey shade are directly related to our video model and they will be discussed in detail throughout this section.

4.1 Integrating the CVOT Model

We start by defining the `T_video` type to model videos. An instance of `T_video` has all the semantics of a video. As we saw in Section 2, a video is segmented into a set of clips. Since a clip set is ordered and each clip has an associated time interval, it is natural to model this set as a history. We model a clip set by defining the behavior *B_clips* in `T_video`. *B_clips* returns a history object of type `T_history< T_clip >`, the elements of which

T_interval	B_lb:	T_instant
	B_ub:	T_instant
	B_length:	T_span
	B_precedes:	T_interval → T_boolean
	B_follows:	T_interval → T_boolean
	B_during:	T_interval → T_boolean
	B_meets:	T_interval → T_boolean
	B_overlaps:	T_interval → T_boolean
	B_starts:	T_interval → T_boolean
	B_finishes:	T_interval → T_boolean
	B_union:	T_interval → T_interval
	B_intersection:	T_interval → T_interval
	B_difference:	T_interval → T_interval
	B_subtract:	T_span → T_interval
	B_add:	T_span → T_interval
T_instant	B_lessthaneqto:	T_instant → T_boolean
	B_greaterthaneqto:	T_instant → T_boolean
	B_elapsed:	T_instant → T_span
	B_subtract:	T_span → T_instant
	B_add:	T_span → T_instant
	B_intersection:	T_interval → T_instant
	B_difference:	T_interval → T_instant
T_span	B_lessthan:	T_span → T_boolean
	B_greaterthan:	T_span → T_boolean
	B_lessthaneqto:	T_span → T_boolean
	B_greaterthaneqto:	T_span → T_boolean
	B_add:	T_span → T_span
	B_subtract:	T_span → T_span
T_history<T_X>	B_history:	T_collection<T_timeStampedObject<T_X>>
	B_insert:	T_X,T_interval → T_boolean
	B_validObjects:	T_interval → T_collection<T_timeStampedObject<T_X>>
T_timeStampedObject<T_X>	B_value:	T_X
	B_timeStamp:	T_interval

Table 1: Behaviors on Time Intervals, Time Instants, and History

are timestamped objects of type T_clip.

EXAMPLE 2. Suppose myVideo is an instance (object) of T_video. Then, myVideo.*B_clips* returns an instance (object) of type T_history< T_clip >. Let this object be myVideoClipHistory. myVideoClipHistory.*B_history* returns a collection (clip set) which contains all the timestamped clip objects of type T_timeStampedObject< T_clip > in myVideo. Let one of these clip history objects be myVideoCHOneClip. myVideoCHOneClip.*B_value* returns the object of myVideoCHOneClip, while myVideoCHOneClip.*B_timeStamp* returns the time interval of myVideoCHOneClip.

T_video	B_clips:	T_history<T_clip>
	B_cvotTree:	T_tree
	B_search:	T_salientObject, T_tree → T_tree
	B_weaklyOrdered:	T_boolean
	B_perfectlyOrdered:	T_boolean
	B_stronglyOrdered:	T_boolean
	B_length:	T_span
	B_publisher:	T_collection<T_company>
	B_producer:	T_collection<T_person>
	B_date:	T_instant
	B_play:	T_boolean
T_clip	B_frames:	T_history< T_frame >
	B_salientObjects:	T_collection<T_timeStampedObject<T_salientObject>>
	B_activities:	T_collection<T_timeStampedObject<T_activity>>
	B_affinity:	T_list<T_clip> → T_integer
	B_play:	T_boolean
T_frame	B_location:	T_instant
	B_content:	T_image

Table 2: Behavior Signatures of Videos, Clips, and Frames

Table 2 gives the behavior signatures of videos. The behavior *B_cvotTree* returns the common video object tree of a video. For example, myVideo.*B_cvotTree* creates a CVOT tree from the clip set of myVideo. Our implementations of *B_cvotTree* is the GMCO algorithm discussed in Section 2.3. *B_search* searches a CVOT tree and returns a subtree which contains a salient object.

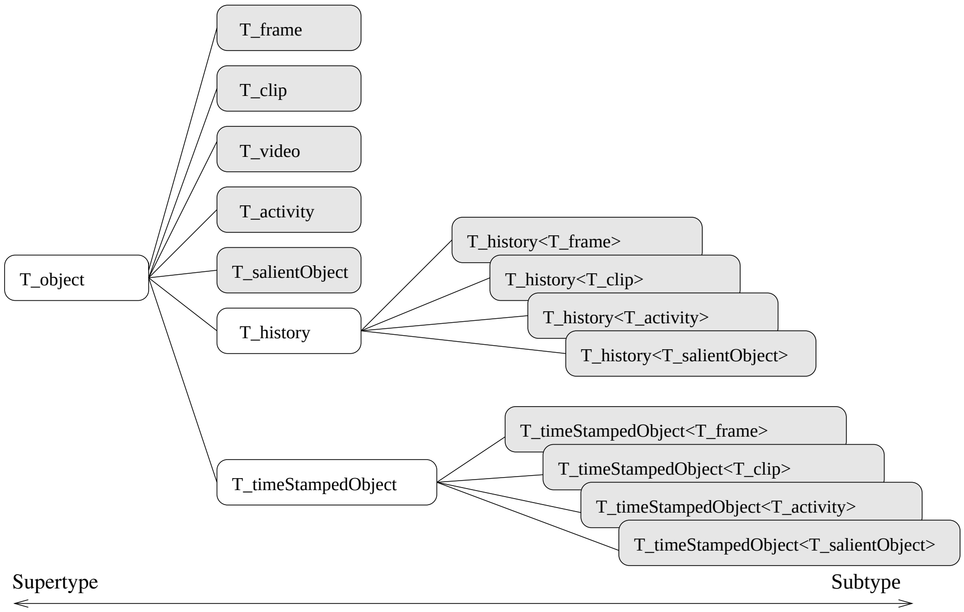

Figure 5: The Video Type System

In the CVOT model, a video knows the ordering of its clips. This ordering is defined by several video behaviors: *B_weaklyOrdered*, *B_perfectlyOrdered* and *B_stronglyOrdered* which simply iterate over the time intervals in the clip set history and determine whether a clip history is weakly ordered, perfectly ordered, or strongly ordered respectively.

A common question to myVideo would be its length (duration). This is modeled by the *B_length* behavior and it returns an object of type `T_span`. If a video is segmented into a perfectly ordered clip set, its length is equal to the total length of all the clips in this set. However, if the clip set is weakly ordered or strongly ordered, the video length is not equal to the total length of all the clips because in such clip sets, clips may overlap or disjoint. Video information should also include metadata, such as the publishers, producers, publishing date, etc. A video can also be played by using *B_play*.

Each clip has a set of consecutive frames, which is modeled by `T_history`<`T_frame`>. Since a clip must be associated with a time interval, we treat clips as timestamped objects. Suppose myClip is a particular clip, then myClip is an instance (object) of type `T_timeStampedObject`< `T_clip` >. The content of myClip is myClip.*B_value* while the interval of myClip is myClip.*B_timeStamp*. Some behavior signatures of clips are shown in Table 2.

All the salient objects within a clip are grouped by the behavior *B_salientObjects* which returns an instance of `T_collection`< `T_history` < `T_salientObjects` >>. Since a salient object can appear several times within a clip, such distinct appearances must be captured within the system. Therefore, history is a natural method to model this behavior. It is legitimate to ask a clip's affinity (*B_affinity*) with other clips. *B_play* of `T_clip` is able to play a clip on an appropriate output device. Other related operations, such as *stop, pause, play backward*, etc., are omitted from the table because they are not important to our discussions. Such omissions are also applicable to the behaviors of `T_video`. The basic building unit of a clip is the frame which is modeled by `T_frame` in Table 2. A frame knows its location within a clip or a video and such a location is modeled by a time instant (*B_location*), which can be a relative frame number. We model frames within a clip as a history which is identical to how we model clips within a video.

4.2 Modeling Video Features

The semantics or contents of a video is usually expressed by its *features* which include video attributes and the relationships between these attributes. Typical video features are salient objects and *activities*. An *activity* may involve many different salient objects over a time period, like holding a party, playing table tennis, and chatting with someone. An activity can occur in different places either within a clip or across multiple clips. For example, the activity johnDrive may occur in multiple clips. Additionally, this activity may occur several times within a clip. Therefore, an appropriate representation is necessary to capture the temporal semantics of general activities. A simple and natural way to model the temporal behavior of activities is to use a historical structure. Thus, we model histories of activities as objects of type `T_history< T_activity >`. Instances, such as johnDrive, of `T_history< T_activity >` consist of timestamped activities. This allows us to keep track of all the activities occurring within a video although an activity may occur in multiple clips or just occur within one clip. In the interest of tracking all the activities occurring within a clip, the behavior *B_activities* is included in `T_clip`.

Similarly, since salient objects can also appear multiple times in a clip or a video, we model the history of a salient object as timestamped object of type `T_history< T_salientObject >`. The behavior *B_salientObjects* of `T_clip` returns all the salient objects within a clip. Using histories to model salient objects and activities results in powerful queries as will be shown in the next subsection. Furthermore, it enables us to uniformly capture the temporal semantics of video data because a video is modeled as a history of clips and a clip is modeled as a history of frames.

T_activity	*B_activityType*:	T_activityType
	B_roles:	T_collection<T_person>
	B_inClips:	T_video → T_history< T_clip >
	B_activityObjects:	T_collection<T_salientObject>
T_salientObject	*B_boundingBox*:	T_history< T_boundingBox >
	B_centroid:	T_point
	B_inClips:	T_video → T_collection< T_timeStampedObject < T_clip >>
	B_status:	T_status

Table 3: Some Behavior Signatures of Activities and Salient Objects

The behavior *B_activityType* of `T_activity`, shown in Table 3, identifies the type of activities `T_activityType` and the behavior *B_roles* indicates all the persons involved in an activity. *B_activityObjects* returns all the salient objects within an activity. *B_inClips* indicates all the clips in which this activity occurs. It is certainly reasonable to include other information, such as the location and time of an activity, into type `T_activity`, but they are not important to our discussion.

The behavior *B_boundingBox* of type `T_salientObject` defines a bounding box on an object so its spatial information can be recorded. The bounding box values of a salient object may change as time goes on. Again a history is excellent to capture such behaviors. *B_centroid* returns the centroid point of a salient object. The behavior *B_inClips* returns all the clips in which the salient object appears. This corresponds to the reverse function $\mathcal{F}^*$ defined in Section 2.2. *B_status* may be used to define some other attributes of an salient object. For example, it is very useful to know whether an object is *rigid* or not if we want to track the motion of the object. Here we assume that `T_status` is such an enumerated type.

5 QUERYING VIDEO DATABASE

Since CVOT is incorporated into TIGUKAT, the queries can also be expressed in TIGUKAT object calculus.[11] This is a powerful calculus with a corresponding algebra that facilitates further optimization. The alphabet of the calculus consists of object constants, object variables, monadic predicates (C, P, Q), dyadic predicates $(=, \in, \notin)$, an n-ary predicate $(Eval)$, a function symbol (β) called *behavior specification* (Bspec), and logical quantifiers and connectives $(\exists, \forall, \wedge, \vee, \neg)$. The "evaluation" of a Bspec is accomplished by predicate $Eval$. A *term* is an object constant, an object variable or a Bspec. An *atomic formula* or *atom* has an equivalent Bspec representation. From atoms, *well-formed formulas* (WFFs) are built to construct the declarative calculus expressions of the language. WFFs are defined recursively from atoms in the usual way using the connectives $\wedge, \vee, \neg$ and the quantifiers $\exists$ and $\forall$. A query is an object calculus expression of the form $\{t_1, \ldots, t_n | \phi(o_1, \ldots, o_n)\}$ where $t_1, \ldots, t_n$ are the terms over the multiple variables $o_1, \ldots, o_n$. ϕ is a WFF.

For simplicity, we assume that all the queries are posted to a particular video instance myVideo. We also assume that all clips are timestamped clips and $\mathsf{c} \in \mathsf{myVideo}.\mathit{B\_clips}.\mathit{B\_history}$ where c is an arbitrary clip. myVideo.*B_clips* returns a history of all the clips in myVideo and myVideo.*B_clips*.*B_history* returns a collection of all the timestamped clips in myVideo. Since c is a timestamped clip, c belongs to the class **C_timeStampedObject** and the type of c is `T_timeStampedObject` $<$ `T_clip` $>$. Since all the clips, salient objects, activities belong to the timestamped object class **C_timeStampedObject**, we omit their specification in the query calculus expressions.

QUERY 1. Is the salient object o in clip c?

$\{\mathsf{o}.\mathit{B\_timeStamp}.\mathit{B\_during}(\mathsf{c}.\mathit{B\_timeStamp})\}$.

If o is a timestamped object, the above query checks whether the time interval of object o is a subinterval of clip c. However, if o is not a timestamp object we have to post the query by using clips associated with o:

$\{\mathsf{o}.\mathit{B\_inClips}(\mathsf{myVideo}).\mathit{B\_contains}(\mathsf{c})\}$.

Here, o.*B_inClips*(myVideo) returns a collection of all timestamped clips containing o. The behavior *B_contains*(c), defined in `T_collection`, checks whether timestamped c is an element of the collection.

For convenience, predicate $\mathit{IN}(\mathsf{o},\mathsf{c})$ is used to denote that object o is in clip c.

QUERY 2. Find the last clip in which person p appears:

$\{\mathsf{c} \mid \forall \mathsf{u}(\mathsf{u} \neq \mathsf{c} \wedge \mathit{IN}(\mathsf{p},\mathsf{u}) \wedge \mathit{IN}(\mathsf{p},\mathsf{c}) \wedge \mathsf{u}.\mathit{B\_timeStamp}.\mathit{B\_before}(\mathsf{c}.\mathit{B\_timeStamp}))\}$

where u and c are different clips. We compare timestamps of all the clips, in which p appears, with each other and choose the one which all others precede. An efficient way of processing this query is to perform the CVOT search depth-first but right-to-left. The first clip containing p is what we are looking for. For example, if we look for ken in Figure 3 we need only search C_5, N_3 and N_2.

QUERY 3. Are there any two clips in which object o simultaneously appears?

$$\begin{aligned}\{\mathsf{c}_1,\mathsf{c}_2 \mid\ & \mathsf{o}_1 \in \mathsf{c}_1.\mathit{B\_salientObjects} \wedge \mathsf{o}_2 \in \mathsf{c}_2.\mathit{B\_salientObjects} \wedge \mathsf{o}_1.\mathit{B\_value} = \mathsf{o} \wedge \mathsf{o}_2.\mathit{B\_value} = \mathsf{o} \wedge \mathsf{c}_1 \neq \mathsf{c}_2 \wedge \\ & \mathsf{o}_1.\mathit{B\_timeStamp}.\mathit{B\_during}(\mathsf{c}_1.\mathit{B\_timeStamp}.\mathit{B\_intersection}(\mathsf{c}_2.\mathit{B\_timeStamp})) \\ & \mathsf{o}_2.\mathit{B\_timeStamp}.\mathit{B\_during}(\mathsf{c}_1.\mathit{B\_timeStamp}.\mathit{B\_intersection}(\mathsf{c}_2.\mathit{B\_timeStamp}))\}.\end{aligned}$$

The tricky part of this query is in finding an overlap part of two neighboring clips. The temporal intersection operation *B_intersection* is perfect to accomplish this operation. Of course, object o must be within such an overlap. The CVOT can help the search to avoid checking every clip here. For example, if an internal (non-leaf) node has two leaf nodes (clips) and the CVOT can help the search to avoid checking every clip here. For example, if an internal (non-leaf) node has only two leaf nodes (clips) and does not have o as a common object, then there is no need to look at those two leaf nodes.

QUERY 4. Find a video clip in which *John* drives a car after he walked out of the table-tennis room:

$$\begin{aligned}\{\mathsf{c} \mid\ & \mathsf{p} \in \mathsf{c}.\mathit{B\_salientObjects} \wedge \mathsf{p}.\mathit{B\_value}.\mathit{B\_name} = \text{`}\mathit{John}\text{'} \wedge \mathsf{d} \in \mathsf{c}.\mathit{B\_activities} \wedge \mathsf{d}.\mathit{B\_value}.\mathit{B\_driver} = \mathsf{p}.\mathit{B\_value} \wedge \\ & \mathsf{w} \in \mathsf{c}.\mathit{B\_activities} \wedge \mathsf{w}.\mathit{B\_value}.\mathit{B\_walker} = \mathsf{p} \wedge \mathsf{w}.\mathit{B\_value}.\mathit{B\_walkFrom}(\mathsf{z}) \wedge \mathsf{o}.\mathit{B\_value}.\mathit{B\_driving}\}\end{aligned}$$

where p is an instance of timestamped `T_person`, d is an instance of timestamped `T_driveActivity`, w is an instance of timestamped `T_walkActivity` with one more behavior *B_walkFrom*, and z is an instance of timestamped `T_room` that represents the able-tennis room. *B_walkFrom* describes a walker walking out from some place. The behavior *B_meet* here guarantees that the drive activity occurs right after the walk activity.

6 RELATED WORK

There is significant current interest in modeling video systems. Gibbs et. al.[12] investigate timed streams as the basic abstraction for modeling temporal media using object-oriented technology. The *media element* in their model corresponds to video frames in ours. A *timed stream* is modeled by a finite sequence of tuples $< e_i, s_i, d_i >$, $i = 1, \ldots, n$, where e_i is a media element, s_i is the start time of e_i and d_i is its duration. Three general structuring mechanisms (interpretation, derivation, and composition) are used to model time-based media. We also model videos as time-based streams. However, the temporal operations are not well supported in their model.

AVIS (Advanced Video Information System),[13] which uses *association maps* to group salient objects and activities, is quite close to our model. A video stream is segmented into a set of frame-sequences $[x, y)$, where x is the start frame and y is the end frame. Based on the association maps, a *frame segment tree* is built to capture objects and activities occurring in the frame-sequences. Then two arrays are created: objectArray and activityArray. Each element of any array is an ordered linked list of pointers to nodes in the frame segment tree. It is shown that such a data structure results in efficient query retrieval. The CVOT model adopted part of this efficient indexing scheme, i.e. the CVOTArray is similar to objectArray. Although AVIS model is similar to CVOT, there are some fundamental differences:

- In CVOT, the segmentation of a video can be arbitrary in the sense that two neighboring clips could overlap as long as they are either weakly or perfectly ordered. However, in AVIS two neighboring clips must be consecutive, i.e., they must be perfectly ordered. This extension in the CVOT model is important because an activity may span multiple clips.
- The frame segment tree in AVIS is a binary tree and, in practice, it is made up of many *empty nodes* (nodes without any common objects or activities from its child nodes). This problem could result in deep binary trees. In CVOT, the tree is an arbitrary tree and only the root node's common object set is allowed to be empty. The major advantage of such a shallow tree is its small number of nodes which can result in significant space saving. The tradeoff could be the higher building cost of an arbitrary tree and more complex searching algorithms.
- Activities are not explicitly modeled in the CVOT because an activity may span multiple clips. Therefore, one part of an activity may be in one clip and the other part may be in another clip. This is particularly important if a video is segmented by a fixed time interval which is actually used in the AVIS prototype system.

Video Semantic Directed Graph (VSDG) is a graph-based conceptual video model.[3] The most important feature of the VSDG model is an unbiased representation of the information while providing a reference framework for constructing a semantically heterogeneous user's view of the video data. Using this model along with an object-oriented hierarchy, a new video system architecture is proposed which can automatically handle video data. The video semantic directed graphs are more complicated than our common video object trees without introducing any more capability. Furthermore, the VSDG model does not directly support range queries such as "*Find all the clips in which John appears.*"

Little and Ghafoor[14] have described a temporal model to capture the timing relationships between objects in composite multimedia objects, and mapped it to a relational database. This model forms a basis for a hierarchical data model and for temporal access control algorithms to allow VCR-like capabilities. A generalized *n-ary* structure is used to model spatial-temporal semantics of multimedia data.

OVID (Object-oriented Video Information Database)[5] is an object-oriented video model. It introduces the notion of a *video object* which can identify an arbitrary video frame sequence (a meaningful scene) as an independent object and describe its contents in a dynamic and incremental way. An architecture, called ViMod,[15] for a video object database based on video features is proposed. The design of this model is the result of studying the metadata characteristics of queries and video features.

7 CONCLUSIONS

A tree-based video model, called the CVOT model, is proposed for specifying the temporal semantics of video data. The major advantages of the CVOT model are the flexibility for video segmentation and the feasibility of automatic video feature extraction. A unique way of integrating the CVOT model into an ODBMS with rich temporal operations is presented. End users are allowed to explore the video object database from their perception of video contents through the object-oriented technology. Such a seamless integration brings a uniform interface to end users. Furthermore, we show that our system supports a broad range of temporal queries and the combination of the CVOT model and object-oriented technology results in an elegant video ODBMS.

For the temporal relationships to span multiple videos, some meta-information must be considered. We do not address this problem in this paper. There are two major directions for our future work on the CVOT model. The first is to combine the spatial and temporal relationships within a single model. This will add significant power to the video model and will enable spatio-temporal querying. The next step is to build a video query language with an underlying calculus and algebra based on the CVOT model.

ACKNOWLEDGEMENTS

This research is supported by a grant from the Canadian Institute for Telecommunications Research (CITR) under the Network of Centres of Excellence (NCE) program of the Government of Canada.

REFERENCES

1. D. Woelk and W. Kim. Multimedia information management in an object-oriented database system. In *Proc. of the 13th Int'l Conf. on Very Large Data Bases*, pages 319—329, Brighton, England, Sept. 1987.

2. S. Christodoulakis and L. Koveos. Multimedia information systems: Issues and approaches. In W. Kim, editor, *Modern Database Systems*, pages 318—337. Addison-Wesley, 1995.

3. Y. F. Day, S. Dagtas, M. Iino, A Khokhar, and A. Ghafoor. Object-oriented conceptual modeling of video data. In *Proc. of the 11th Int'l Conf. on Data Engineering*, pages 401—408, Taipei, Taiwan, 1995.

4. S. W. Smoliar and H. J. Zhang. Multimedia information systems. *IEEE Multimedia Systems*, 1(2):62—72, Summer 1994.

5. E. Oomoto and K. Tanaka. OVID: Design and implementation of a video-object database system. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):629—643, August 1993.

6. T. D. C. Little, G. Ahanger, R. J. Folz, J. F. Gibbon, F. W. Reeve, D. H. Schelleng, and D. Venkatesh. Digital on-demand video service supporting content-based queries. In *Proc. of the First ACM Int'l Conf. on Multimedia*, pages 427—436, Anaheim, CA, 1993.

7. M. T. Özsu, R. J. Peters, D. Szafron, B. Irani, A. Lipka, and A. Munoz. TIGUKAT: A uniform behavioral objectbase management system. *The VLDB Journal*, 4:100—147, 1995.

8. J. Z. Li, I. Goralwalla, M. T. Özsu, and D. Szafron. Video modeling and its integration in a temporal object model. Technical Report TR-96-02, Department of Computing Science, University of Alberta, January 1996.

9. I.A. Goralwalla, M.T. Özsu, and D. Szafron. Modeling Medical Trials in Pharmacoeconomics using a Temporal Object Model. *Computers in Biology and Medicine*, 1996. Accepted for publication - Special Issue on Time-Oriented Systems in Medicine.

10. J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of ACM*, 26(11):832—843, 1983.

11. R. Peters. TIGUKAT: A uniform behavioral objectbase management system. PhD thesis, Department of Computing Science, University of Alberta. Available as Technical Report TR-94-06, 1994.

12. S. Gibbs, C. Breiteneder, and D. Tscichritzis. Data modeling of time-based media. In *Proc. of ACM SIGMOD Int'l Conf. on Management of Data*, pages 91—102, Minneapolis, Minnesota, May 1994.

13. S. Adali, K. S. Candan, S. S. Chen, K. Erol, and V. S. Subrahmanian. The advanced video information system: Data structures and query processing. *ACM-Springer Multimedia Systems Journal*, 4(August):172—186, 1996.

14. T. C. C. Little and A. Ghafoor. Interval-based conceptual models for time-dependent multimedia data. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):551—563, August 1993.

15. R. Jain and A. Hampapur. Metadata in video database. *ACM SIGMOD RECORD*, 23(4):27—33, Dec. 1994.