A Graphical Object based Approach to Representing and Querying Video Data

Duc A. Tran, Kien A. Hua, Khanh Vu

School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816-2362, USA.
Email: {dtran, kienhua, khanh}@cs.ucf.edu

Modeling video data poses a great challenge since they do not have as clear an underlying structure as traditional databases do. We propose a graphical object-based model, called VideoGraph, in this paper. This scheme has the following advantages: (1) In addition to semantics of video individual events, we capture their temporal relationships as well. (2) The inter-event relationships allow us to deduce implicit video information. (3) Uncertainty can also be handled by associating the video event with a temporal Boolean-like expression. This also allows us to exploit incomplete information.

The above features make VideoGraph very flexible in representing various metadata types extracted from diverse information sources. To facilitate video retrieval, we also introduce a formalism for the query language based on path expressions. Query processing involves only simple traversal of the video graphs.

Keywords: Video content retrieval, video modeling, query language.

1. Introduction

We deal with the modeling aspect of VDBMSs [Elmagarmid (1997)] in this paper. Playing an important role in VDBMSs, this is the process of designing the high-level abstraction of raw video to facilitate various information retrieval and manipulation operations. It determines what features are to be used in the retrieval, and therefore in the indexing process. Other components, such as content analysis tools and query processing techniques, are also more or less dependent on it. A good model is essential to enabling a wider range of applications.

Much research has been done in the area of video modeling/retrieval based on audio-visual content, such as audio, color, texture and motion (e.g., [Zhong (1999), Rui (1999), Oh (2000), Ardizzone (1997), Chang (1997), Yoshitaka (1996), Dimitrova (1997)]). The advantage of this approach is that features can be extracted automatically. The low-level schemes, however, are very limited in expressing queries. For example, it would be difficult to ask for a video clip showing the sinking of the ship in the movie "Titanic" using only color, texture, and audio information. In contrast, video data models based on semantic content (e.g., [Swanberg (1993), Smith (1992), Jiang (1997)]) are capable of supporting more natural queries. They, however, must rely partially on manual annotation. A limitation of this approach is that semantic content can be ambiguous and context dependent. This problem can be controlled by limiting the context and providing multiple semantic descriptions for different types of applications.

We focus on the semantic level in this paper. In particular, we consider two types of video semantics:

1. Event Description: This type of description indicates the video segments that show a particular event. Some examples of event description are ["Ship colliding with iceberg", 35th minute - 37th minute] and ["Captain dying", 48th minute - 49th minute]. The first description indicates that the 2-minute video segment, from time 35th minute to time 37th minute, shows the scene of a ship colliding with an iceberg. Similarly, the second description indicates that the scene of the captain dying begins at the 48th minute and ends at the 49th minute of the video.

2. Inter-event Description: This type of description describes the temporal relationship between two events. Some examples of inter-event description are ["Ship collides with iceberg before it sinks "] and ["Captain dies after the ship sinks"]. This type of semantic information is not associated with any video segment. Instead, it states the temporal relationship between two events. As an example, such information can be obtained from the script. Under this circumstance, we can report on the order of various events, but not the exact locations of their occurrences in the video stream.

We note that each event mentioned in an inter-event description may or may not have an explicit event description. In the first inter-event description given above, only the "colliding" event has an event description (i.e., from 35^{th} minute to 37^{th} minute). This situation arises in practice since information extractors are not perfect. As an example, an extractor based on explicit models may recognize a ship and the "colliding" scene, but lacks the knowledge to determine the "sinking" event.

Existing semantic-level video data models [Swanberg (1993), Smith (1992), Hjelsvold (1994), Jiang (1997), Decleir (1999)] support only event, not the inter-event, descriptions. In this paper, we introduce a new model, called VideoGraph, which can accommodate both in one framework. This enables us to deduce implicit event descriptions, and therefore retrieve implicit video information as well. For instance, using the following metadata: ["Ship collides with iceberg before it sinks "], ["Captain dies after the ship sinks"], ["Ship colliding with iceberg", 35^{th} minute - 37^{th} minute], and ["Captain dying", 48^{th} minute - 49^{th} minute], we can imply the implicit temporal description ["Ship sinking", 37^{th} minute - 47^{th} minute] This implicit semantic allows us to answer queries such as "showing the scene of the sinking ship". Existing techniques based on only explicit descriptions would fail to process these queries. Our new capability is reminiscent of implicit information in a deductive database management system. To the best of our knowledge, video semantic implicity has not been exploited in literature.

Another new feature considered in our model is the flexibility to associate an event with a temporal Boolean-like expression. This enhancement allows us to handle uncertainty. For instance, we can associate the event "Jack went aboard the ship" with the temporal description "[5^{th} minute, 7^{th} minute] or [9^{th} minute, 10^{th} minute]." This expression indicates that the event must be in one of the two video segments. Such conditions occur when we infer implicit video semantics from incomplete semantics. Thus VideoGraph can also deal with incomplete information, in addition to implicit information. To facilitate video retrieval, we present a formal query language for VideoGraph. The language is an extension to relational calculus with path expressions and has both a clear declarative and operational semantics. It is simple yet powerful enough to allow formulation of complex queries. Query processing involves only simple graph traversal.

The remainder of this paper is organized as follows. We formally present the details of VideoGraph in Section 2. The query language formalism is described in Section 3. The algorithm for computing the implicit video information is presented in Section 4. We discuss related work in Section 5. Finally, Section 6 summarizes our approach and indicates some directions for future research.

3. Video Data Model

In this section, we introduce a video model called VideoGraph that provides the aforementioned features. Intuitively, a VideoGraph database is a set of edge labeled rooted graphs, each representing the semantics of a single video. Such a graph is a collection of nodes, each in turn representing a single event. Nodes are linked to each other based on their containment and temporal relationships [Allen (1983)]. The relationship between two nodes captures inter-event information involving the two corresponding events. A node may be associated with explicit temporal information or not. If not, its implicit information can still be obtained by reasoning on the graph. We will discuss this in section 4.

VideoGraph is built on the concept of objects, each appearing as an internal node in the video graph. Prior to defining them, let us assume that **ATYPE** denotes the set of all integers, real numbers and strings. Let **TYPE** be a finite set containing special strings classified as data

types in the video database, in other words, **TYPE** = {$type_1$, $type_2$, .., $type_p$, **TIME**} where $type_i$ is a string. Each type $type_i$ has a value domain, denoted as dom($type_i$), such that dom($type_i$) ⊆ **ATYPE** if i ≤ p, and dom(**TIME**) = Ω that is defined later. In what follows, unless we explicitly mention, concepts to be defined are considered within the same context of a single video.

Definition 2.1. [Atomic object] *Let* l_1, l_2 , .., l_n *(*n ≥ 1*) be strings in* **TYPE**, v_1, v_2 , .., v_n *values such that* v_i ∈ dom(l_i). *Let us consider a graph* G *containing a node* O, *called the* root *of* G, *and* n *nodes* O_1 , O_2 , .., O_n *with the following properties:*

- *Node* O *stores a unique integer value.*
- *For each* i ∈ {1, 2, .., n}, *node* O_i *stores value* v_i.
- *For each* i ∈ {1, 2, .., n}, *there is a directed link labeled* l_i *from* O *to* O_i.

Then node O *is said to be the* atomic object *represented by graph* G *and the value stored in the node is called the* identifier *of the atomic object.*

Definition 2.2. [Object] *Objects are recursively defined as follows.*

1. *Any atomic object is an object.*
2. *Let* G_1, G_2, .., G_n *represent* n *objects,* O_1, O_2 , .., O_m *be* m *single nodes (*n +m > 0*) and* l_1, l_2, .., l_{n+m} *be* n + m *strings in* **TYPE**. *Then the graph* G *built below represents an object.*
   - G *contains a node* O, *which is called the* root *of* G, *nodes* O_i*'s and graphs* G_j*'s for* i ∈ {1, 2, .., m} *and* j ∈ {1, 2, .., n}.
   - *Node O stores a unique identifier for the object.*
   - *For each* i ∈ {1, 2, .., n}*, there is a directed link from* O *to the root of* G_i *labeled* l_i.
   - *For each* i ∈ {1, 2, .., m}*, there is a directed link from* O *to* O_i *labeled* l_{i+n} *and node* O_i *stores a value* v_i ∈ dom(l_{i+n})

   We can say that node O *is the* object *represented by graph* G.

Here after, for flexibility, the terms object and internal node are interchangeably used. That is, we implicitly refer to an object as an internal node and vice versa.

Links between any two nodes in the above definitions are called *c-links*. If a node O_1 has a c-link labeled l departing from it and going to another node O_2, then l is a *component* of O_1 and the *type* of node O_2 with respect to O_1. O_2 is the value of component l of O_1. Components of an object can be duplicated. Furthermore, if O_2 is an internal node, it is also called a *sub-object* of O_1 which in turn is said to be a *super-object* of O_2.

We divide the set of objects into two classes, key objects and non-key objects. Informally, a key object is an object that has temporal related information telling what parts of the video associate with the object. That can be complete or incomplete (that is, the exact video segment for an event is not known). Before giving the formal definition of a key object, we need to describe a new type for video temporal values.

Definition 2.3. [I-expression] I-expressions *(*i *stands for "interval") are defined as follows:*

1. *For any* t_1 *and* t_2 *integers (*$t_1 \le t_2$*),* $[t_1, t_2]$ *is an i-expression. It is also classified as an* interval.
2. *If* p *and* q *are two i-expressions, then so are (*p & q*) and (*p | q*). The meaning of operations "&" and "|" is that if an object associates with an i-expression* p & q *(or* p | q*), then it associates with both (or one) of* p *and* q.

Let Ω denote the set of all i-expressions. We recall that dom(**TIME**) = Ω. I-expressions tell how to look up the video and they can be used to express incomplete information. For instance, [15, 18] & [25, 30] corresponds to a set of two video segments, one represented by [15, 18] and the other one by [25, 30]; [25, 28] | [15, 20] corresponds to only one video segment, [25, 28] or [15, 20], but it is not known to be which.

Definition 2.4. [Key object] *If* O *is an object having* **TIME** *as a component and its corresponding* **TIME** *value is an i-expression, then* O *is categorized as a* key object.

A video graph is built on objects each corresponding to a single event in reality and their temporal relations reflects inter-event descriptions. An inter-event relation can be described as an element of the set **REL** = {ABBD, ABCD, ACBD, ACDB, AABD, AABB, ACBB}. Event I [a, b] has a relationship r with event II [c, d] (a ≤ c) if and only if (1) b = c: r = ABBD, (2) b

< c: r = ABCD, (3) c < b < d: r = ACBD, (4) d < b: r = ACDB, (5) a = c and b < d: r = AABD, (6) a = c and b = d: r = AABB, (7) c < b = d: r = ACBB.

Definition 2.5. [Video graph] *Given a video* V, *let graphs* G_1, G_2 , .., G_n *(*$n \geq 1$*) represent its objects, that are not sub-objects of any others, the* video graph *of* V *is an edge labeled rooted graph* G *defined as follows.*

- *The root* A *of the graph stores the identifier of* V.
- G *contains every graph* G_i *for* $i \in \{1, 2, .., n\}$.
- *For each* $i \in \{1, 2, .., n\}$*, there is a link from* A *to the root of* G_i *labeled* **SEM**.
- *If two internal nodes* O_1 *and* O_2 *of* G *have a temporal relation* $r \in$ **REL**, *then there is a directed link from* O_1 *to* O_2 *labeled* r *and classified as an* r-link.

A *video database* consists of a number of video graphs, each representing knowledge about an individual video in the database.

The VideoGraph model encompasses both video data and the structure of them. Before going any further, let us give an example of a video database. We are interested in a single movie and the video graph for it is shown in Figure 1 where directed solid lines describe c-links and dotted curves describe r-links. In this video graph, the atomic objects are o_4, o_7, o_9, o_{10}, o_{11} and o_{12}. The key objects are o_1, o_3, o_5, o_8 and o_{10}. The others are non-key objects. Note that o_{10} has incomplete temporal information ([12, 30] | [20, 40]) because it is not known that node o_{10} associates with which interval, [12, 30] or [20, 40]. VideoGraph allows different kinds of temporal information, which are encapsulated in one type Ω (i-expressions). This distinct property was not possible in previous models. Hence, they have limited capabilities for utilizing semantics extracted from diverse and dynamic knowledge sources because all events that do not have a temporal description or that have incomplete temporal information are not considered.

The VideoGraph model can be considered an extension to OEM model (Object Exchange Model) [Papakonstantinou (1995)] for semi-structured data [Papakonstantinou (1995), Abiteboul (1997), Buneman (1997)]. However, while OEM was designed for general purposes, our VideoGraph deals with the problem of organizing semantic contents of videos in a way such that it helps users better retrieve and query the video in a special manner. VideoGraph differs from OEM in three considerable ways. First, in a video graph, the same object may appear at multiple places regarding its possible multiple occurrences in the video. Second, the temporal relations among VideoGraph objects are taken into account. They are represented by r-links in the video graph. Finally, in terms of data access, queries in video databases are essentially to search for part of the video satisfying some semantics or to obtain the semantic contents in a video segment. In comparison with previous video data models, VideoGraph has the following distinguishing features: (1) it is able to capture inter-event descriptions represented by r-links in the video graph; (2) the expression of incomplete information is made possible by using i-expressions; (3) non-key objects are stored in the database as key objects, whereas in other models only key objects are considered. Exceptionally, it allows the deduction of implicit information.

3. Query Language

We will now introduce a content-based mechanism to support users' querying the video database. We begin by clarifying what kinds of outputs are allowed in the querying system. We focus on answers of the following types: (1) Video segments: When the user searches for video segments that satisfy some semantic constraint. (2) Semantic contents: When the user asks for information about a video clip.

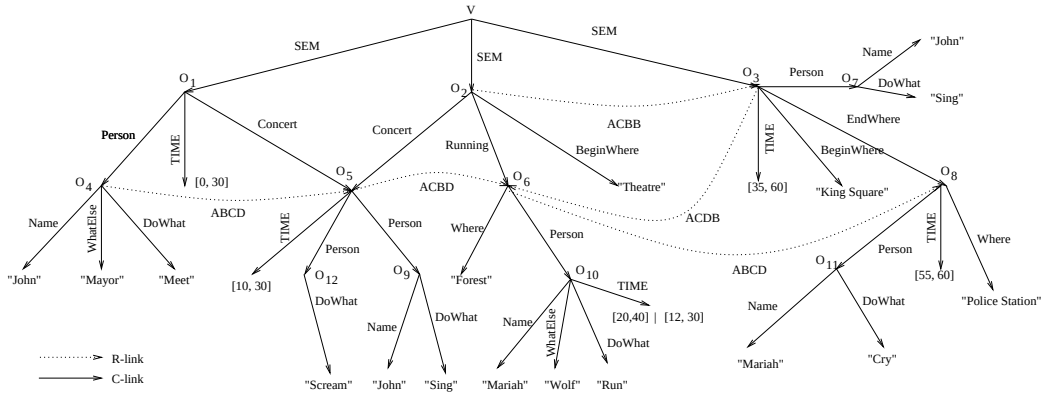

Figure 1. An example of video graphs

We limit our queries to retrievals only, however update queries can be easily embedded. The language is declarative and allows the user to describe the set of answers without being explicit about how they should be computed. Our video query language is based on path expressions formulated according to the graphical structure of the database. The queries are also expressed on the inter-object temporal relationships. Before formally presenting what the syntax and semantics of queries are, we give some necessary definitions.

Without loss of generality, we consider only one video graph in the database which we are going to formulate queries on. Let o_1 and o_2 be two nodes and L labeled l be a link from o_1 to o_2. Then we can express o_1 = **from(**L) and o_2 = **to(**L).

Definition 3.1. [Strict Path Sample] *A* strict path sample *is of the form* $l_1 \rightarrow l_2 \rightarrow \ldots \rightarrow l_n$ *where* $l_1, l_2, .., l_n \in$ **TYPE** *and* $\forall$ i $\in$ {1, 2, .., n-1}, *there exist two c-links* L_i *and* L_{i+1} *labeled* l_i *and* l_{i+1} *respectively such that* **to(**L_i) = **from(**L_{i+1}).

Definition 3.2. [Path Sample] *Let* θ *be either* $\rightarrow$ *or* $\rightarrow\rightarrow$ *and* $l_1, l_2, .., l_n \in$ **TYPE**. *A* path sample *is of the form* $l_1\ \theta\ l_2\ \theta\ ..\ \theta\ l_n$ *where there exist components* $l_{i1}, l_{i2}, .., l_{iki}$ (i $\in$ {1, 2, .., n-1}) *such that* $l_1 \rightarrow l_{11} \rightarrow l_{12} \rightarrow .. \rightarrow l_{1k1} \rightarrow l_2 \rightarrow l_{21} \rightarrow l_{22} \rightarrow .. \rightarrow l_{2k2} \rightarrow .. \rightarrow l_n$ *is a strict path sample.*

Our query language is based on the expression of paths. We allow the user to formulate any path he or she is interested in. The path may or may not exist in the graph, in other words, it either conforms to a path sample or does not conform to any. In order to facilitate users' querying, we introduce the notion of path expression.

Definition 3.3. [Path Expression] *Let* O *be an internal node,* l *be a label,* θ *be either* $\rightarrow$ *or* $\rightarrow\rightarrow$. *A* path expression *is recursively defined as follows: (1)* O. *is a trivial path expression, which contains only a node; (2) If* α *is a path expression, then so are* $\alpha\theta$ *and* $\alpha\theta$(O).

A query will be run successfully if it contains path expressions, each conforming to some path sample of the video graph. Otherwise, the query returns nothing. For the semantics of the query, we define the validity property of a path expression below.

Definition 3.4. [Path Validity] *Let* θ *be either* $\rightarrow$ *or* $\rightarrow\rightarrow$. *The validity of a path expression is presented as follows:*

1. Any trivial path expression is a valid path expression
2. *A non-trivial path expression* PE *is valid if the conditions below hold:*

- *If* O. θ l *appears in* PE, *then there must exist a c-link* L *labeled* l *such that* O = **from(**L)
- *If* $l_1\ \theta\ l_2$ *appears in* PE, $l_1\ \theta\ l_2$ *must be a path sample*
- *If* l_1(O) θ l_2 *appears in* PE, *then* $l_1\ \theta\ l_2$ *must be a path sample and there must be two c-links,* L_1 *labeled* l_1 *and* L_2 *labeled* l_2, *such that* O = **to(**L_1) = **from(**L_2*)*
- *If* $l_1\ \theta\ l_2$(O) *appears in* PE, $l_1\ \theta\ l_2$ *must be a path sample and there must exist a c-link* L *labeled* l_2 *such that* O = **to(**L)

Having presented the above concepts, we are ready to give a description of queries and their semantics. We will see that the basic construct in the query language is an expression of the form:

(**TIME**) ⇐ *PathExpression, StartNode, SelectionCondition* (1)

or

(**SEM**) ⇐ *PathExpression, StartNode, SelectionCondition* (2)

where **TIME** and **SEM** are called the *filters* of the query, *PathExpression* is a path expression, *StartNode* is a node (a node identifier to be exact), the starting node in the graph to traverse and look for the result of the query. It can be the root or any internal node. *SelectionCondition* denotes a formula describing the condition that the answer must satisfy. We will shortly define conditions and queries rigorously. The result of this query is, based upon which filter is used, the i-expression value of the object or a set of objects represented by *PathExpression* for which the formula *SelectionCondition* evaluates to true.

The language for writing formula *SelectionCondition* is the heart of our query formulation. Our queries use three salient operators, ∇, ⊕ and ⊗. Given a node A, ∇(A) gives the set of all nodes that are reachable from it by traversing the graph (Node B is said to be reachable from node A if and only if there exists a valid path expression starting with A and ending with B). ⊕(A, B) returns the temporal relation between two internal nodes A and B. ⊗(A, ie) returns the temporal relation between the **TIME** value of internal node A and an i-expression ie. The result of ⊕ and ⊗ operations must be an element of the set **REL**.

3. 1 Syntax of VideoGraph queries

Definition 3.5. [Atomic Condition] *Let assume that* Θ *is an operator in the set {<, =, >, ≥, ≤},* PE *and* PE*' are path expressions,* o_1 *and* o_2 *are internal nodes,* ie *is an i-expression,* r ∈ **REL**, v ∈ **ATYPE**. *An atomic condition has one of the forms: (1)* PE*; (2)* PE Θ v*; (3)* PE Θ PE*'; (4)* ⊕*(*PE, PE*')* = r*; (5)* ⊗*(*PE, ie*)* = r*; (6)* $o_1 \in \nabla(o_2)$*.*

Definition 3.6. [Condition] *Let* p *and* q *be themselves conditions,* f(O) *be a condition in which* O *appears, a* condition *is recursively defined to be one of the following: (1) any atomic condition; (2)* ¬p, p ∧ q, p ∨ q, *or* p ⇒ q*; (3)* ∃O(f(O)), *where* O *is a variable representing an internal node; (4)* ∀O(f(O)), *where* O *is a variable representing an internal node*

In this definition, ∃ and ∀ are two quantifiers in traditional logic and are said to *bind* to the variable O.

Definition 3.7. [Free variable] *A variable is said to be* free *in a condition or a sub-condition (a condition contained in a larger condition) if the (sub-)condition does not contain an occurrence of a quantifier that binds it.*

Now is time for the formal syntax of a VideoGraph query.

Definition 3.8. [VideoGraph Query] *A* VideoGraph query *is defined as an expression of the form:*

*(***TIME***)* ⇐ PE(O_1, O_2, .., O_n), SN, SC(O_1, O_2, .., O_n) (3)

or

*(***SEM***)* ⇐ PE(O_1, O_2, .., O_n), SN, SC(O_1, O_2, .., O_n) (4)

where SN *is a predefined node,* O_i*'s (*i = 1..n*) are internal node variables and the only free variables in the formulas* SC(O_1, O_2, .., O_n) *and* PE(O_1, O_2, .., O_n), SC(O_1, O_2, .., O_n) *is a condition containing one or more occurrence of each* O_i , PE(O_1, O_2, .., O_n) *is a path expression containing one or more occurrence of each* O_i, **TIME**, **SEM** *are special symbols describing what kinds of output are to be returned, an i-expression (temporal information) or a set of nodes (objects).*

3. 2 Semantics of VideoGraph queries

The answer to a VideoGraph query (**SEM** | **TIME**) ⇐ PE(O_1, O_2, .., O_n) , SN , SC(O_1, O_2, .., O_n), as we noted earlier, is the set of all objects (graph nodes) if the filter is **SEM** or an i-expression otherwise that is obtained by computing PE(o_1, o_2, .., o_n) where o_1, o_2 , .., o_n, assigned to O_1, O_2, .., O_n respectively, make SC(O_1, O_2, .., O_n) evaluate to true. To complete this definition, we must state which value assignments to free variables in a condition make the condition true.

A query is evaluated in any given instance of the video database. Let each free variable O_i in a condition $SC(O_1, O_2, .., O_n)$ (we call it F for brevity) be bound to a value o_i (a node in the graph). With respect to the video database and for the assignments of values to variables, the condition F must be true if one of the following holds:

- F is an atomic condition PE, and PE is a valid path expression.
- F is an atomic condition PE Θ v, and PE is a valid path expression which by traversing we can obtain a node value v' (an atomic value or an object identifier) that makes the comparison v' Θ v true.
- F is an atomic condition PE Θ PE', and PE, PE' are valid path expressions which by traversing we can obtain two nodes whose values, v_1 and v_2, make $v_1 \Theta v_2$ true.
- F is an atomic condition $\oplus$(PE, PE') = r, and PE, PE' are valid path expressions which by traversing we can obtain two internal nodes such that their implicit or explicit **TIME** values are related to each other by relation r.
- F is an atomic condition $\otimes$(PE, ie) = r, and PE is a valid path expression which by traversing we can obtain an internal node whose implicit or explicit temporal relationship with the i-expression ie is equivalent to relation r.
- F is an atomic condition $o_1 \in \nabla(o_2)$, and o_1 is reachable from o_2.
- F is of the form $\neg$p, and p is not true; or of the form p $\wedge$ q, and both p and q are true; or of the form p $\vee$ q, and one of them is true; or of the form p $\Rightarrow$ q, and q is true whenever p is true.
- F is of the form $\exists$O(f(O)), and there is some assignment of values to the free variables in f(O) and variable O, that makes it true.
- F is of the form $\forall$O(f(O)), and there is some assignment of values to the free variables in f(O) that make it true no matter what value is assigned to variable O.

Now we need to be clear how the answer of a query is returned, that is, we need to formally define what the semantics of $PE(O_1, O_2, .., O_n)$ is, given an assignment of values to variables $O_1, O_2, .., O_n$. The query returns nothing if PE is not a valid path expression. Otherwise, PE represents a set of nodes that are obtained by traversing the video graph based on PE. We call those nodes instances of the path expression PE.

Definition 3.9. [Instances] *An object* O *is an* instance *of a path expression* PE *if one of the following holds:*

- PE *is* O.
- PE *is* PE' $\rightarrow$ l, *and* $\exists$O' *an instance of path expression* PE' *and a c-link* L *labeled* l *such that* O' = **from(**L) *and* O = **to(**L).
- PE *is* PE' $\rightarrow\!\!\rightarrow$ l, *and* $\exists$O' *an instance of path expression* PE' *and a c-link* L *labeled* l *such that* O = **to(**L) *and* **from(**L) $\in \nabla$(O').
- PE *is* PE' $\rightarrow$ l(O), *and* $\exists$O' *an instance of path expression* PE' *and a c-link* L *labeled* l *such that* O' = **from(**L) *and* O = **to(**L).
- PE *is* PE' $\rightarrow\!\!\rightarrow$ l(O), *and* $\exists$O' *an instance of path expression* PE' *and a c-link* L *labeled* l *such that* O = **to(**L) *and* O = **from(**L) $\in \nabla$(O').

Since the query outputs can be of two types, an i-expression or a set of node identifiers from which semantic contents are withdrawn, we consider the following cases: (1) **SEM** filter: The answer to a query is a set of objects (node identifiers), each being an instance of the path expression $PE(O_1, O_2, .., O_n)$ where O_i's $\in \nabla$(SN) and O_i's make the condition $SC(O_1, O_2, .., O_n)$ true. (2) **TIME** filter: The answer to a query is an i-expression which is computed by applying "&" operation on the **TIME** values of all the instances of the path expression $PE(O_1, O_2, .., O_n)$ where O_i's $\in \nabla$(SN) and O_i's make the condition $SC(O_1, O_2, .., O_n)$ true.

4. Implicit Information Inference

Having presented the video model and its related formal query language, one issue left is how to compute the implicit information from a VideoGraph database. This can be considered a preprocessing refinement phase. In our VideoGraph model, an internal node may or may not have a **TIME** link from it. If not, its temporal information can still be obtained by traversing the graph taking into account r-links to other nodes that have a temporal value. In other

words, we are able to obtain implicit complete semantics from event descriptions, inter-event descriptions and incomplete information. In what follows, we introduce a simple way of how to do it. The algorithm converts an instance of the video data model to another called refined graph. The refined graph has direct temporal features associated with the internal nodes (that is, each node has a **TIME** link and a **TIME** value). Now we define what a refined graph is and shortly introduce a simple algorithm to compute the refined graph of a given video graph.

Given a single video, its video graph can be represented as a tuple G = (V, e, f, g, h) where V is the set of nodes; e: $V \rightarrow$ {INTERNAL, LEAF}, a unary function returning the type of each node; f: $V \rightarrow$ **ATYPE** $\cup \Omega$ a unary function returning the value stored in each node; g: $V \times V \rightarrow$ **TYPE** $\cup$ VOID, returning the label of the link from one node to another, VOID if there is no c-link between them; h: $V \times V \rightarrow$ **REL** $\cup$ VOID, returning the temporal relationship between one node to another, VOID if there is no r-link between them.

Definition 4.1. [Refined Graph] *Given a video graph* G = (V, e, f, g, h) *of a video. Its* refined graph *is also a VideoGraph* $G_1 = (V_1, e_1, f_1, g_1, h_1)$ *with the following properties.*

- $V \subseteq V_1$
- $card(V_1) = card(V) + card(V_2)$ *where* $V_2 = \{v \in V \mid e(v) = \text{INTERNAL} \wedge \forall v_1 \in V: g(v, v_1) \neq \textbf{TIME}\}$
- *For each* $v \in V_2$*, there exists only a node* $v_1 \in V_1 - V$ *such that* $g_1(v, v_1) =$ **TIME**. *Conversely, for each* $v_1 \in V_1 - V$*, there is only a node* v *of* V_2 *such that the above condition holds.*
- $e_1(v) = e(v)$ *if* $v \in V$, LEAF *otherwise*
- *If* $v, v_1 \in V$*, then* $f_1(v) = f(v)$, $g_1(v, v_1) = g(v, v_1)$
- *For* $v, v_1 \in V_1$ *such that* g_1 *is not yet defined for,* $g_1(v, v_1) = \text{VOID}$
- $h_1(v, v_1) = \text{VOID}\ \forall v, v_1 \in V_1$
- *If* $v_1, v_2 \in V$ *such that* $e(v_1) = e(v_2) = \text{INTERNAL}$, *and* $v_1', v_2' \in V_1$ *such that* $g_1(v_1, v_1') = g_1(v_2, v_2') =$ **TIME**, *then the relationship between* $f_1(v_1)$ *and* $f_1(v_2)$ *must not conflict with* $h(v_1, v_2)$.

The corresponding refined graph of the video graph in Figure 1 is in Figure 2. We note that node o_{10} now has a more meaningful **TIME** value, which tells that it associates with i-expression [36, 40], not like in the source video graph where o_{10}'s certain temporal information was unknown. Now comes an algorithm to determine the refined graph of a video graph.

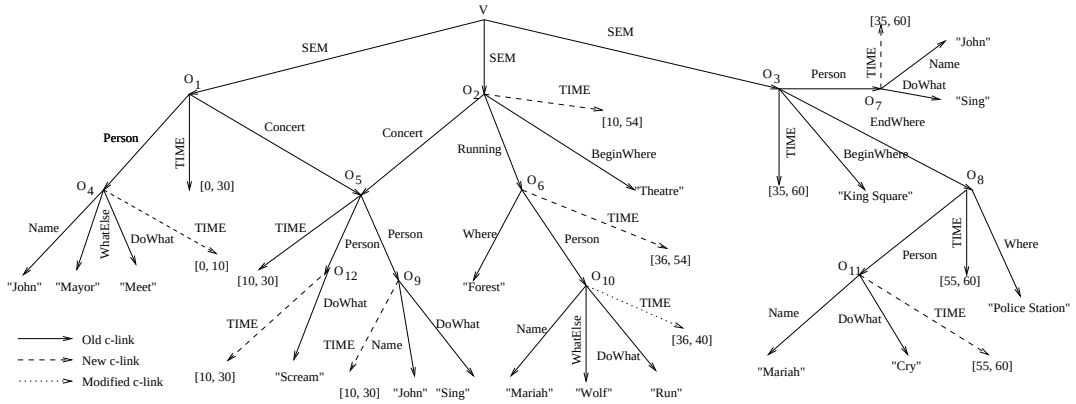

Figure 2. An example of refined graphs

Algorithm 4.1. [Refinement Algorithm] *Given a video graph* G = (V, e, f, g, h) *for a video. The corresponding refined graph is built as the following steps.*

1. V_2 *is initialized to the set of all non-key objects in the source video graph.*
2. *For each node* $v \in V_2$*, add a new node storing [0, ∞] and add a link labeled* **TIME** *from* v *to the new node.*
3. *Initialize* UpdateCounter *and* UpdateCounter_1 *to 0.*

4. *For any two internal nodes* v *and* v' *that are connected by an r-link, adjust the value of their* **TIME** *node so as to satisfy* h(v, v'). *If a node is a key object with a complete* **TIME** *value, do not change the value. If one of the values is changed, increase* UpdateCounter_1 *and* UpdateCounter *both by 1.*

5. *If* UpdateCounter1 $\neq 0$*, go back to step 3.*

6. *For any two internal nodes* v *and* v' *such that the former is a sub-object of the latter, adjust the value of their* **TIME** *node so that the* **TIME** *values of* v *and* v' *are related by the relation* ACDB *(containment relationship). If a node is a key object with a complete* **TIME** *value, do not change the value. If one of the values is changed, increase* UpdateCounter_1 *and* UpdateCounter *by 1.*

7. *If* $\text{UpdateCounter}_1 \neq 0$*, assign it to 0 and go back to step 6.*

8. *If* UpdateCounter $\neq 0$*, go back to step 3.*

9. *Remove all the r-links resulting in a new graph, which is the refined graph of* G.

In a refined graph, all the r-links have been removed. It has another property that every object (internal node) has a temporal descriptor which is captured by applying the refinement algorithm above on the source video graph. Thus whenever a node is visited for its temporal information, no further graph traversal is needed. The merit of the algorithm is that it helps reduce the overhead of query processing. For example, if we only have the source video graph (i.e., without applying the refinement algorithm) as in Figure 2 and if the user very often wants to watch the scene associated with object o_{10}, then the query system will have to compute the implicit temporal description of o_{10} many times back and forth.

In environments where most objects in the video database are accessed with high frequency, it is a good idea to build the refined graph just once, ahead of time, and store it in the database. Whenever the video graph is updated, its corresponding refined graph is recalculated. Subsequent query processing steps will be taken on the refined graph, resulting in more processing overhead being reduced. However, it is not always necessary to save it permanently, especially if we take into account the cost of storing an additional graph. Depending upon the context where the video database is, we have to consider the tradeoff between the efficiency of using the refined graph and the storage cost charged. From that standpoint, we can decide to compute it, whether or not on the fly, as the user questions the video.

5. Related Work

This paper deals with video data modeling. As we have mentioned in Section 1, there are basically two major approaches, physical feature-based and semantic content-based. The choice depends on the purpose and use of the video data. In an application like astronomy VDBMS, the motion information of stars is the most important content of the video data. On the other hand, applications such as digital libraries, semantic contents are necessary for the user to retrieve information. In this section, we briefly discuss some related semantic-based models.

In the past several years, many of them first segment the video stream into a set of temporally ordered shots, and then build a multi-level abstraction upon these shots. This approach is referred to as the segmentation-based model. One such scheme was proposed in [Swanberg (1993)]. This technique identifies the type of each shot using domain-specific shot models. The typed shots are then grouped into bigger units at the next higher level in the hierarchy by matching the higher-level models to the typed shots. This procedure can be applied recursively until we obtain a single unit that represents the entire video. Instead of relying on explicit models, Chua and Ruan proposed to describe each video segment using natural language [Chua (1995)]. They, however, use keywords to facilitate video retrieval. Some other segmentation-based models are presented in [Ardizzone (1997), Hampapur (1995), Zhang (1992), Gupta (1991)].

A drawback of video segmentation-based models is lack of flexibility. Smith and Davenport et al. [Smith (1992)] proposed a layered annotation representation model called the stratification model. This scheme segments contextual information of the video instead of

simply partitioning the video stream. The video units, called stratum, can overlap or encompass each other. This approach approximates the movie editor's perspective on a movie. Some other stratification-based models are as follows. Jiang et al introduced a VideoText model in [Jiang (1997)]. This model is based on free text annotations rather than a fixed set of keywords to index the strata. Text retrieval techniques are used to provide content-based access to the video database. A stratification scheme based on a video algebra is presented in [Weiss (1994)]. The fundamental entity of this model is a presentation. A presentation is a multi-window spatial, temporal, and content combination of video strata. Presentations are described by video expressions, which are constructed from basic video strata using video algebraic operations. Another stratification-based model is presented in [Adali96]. They associate strata with events in the video.

A video object model is used in a prototype named OVID developed by Oomoto and Tanaka [Oomoto (1993)]. In this model, a video object is defined as an arbitrary sequence of video frames. Each video object consists of a unique identifier, an interval represented by its starting and ending frame numbers, and a collection of attribute-value pairs describing the content of the frame sequence. Arbitrary attributes can be attached to each video object if necessary. Also, interval inclusion inheritance is applied to ease the effort of providing description data when an existing video is composed into new video objects using the generalization hierarchy concept. Inclusion inheritance enables these video objects to share their descriptive data.

6. Conclusions

Video content is very complex. No single information extraction scheme is a panacea. Instead, we have to rely on multiple techniques to extract meta information from different knowledge sources (e.g., script, caption) in order to overcome the errors of any one method. To support such an environment, the video data model must be able to accommodate metadata obtained by different extraction tools. Furthermore, it should allow the deduction of implicit information from these otherwise independent metadata types. This concept of semantic implicity has not been considered previously. To provide this desirable capability, we introduced in this paper a new video data model called VideoGraph. This model can capture not only descriptions of individual events, but also their inter-event relationships. To fully benefit from these two types of semantics, we provide an algorithm to compute the implicit information from the initial metadata.

Another contribution of this paper is the support for uncertainty. This arises because information extraction tools usually fail to produce the complete metadata set and reasoning on incomplete information often results in uncertain information. To address this, VideoGraph associates each event with an i-expression. For instance, a disjunctive (“|”) form can be used to indicate that the video event must occur in one of the listed video segments. Our scheme hence is more intelligent and expressible than existing techniques. To facilitate video retrieval, we also presented in this paper a declarative query language based on path expressions. To the best of our knowledge, it is the first query language of this kind. Path expressions are easy to use yet powerful enough to allow the expression of fairly complex video queries.

References

Abiteboul, S. (1997). Querying semi-structured data. *In Proc. of IEEE Int’l Conference on Data Engineering (ICDE).*

Adali, S., Candan, K. S., Chen S. S., Erol, K. and Subrahmanian, V. S. (1996). Advanced video information system. Data structures and query processing. *ACM-Springer Multimedia Systems Journal*. vol. (7), 172-186.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM, Springer-Verlag*, vol. 26(11), 832-843.

Ardizzone, E. and Cascia, M. L. (1997). Automatic video database indexing and retrieval. *Journal of Multimedia Tools and Applications*, vol. 4(1), 29-56.

Buneman, P. (1997). Adding structure to unstructured data. *In Proc. of IEEE Int'l Conference on Data Engineering (ICDE)*, January.

Chang, S.-F., Chen, W., Meng, H., Sundaram, H. and Zhong, D. (1997). Videoq. An automated content based video search system using visual cues. *In Proc. of ACM Int'l Conference on Multimedia*, November.

Chua, T. -S. and Ruan, L. Q. (1995). A video retrieval and sequencing system. *ACM Transactions on Information Systems*, vol. 13(4), 373-407.

Decleir, C. and Hacid, M. –S. (1999). A database approach for modeling and querying video data. *In Proc. of IEEE Int'l Conference on Data Engineering (ICDE)*, 6-13, Australia.

Dimitrova, N., McGee, T. and Elenbaas, H. (1997). Video keyframe extraction and filtering. A key frame is not a keyframe to everyone. *In Proc. of ACM Int'l Conference on Information and Knowledge Management (CIKM)*.

Elmagarmid, A. K., Jiang, H., Helal, A. A., Joshi, A. and Ahmed, M. (1997). *Video Database Systems: Issues, Products, and Applications*. Kluwer Academic Publishers.

Gupta, A., Weymouth, T. and Jain, R. (1991). Semantic queries with pictures. The VIMSIS model. *In Proc. of 17th Int'l Conference on Very Large Databases (VLDB)*, September.

Hampapur, A., Jain, R. and Weymouth, T. (1995). Production model based digital video segmentation. *Journal of Multimedia Tools and Applications*, vol. 1(1), 9-46.

Hjelsvold, R. and Midstraum, R. (1994). Modeling and querying video data. *In Proc. Of the 20th Int'l Conference on Very Large Databases (VLDB)*, 686-694.

Jiang, H., Montesi, D. and Elmagarmid, K. (1997). Videotext database systems. *In Proc. of IEEE Int'l Conference on Multimedia Computing and Systems*, Ontario, Canada, June.

Oh, J. and Hua, K. A. (2000). An efficient and cost-effective technique for browsing, querying and indexing large video databases. *In Proc. of ACM Int'l Conference on Management of Data (SIGMOD)*, Dallas, TX, May.

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

Papakonstantinou, Y., Garcia-Molina, H. and Widom, J. (1995). Object exchange across heterogeneous information sources. *In Proc. of IEEE Int'l Conference on Data Engineering (ICDE)*, March.

Rui, Y., Huang, S. and Mehrotra, S. (1999). Constructing table-of-content for videos. *ACM Springer-Verlag Multimedia Systems Journal*, vol. 7(5), 409-423.

Smith, T. G. A and Davenport, G. (1992). The stratification system. A design environment for random access video. *In Proceedings of the 3rd Int'l Workshop on Network and Operating System Support for Digital Audio and Video*, La Jolla, CA.

Swanberg, D., Shu, C. –F. and Jain, R. (1993). Knowledge guided parsing in video databases. *In Proc. of IS&T/SPIE Conference on Image and Video Processing*, vol. 1908, 13-24. San Jose, CA, February.

Weiss, R., Duda, A. and Gifford, D. (1994). Content-based access to algebraic video. *In Proc. of IEEE Int'l Conference on Multimedia Computing and Systems*, Boston, USA.

Yoshitaka, A., Hosoda, Y., Yoshimisu, M., Hirakawa, M. and Ichikawa, T. (1996). Violone. Video retrieval by motion example. *Journal of Visual Languages and Computing*, vol. 7, 423-443.

Zhang., H. J., Gong, Y., Smoliar, S. W and Tan, S. Y. (1992). Automatic parsing of news video. *In Proc. of IEEE Int'l Conference on Multimedia Computing Systems*, May.

Zhong, D. and Chang, S. –F. (1999). An integrated approach for content-based video object segmentation and retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9(8), 1259-1268, December.