A Query Model to Synthesize Answer Intervals from Indexed Video Units

Sujeet Pradhan, *Member, IEEE*, Keishi Tajima, and Katsumi Tanaka, *Member, IEEE*

***Abstract*—While a query result in a traditional database is a subset of the database, in a video database, it is a set of subintervals extracted from the raw video sequence. It is very hard, if not impossible, to predetermine all the queries that will be issued in future, and all the subintervals that will become necessary to answer them. As a result, conventional query frameworks are not applicable to video databases. In this paper, we propose a new video query model that computes query results by dynamically synthesizing needed subintervals from fragmentary indexed intervals in the database. We introduce new interval operations required for that computation. We also propose methods to compute relative relevance of synthesized intervals to a given query. A query result is a list of synthesized intervals sorted in the order of their degree of relevance.**

***Index Terms*—video database, video retrieval, continuous data, indexing units, interval query, interval operations**

I. Introduction

CONTINUITY is one of the most distinctive features of video data that makes it different from other kinds of data. A video data is a seamless continuous sequence of frames. By this nature of video data, any arbitrary subsequences of these frames may be meaningful units for indexing as well as for querying. However, there are practical difficulties in identifying all such units, which generally causes indexing to be limited to a certain number of units only. Traditional frameworks for query processing thus cannot be applied to video databases. A traditional database storing non-continuous data, in general, is a set of discrete data. An answer to a query on such a database is a subset of the data that satisfies the query conditions. A video database, on the other hand, is a set of video intervals, i.e. a set of continuous sequences. An answer to a query on a video database, in general, is a set of only certain portions of those continuous sequences.

Generally, there are two different approaches to parsing raw video data into smaller units for indexing. Indexing schemes which rely largely on automatic parsing decomposes the raw video data into automatically detected shots [4], [29]. On the other hand, works by [8], [18], [27], which are based on manual annotation, choose arbitrary video intervals defined by so-called semantic boundaries as indexing units. In either of these approaches, the problem that end-users often face is that the intervals they are hoping to find may not have been defined as answer units in the database. Automatically detected shots may not be appropriate units for answering a query. On the other hand, defining all semantic boundaries and having all meaningful units ready in the database manually can be hardly achieved. It is because there is no common consensus on what semantic boundaries are. As a result, what is meaningful unit to an annotator may not be meaningful to the user who issues the query. Therefore, manual indexing is labor intensive and yet may never be complete. All these things suggest that there is bound to be discrepancies in granularity between the indexed units and the intervals to be retrieved.

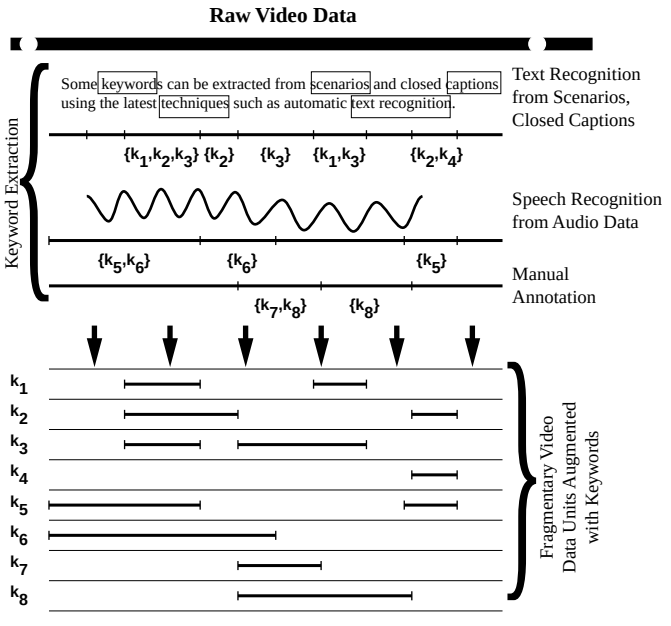

Fig. 1. Various techniques for keyword extraction

In order to solve this basic problem, which is highly prevalent in video databases, we propose a query framework where query results are dynamically synthesized from the video units defined in the database. In our framework,

Sujeet Pradhan is with the Department of Computer Science and Mathematics, Kurashiki University of Science and the Arts, 2640 Nishinoura, Tsurajima-cho, Kurashiki, 712-8505 Japan. Phone: +81 86 440–1089, fax: +81 86 440–1062, e-mail: sujeet@soft.kusa.ac.jp This work was performed while the author was with the Graduate School of Science and Technology, Kobe University, Japan.

Keishi Tajima is with the Department of Computer Science and Engineering, Kobe University, 1-1 Rokkodai-cho, Nada-ku, Kobe 657-8501, Japan. Phone: +81 78 803–6234, fax: + 78 803–6234, e-mail: tajima@db.cs.kobe-u.ac.jp

Katsumi Tanaka is with the Graduate School of Science and Technology, Kobe University, 1-1 Rokkodai-cho, Nada-ku, Kobe 657-8501, Japan. Phone: +81 78 803–6226, fax: +81 78 803–6390, e-mail: tanaka@db.cs.kobe-u.ac.jp

we assume that keyword information are associated with fragmentary video units in the database. It should be noted here that such information can be not only added manually but also can be extracted automatically (see Fig. 1). For example, speech recognizers can automatically transcribe video soundtracks [6]. The current state-of-art also allows one to automatically recognize text captions. Some video data even have scenarios or textual description added during the production time. Natural language processing technique can automatically transform all such textual information into a set of descriptive keywords [26].

Our framework allows users to formulate queries declaratively by listing a keywords with some quantifiers. The queries thus specified are transformed into algebraic expressions which will include the query keywords and the respective interval operators. Each query keyword is matched with the keywords associated with the indexed video units to produce intermediate result sets. Final answer intervals are then composed by performing interval operations on the sets of matched video units. Since such a set of answers may contain plenty of intervals, answers will be ranked in the order of relative relevance to the query.

The main goal of our research is to design a query mechanism which will enable us to dynamically compute answer intervals that exist in the raw video data, even though such intervals are not indexed as answer units in the database. In order to achieve this goal, we

1. define new interval operations required to compute answers,
2. provide a framework for query formulation in which ambiguity in query semantics can be avoided with the help of quantified keywords, and
3. present methodologies to rank answers.

The rest of the paper is organized as follows: In Section II, we present some motivating examples behind our work. Section III presents comparisons of our work with other related work. Some basic definitions of indexed video units and video database are presented in Section IV. In Section V, we explain our query mechanism in detail. First, we formally define the syntax of our declarative query. We then define various interval operations, and show how our declarative queries are transformed into expressions with those interval operations to compute the query result. In Section VI, we discuss a couple of methods to calculate the degree of relevance so that answers can be ranked and presented to users in order. Query approximation is presented in Section VII. Finally, in Section VIII we summarize the main contributions of this paper and present some prospects for our further research.

II. Motivation

In the past, query models for video data have been developed with particular annotation models [8], [9], [18], [27] in consideration. The usual approach is to assign attributes or descriptions to arbitrary video units and then define several interval operations such as *interval union*, *interval intersection*, and *interval concatenation* in order to compute video intervals as query results. The operations proposed in those research works, however, cannot always produce appropriate intervals that users intend to find in the first place.

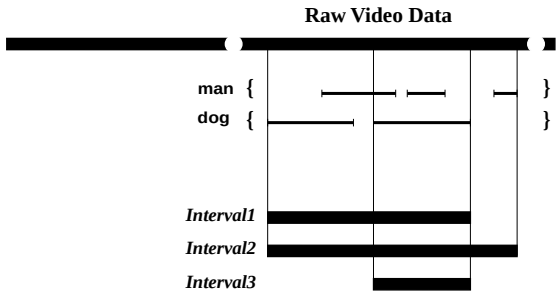

Fig. 2. What does a user expect?

For example, consider a query "*retrieve video intervals that show a dog running after a man*." In conventional approaches, a query result is computed by simply taking the *intersection* between those video units with an attribute value 'man' and those with an attribute value 'dog'. An actual scene of '*a dog running after a man*', however, usually involve lots of camera movements, zooming, and panning. It rarely happens that every frame in that scene contains both a 'dog' and a 'man' (see Fig. 2). Some of them may show only either of them, and there may be even frames showing none of them but the road between them. Therefore, the user who issues this query in the first place does not necessarily expects a video interval that shows 'dog' and 'man' in every frame throughout its play. Such an informally phrased query thus can be interpreted in a number of ways:

- Does it mean that the user wishes to find contiguous intervals consisting of frames showing either a dog or a man? (See Fig. 2 — Interval 1)
- Or does it mean that the user wants to find intervals which start with a frame in which either a man or a dog first appears and end with the final frame in which either one of them emerges? (See Fig. 2 — Interval 2)
- Or does it mean that the user is focusing on the dog, and his intention may be intervals with a dog in every frame and a man in some frames? (See Fig. 2 — Interval 3)

Producing appropriate intervals from the actual video data for even what seems to be such a simple query asks for new interval operations. Our goal is to develop such a set of operations that can compute answers to queries on a video database. For example, in order to generate an answer such as Interval 2 (See Fig. 2), we define a novel operation called *extended union*. It should be noted here that the usual *union* operation can produce the Interval 1 but not the Interval 2.

Specifying a query using those operations on video intervals, however, is not what an end-user would like to do. We therefore define a framework where end-users can formulate their queries by simply specifying a list of keywords with some quantifiers. This list of keywords with quantifiers declaratively specifies the condition for the query re-

sult. For example, the Interval 3 can be obtained by specifying the query as $((\texttt{dog})^{\forall} \wedge (\texttt{man})^{\exists})$. Our query mechanism translates the keywords into appropriate query expressions composed of our interval operations.

One problem in applying such a query framework is the existence of *noise* or the frames that cannot be matched with a query. For example, suppose we want to find a contiguous interval consisting of frames showing either a man or a dog. In the actual video, however, as we mentioned above, there may be a short sequence of frames showing only the road between them. If we strictly interpret the semantics of the query, an interval that includes those frames inside it, is not considered contiguous, and therefore will not be included in the query result. In practical applications of video databases, however, such a strict interpretation of the query semantics makes little sense.

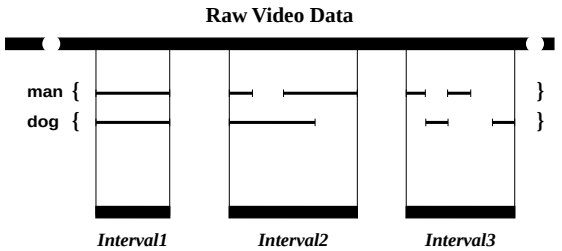

Fig. 3. Three unequal video interval answers

Consider the video data shown in Fig. 3 where arbitrary video units are indexed by the keywords '`man`' and '`dog`'. Suppose there are actually three different scenes of the dog running after the man. Then for a query "*retrieve video intervals that show either a dog or a man*", if we follow the strict semantics of the query, then it is clear that only the first two video intervals can be taken as the definite answer to this query. If, however, we relax the semantics of the query, then many intervals can be considered as possible answers. Intuitively, one can say that not all of these answers will be equally important to the user. Considering the three most relevant intervals as shown in the Fig. 3, one can intuitively say that the first video interval is preferable to the second and the second is preferable to the third one. However, such reasoning without any theoretical background is not sufficient to estimate the query response. We need to develop these intuitive notions into a consistent, useful methodology for query formulation, query processing, estimating relevance among the answers, and ranking output for presentation to a user.

Another problem that normally arises is from the discrepancy between the user who submits the query and the annotations that are actually stored in the database. For this discrepancy, it is often the case in video databases that there does not exist any answer intervals that exactly match with a given query formula even though there may be intervals that conceptually fulfill the users' purpose. We thus propose a query evaluation method based on approximation. When a query is issued and the result would be an empty set, the condition given in the query is relaxed and reformulated.

III. Related Work

There is now growing interest in querying the large resources of digital video data. Allen's work on temporal intervals [1] laid the foundation for many researches concerned with time intervals [14], [15]. He showed that there are 13 distinct temporal relationships that can exist between two arbitrary time intervals. Video data is also temporal although not in the traditional sense [7]. Some researches on video databases have thus been greatly influenced by Allen's temporal model. In [14], temporal-interval-based models have been presented for time-dependent multimedia data such as video. Our work, however, has mainly focussed on the operations for synthesizing new intervals but not on the relationship between given intervals. Therefore, our work and those work on interval relationships are orthogonal. As we saw in the example above, in any meaningful video interval, the same keyword may emerge for an indefinite number of times in no particular order. Some further investigation is required in order to integrate the condition specifications based on their temporal relationships and the interval operations into one query language. This issue will be considered in our future work.

Over the past few years, researchers around the world have developed systems which are capable of providing database support to video data [7], [9], [18], [27]. We review some of them briefly below. It should be noted here that none of the systems described below have defined operations that can synthesize possible answer intervals to a query from the existing video units.

A. OVID

Object Video Database (OVID) [18] is an instance-based video database system. In OVID, an arbitrary set of contiguous intervals can be defined as a meaningful entity, which they call a video object. Inheritable and non-inheritable attributes can be assigned to those video objects. Inheritable attributes are shared between objects on the basis of interval-inclusion relationship between them. The basic idea is to let video objects share inheritable attributes with their parent video object and relieve some burden of the manual annotation process.

Our objective is different from that of OVID. In OVID, a special emphasis has been put on annotation model rather than on query model. It also assumes that all meaningful interval units are predefined manually and stored in the database. However, as stated above, what is meaningful to a person who annotates the video data may not necessarily be meaningful to the one who issues the query. Furthermore, the operations defined in OVID cannot compute contiguous intervals if two input intervals are neither overlapping nor adjacent.

B. Algebraic Video

Algebraic video data model [27] is based on stratification approach [23]. Unlike simple stratification, however, in the

algebraic video model, they can define the hierarchical relationship between descriptions that are associated with the same video data. Parent nodes in the hierarchy represent the context of their child nodes. By this hierarchy, they can attach multiple views with different context to the same video data.

Their work also focuses on an annotation model rather than on a query model. Annotation should be done very carefully in order to ensure that answers are available for any kind of query. It also puts an extra burden on annotators to define relationships between descriptions. Defining all possible interval answers in advance in the indexing scheme, however, is not our approach in the first place.

C. VideoSTAR

VideoSTAR [8] is another video database project developed at Norwegian Institute of Technology. A generic model to support both the structural indexing and contents indexing has been proposed. Various interval-based and set-based operations have been defined as well. Three different kinds of descriptions — basic, primary and secondary — based on the context in which they appear can be associated with a video data. Video documents can be composed from various sources of video data and only those descriptions that are contextually related will be inherited by the newly composed documents. Their work is also focussed more on building a perfect annotation model for indexing which makes the system even more complex. As far as the query processing is concerned, no operations have been defined to compute contiguous intervals from two non-contiguous and non-overlapping intervals.

D. VIQS

Video Query System of VIQS [9] also defines a SQL-type of query language. It includes operations for determining whether two intervals of video overlap each other. Other operations include union, complement and intersection operations. Set iteration operations such as FORALL and FOREACH can also be performed. Users can provide parameters to choose whether to see a single frame, a frame of certain length or the complete sequence of frames. The answer presentation module is provided to merge different video-segments that are retrieved as answers to a query. This work also lacks the algebraic operations required for our purpose.

E. Informedia

Informedia project [26] uses techniques such as image processing, speech recognition, text recognition and language understanding to automatically parse visual, audio and textual information contained in video data. Video paragraphs are the units that are returned as answers to any query. Key frames are dynamically generated on the basis of query terms for answer presentation. Full text information retrieval (IR) system based on well-known technique of *tf/idf* (term frequency/inverse document frequency) has been used for keyword-based queries.

In an another IR-based approach, [22] has proposed an interactive retrieval process of video and still images using term weighting, relevance feedback and result ranking. Most of the studies that have been conducted in the field of information retrieval are concerned with textual documents. Unlike textual documents, however, there is no common consensus on basic indexing units for video documents in general. This is the main reason why IR models cannot be directly adapted to video data. In video databases, keywords can be associated with only fragmentary video units, and that makes it necessary to dynamically synthesize video intervals from those fragmentary intervals to answer queries.

IV. Video Indexing

There are two general approaches to associate descriptive information about the contents of video data — segmentation approach and stratification approach. The segmentation approach [13], [22], [24], [29] physically divides the raw video data into shots, scenes etc. Annotation models based on this approach are focussed specifically on the structural information of video data. This approach is straightforward and query processing is rather easy. There, however, have been many criticisms of this approach for its inflexibility and inability to segment the contextual information [18], [23], [27] contained in video data.

The stratification approach, on the other hand, is based on a layered annotation representation model which segments contextual information of the video [23]. The basic idea is not to segment video data but to segment descriptions. Each description is called a stratum that refers to a sequence of video frames. The strata may overlap or totally encompass each other. There already exist many annotation models based on this approach [18], [19], [27] .

Various information such as closed caption, the result of speech recognition, and manual annotation can be combined and used as the source of annotation information as illustrated in Fig. 1. No matter whether such information are extracted automatically or added manually, we believe, they can be associated with only fragmentary video units. However, these units may not be the ones what end-users are hoping to retrieve. As a result, meaningful intervals, which are not necessarily defined in the database, need to be computed in demand with the help of available indexed units. This will allow us to compute interval answers, that may possibly exist in the raw video data, even though such answer units are not indexed in the database.

Before we define and discuss our query mechanism, here we define the model of video annotations and video databases that we use as the base of our development. A video database, in general, is a collection of raw video streams. For the sake of simplicity, here we consider a single raw video stream.

Definition 1: A video stream F is a nonempty finite set of frames $f_1, f_2, \ldots, f_n$, which is a totally ordered set with respect to $<$ as $f_1 < f_2 < \ldots < f_n$.

Definition 2: If $f_s, f_e \in F$ and $f_s < f_e$, then a video interval $I[f_s, f_e]$ over F is the set of frames $\{f_k \in F \mid f_s \leq$

$f_k \leq f_e\}$.
f_s and f_e of $I[f_s, f_e]$ are the starting frame and the ending frame, and they are denoted by $start(I)$ and $end(I)$ respectively. A video interval is thus a stream of contiguous frames and is uniquely defined by its start frame f_s and end frame f_e. An interval is denoted by $I[f_s, f_e]$ or simply by I wherever $[f_s, f_e]$ can be omitted. $I(F)$ denotes the set of all intervals over F.

DEFINITION 3: An interval $I_1 \in I(F)$ is said to be identical to another interval $I_2 \in I(F)$, iff $start(I_1) = start(I_2) \wedge end(I_1) = end(I_2)$.
We denote $I_1 = I_2$ to denote I_1 is identical to I_2. We also denote $I_1 \neq I_2$ to denote I_1 is not identical to I_2.

DEFINITION 4: An interval $I' \in I(F)$ is said to be sub-interval of another interval $I \in I(F)$, if $(start(I') \geq start(I) \wedge end(I') \leq end(I)) \wedge (I' \neq I)$.
We denote $I' \subset I$ to denote I' is a sub-interval of I.

Now a *video description database* $\mathcal{VD}$ for a video F is defined as a set of video descriptions for F and has the following form:

$$\mathcal{VD} = \{(k_1, I_1), (k_2, I_2), \ldots, (k_n, I_n)\}$$

where each (k_i, I_i) is a unit *video description* for a video F. Each k in a unit description is a single keyword and I is an element of $I(F)$. Intuitively, here we denote (k, I) to suggest that the information described by the keyword k emerges everywhere *throughout* the video interval $I[f_s, f_e]$, that is:

$$(k, I) \text{ means } \forall f \in I[f_s, f_e].(k \text{ emerges in } f)$$

V. QUERY MECHANISM

Querying a video database is different from querying a traditional database. The semantics of query mechanism in a traditional database are obvious. Query conditions are defined by using primitive predicates and boolean operations (AND, OR, NOT), and the answer of the query is a set of data items satisfying the given condition. Even when some constructors are used for creating new data structure, they are simply evaluated for all combinations of those selected data items. It does not add any complexity to the phase of data item selection. The answer of the query is thus always uniquely defined, and no ranking is necessary because the data items in the answer are equally important. However, this is not the case when we are dealing with video database query processing. One particular reason is that there may not exist any predefined units of video intervals that can be directly returned as answers to a user. As a result, the answers to a query need to be synthesized dynamically using the fragmentary video units annotated with the pieces of information. In some cases, however, no intervals can be claimed as the best answer to a query. In those cases, the best we can do is to return a set of intervals which are likely to satisfy the users' need. In such a query result, all the intervals may not be equally relevant since some intervals may satisfy the query condition more closely than others. A ranking strategy thus must be applied in order to present the answers to a user in the best possible manner.

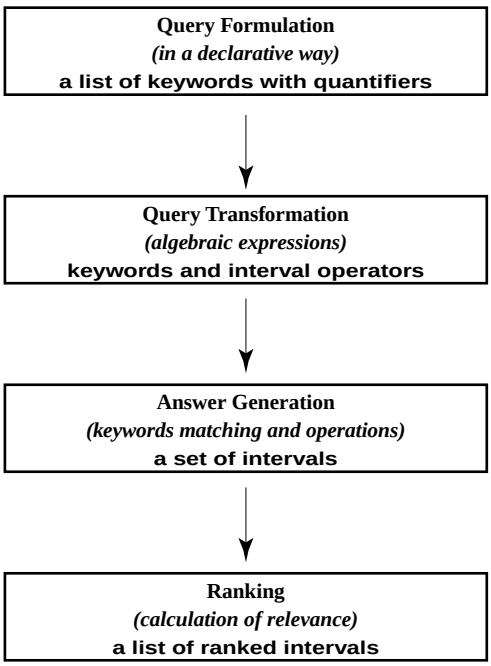

Fig. 4. A framework for our query mechanism

Our query mechanism can be summarized as shown in the Fig. 4. Users formulate queries by listing a keywords with quantifiers in a declarative way. The queries thus specified are transformed into algebraic expressions. These expressions will include the original keywords and the respective interval operators which we are going to define later in this paper. The keywords specified in the query are matched with the keywords associated with the indexed video units. Interval operations are performed on the matched video units in order to generate answer intervals. The answers thus computed are finally ranked for presentation to the users.

A. Query Formulation

We assume that users are interested in retrieving contiguous video intervals having some contextual meaning rather than bits and pieces of frames, or concatenation of short intervals collected from different parts of the video. We also assume that the most likely form of users request is "*find me contiguous video intervals in which a cyborg is fighting with a monster*". This kind of user's request can be represented in the following query form:

$$\texttt{Cyborg} \wedge \texttt{Fight} \wedge \texttt{Monster}$$

However, a user's intention, as mentioned in the Section II, is ambiguous in this form of query. The simplest interpretation may be the *intersection* of intervals with those keywords. In an actual video database, however, there may exist no video units in which all of the three keywords appear together at all. Even if there does exist such units, they may have lost the context of the actual scene since such units are usually fragmentary. Therefore, the more likely interpretation of the above query will be "*video intervals*

in which each keyword `cyborg`*,* `fight`*,* `monster` *emerges somewhere at least once*", or "*contiguous video intervals in which at least one of these keywords emerge in every frame.*

In order to allow users to specify their intention clearly and declaratively in queries, we introduce new operators and define a new query syntax as follows:

$$q ::= (k_1 \bullet k_2 \bullet \ldots \bullet k_n)^{\exists} \mid (k_1 + k_2 + \ldots + k_n)^{\forall} \mid q \wedge q \mid q \vee q$$

$\wedge$ and $\vee$ are the usual logical conjunction and disjunction. The meanings of the other two constructs are as follows:

1. $(k_1 \bullet k_2 \bullet \ldots \bullet k_n)^{\exists}$ — this means that all keywords k_i must emerge simultaneously somewhere in the answer interval. That is, this query specification requires that an answer interval $I[f_s, f_e]$ should satisfy the following condition:

$$\exists f \in I[f_s, f_e].\forall i \in \{1, \ldots, n\}.(k_i \text{ emerges in } f)$$

2. $(k_1 + k_2 + \ldots + k_n)^{\forall}$ — this means that at least one of $k_1, \ldots, k_n$ must emerge everywhere, i.e. in every frame, in the answer interval. That is, this query specification requires that an answer interval $I[f_s, f_e]$ should satisfy the following condition:

$$\forall f \in I[f_s, f_e].\exists i \in \{1, \ldots, n\}.(k_i \text{ emerges in } f)$$

Readers may think that $(k_1 \bullet \ldots \bullet k_n)^{\forall}$ and $(k_1 + \ldots + k_n)^{\exists}$ are also necessary. However, they are not needed because the following equivalence holds:

$$\begin{aligned} (k_1 \bullet \ldots \bullet k_n)^{\forall} &\equiv (k_1)^{\forall} \wedge \ldots \wedge (k_n)^{\forall} \\ (k_1 + \ldots + k_n)^{\exists} &\equiv (k_1)^{\exists} \vee \ldots \vee (k_n)^{\exists} \end{aligned}$$

The equivalence above is obvious because they correspond to the following theorem in the predicate logic:

$$\begin{aligned} \forall x.P(x) \wedge Q(x) &\equiv (\forall x.P(x)) \wedge (\forall x.Q(x)) \\ \exists x.P(x) \vee Q(x) &\equiv (\exists x.P(x)) \vee (\exists x.Q(x)) \end{aligned}$$

Readers may also notice that both $\bullet$ and $\wedge$ mean conjunction, and both $+$ and $\vee$ disjunction. In this paper, however, we use different symbols depending upon whether they are used inside or outside of the quantifiers $^{\exists}$ and $^{\forall}$. The reason is, as we explain later, they need different processing during the answer computation.

As a result, the general form of the query composed of those constructs can be expressed as below:

$$\begin{aligned} & (k \bullet \ldots \bullet k)^{\exists} \wedge \ldots \wedge (k \bullet \ldots \bullet k)^{\exists} \wedge \\ & \qquad (k + \ldots + k)^{\forall} \wedge \ldots \wedge (k + \ldots + k)^{\forall} \\ \vee \quad & (k \bullet \ldots \bullet k)^{\exists} \wedge \ldots \wedge (k \bullet \ldots \bullet k)^{\exists} \wedge \\ & \qquad (k + \ldots + k)^{\forall} \wedge \ldots \wedge (k + \ldots + k)^{\forall} \\ & \qquad \vdots \\ \vee \quad & (k \bullet \ldots \bullet k)^{\exists} \wedge \ldots \wedge (k \bullet \ldots \bullet k)^{\exists} \wedge \\ & \qquad (k + \ldots + k)^{\forall} \wedge \ldots \wedge (k + \ldots + k)^{\forall} \end{aligned}$$

As in usual expressions with $\wedge$ and $\vee$, we consider that $\wedge$ has priority to $\vee$ during the evaluation.

Let us consider some example queries:

1. $(\texttt{Cyborg})^{\exists} \wedge (\texttt{Fight})^{\exists} \wedge (\texttt{Monster})^{\exists}$
2. $(\texttt{Cyborg} \bullet \texttt{Monster})^{\exists} \wedge (\texttt{Mountain})^{\forall}$
3. $(\texttt{Cyborg} + \texttt{Monster})^{\forall} \wedge (\texttt{Cyborg} \bullet \texttt{Shoot})^{\exists}$
4. $(\texttt{Cyborg})^{\exists} \wedge (\texttt{Fight})^{\exists} \wedge (\texttt{Mountain})^{\forall}$
5. $(\texttt{Cyborg})^{\forall} \wedge (\texttt{Monster})^{\forall} \vee (\texttt{Fight})^{\forall}$

Intuitively, these queries can be interpreted as follows respectively:

1. *"find me video intervals in which each* `cyborg` *and* `monster` *appears somewhere and* `fight`*ing is taking place somewhere".*
2. *"find me video intervals in which the* `mountain` *appears throughout the scene and somewhere in the scene* `cyborg` *and* `monster` *appear together".*
3. *"find me video intervals in which either the* `cyborg` *or the* `monster` *appear everywhere and somewhere in the scene* `cyborg` *is seen* `shoot`*ing".*
4. *"find me video intervals in which the* `mountain`*s appear everywhere and the* `cyborg` *and the* `monster` *appear somewhere but not necessarily together".*
5. *"find me video intervals in which both* `cyborg` *and* `monster` *appear throughout the scene or* `fight`*ing has taken place throughout the scene".*

B. Interval Operations

To formally define the operational semantics of the queries specified by a user, we first need to define video operations on intervals and sets of intervals. These operations defined below will eventually be used to compute the interval answers from video units indexed in the database.

As stated before, a video interval is represented by a set of frames. Therefore, the set-theoretic operations can be applied to video intervals. Previous works on video databases have also defined variants of such set operations [7], [18]. However, some operations have been modified and some additional operations that will be relevant to our query mechanism will be introduced as well.

For the sake of clarifying the motivation behind our work, we have been using the term *video units* to refer to the stored intervals and *answer intervals* to refer to the computed answers to a video query. Hereafter, unless such distinction is necessary, we will simply use the term *interval* to refer to any video data whether they are the stored video units, the intermediate results yielded by interval operations or the resulting final answers to a query.

B.1 Interval Set Union

Given two sets of video intervals X and Y such that $X \subseteq I(F)$ and $Y \subseteq I(F)$, the operation *interval union* ($\uplus$) on these sets yields a single set of video interval S as follows:

$$\begin{aligned} X \uplus Y = \{I[f_s, f_e] \mid & (\forall f \in I.\exists I' \in X \cup Y.f \in I') \wedge \\ & (\forall I' \in X \cup Y.f_{s-1} \notin I' \wedge f_{e+1} \notin I')\} \end{aligned}$$

This operation takes input from two sets of intervals, which may contain overlapping and several adjacent intervals, and produces a single set of non-overlapping contiguous intervals. Adjacent intervals are concatenated to produce a single contiguous interval. Thus supposing there are two

sets of video intervals $X = \{I_1[10, 20], I_2[30, 40]\}$ and $Y = \{I_3[15, 35], I_4[45, 55]\}$, then $X \uplus Y = \{I_5[10, 40], I_6[45, 55]\}$.

B.2 Extended Union

Given two video intervals I_1 and I_2 over $I(F)$, the operation *extended union* ($\oplus$) on these two video intervals yields a single video interval I as follows:

$$\begin{aligned} I_1 \oplus I_2 = I[f_s, f_e] \text{ where} \\ f_s &= \min(start(I_1), start(I_2)) \text{ and} \\ f_e &= \max(end(I_1), end(I_2)). \end{aligned}$$

This operation takes two intervals as input and produce a single contiguous interval. The resulting interval will be contiguous no matter whether the input intervals are *adjacent*, *overlapping*, or *nonoverlapping and nonadjacent*. We emphasize here that the various interval operations defined in [7], [9], [18], [27] are unable to do that. Thus supposing there are two video intervals $I_1[10, 20]$ and $I_2[30, 40]$ then $I_1 \oplus I_2 = I[10, 40]$.

B.3 Set Extended Union

This is the set variant of the extended union operation. Given two sets of video intervals X and Y, the operation *set extended union* ($\bigoplus$) returns a set of video intervals yielded by the pairwise extended union ($\oplus$) between the elements of the two input sets.

$$X \bigoplus Y = \{x \oplus y \mid x \in X, y \in Y\}$$

Fig. 5 shows an example of the operation. Supposing we have two sets of intervals $X = \{x_1, x_2, x_3\}$ and $Y = \{y_1, y_2\}$ with three and two elements respectively as shown in the top of the figure, $X \oplus Y$ yields a set of six intervals $I_1, \ldots, I_6$ as shown at the bottom of the figure.

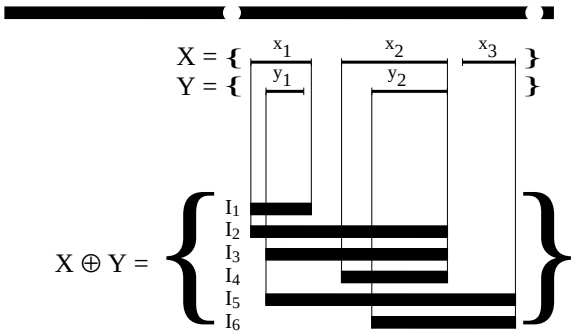

Fig. 5. Set Extended Union

B.4 Powerset Extended Union

Given two sets of video intervals X and Y, the operation *powerset extended union* ($\bigotimes$) returns a set of video intervals. These intervals are yielded by applying extended union ($\oplus$) to an arbitrary number (but not 0) of elements in X and Y. Formally, it is defined as follows:

$$\begin{aligned} X \bigotimes Y = \{\bigoplus(X' \cup Y') \mid \\ X' \subseteq X, Y' \subseteq Y, X' \neq \emptyset, Y' \neq \emptyset\} \end{aligned}$$

where $\bigoplus\{i_1, i_2, \ldots, i_n\} = i_1 \oplus \ldots \oplus i_n$.

This definition can be expanded as:

$$\begin{aligned} X \bigotimes Y \;=\; & (X \bigoplus Y) \cup \\ & (X \bigoplus X \bigoplus Y) \cup \\ & (X \bigoplus Y \bigoplus Y) \cup \\ & (X \bigoplus X \bigoplus X \bigoplus Y) \cup \\ & (X \bigoplus X \bigoplus Y \bigoplus Y) \cup \\ & (X \bigoplus Y \bigoplus Y \bigoplus Y) \cup \\ & \vdots \end{aligned}$$

Suppose we have two sets of intervals $X = \{x_1, x_2, x_3\}$ and $Y = \{y_1, y_2\}$ with three and two elements respectively as shown at the top of the Fig. 6. Intervals $I_2, \ldots, I_7$, shown at the bottom of the figure, are yielded by applying extended union to different pairs of intervals — each pair consisting of one element from X and one from Y. The same operation yields interval I_1 when performed on an interval set consisting of the two rightmost elements x_2 and x_3 from X and one element y_2 from Y. Interval I_8 is yielded when performed on another interval set that consists of at least two extreme elements x_1 and x_3 from X and either one or both of the elements from Y. We may also consider other combinations but the result will be one of the eight intervals $I_1, \ldots, I_8$.

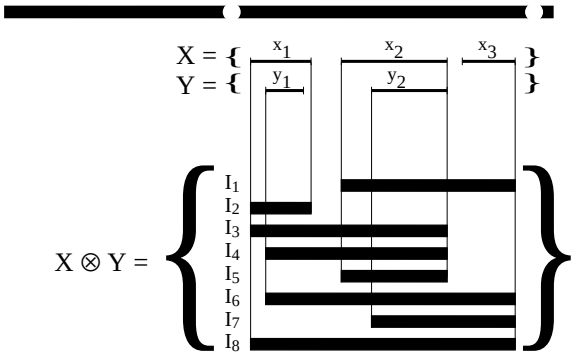

Fig. 6. Powerset Extended Union of X and Y

Although the powerset extended union seems to be a complex operation, its excellent property allows us to establish a simpler and more efficient definition. A closer study reveals that the same operation can be computed by the formula below.

$$X \bigotimes Y = (X \bigoplus X) \bigoplus (Y \bigoplus Y)$$

The transformation of the powerset extended union into three pairwise extended union operations is possible because both extended union and pairwise extended union hold certain algebraic properties. Readers are suggested to refer to our more recent publication for the details of these properties [21]. Here we give only an intuitive explanation. Since extended union operation between two intervals returns the minimal interval that contains both

of the input intervals, any other intervals encompassed by the two input intervals will also be contained in the resulting interval. As a result, when an interval set performs set extended union operation on itself such as $X \bigoplus X$, the resulting set will have already contained all the intervals that would have been yielded by performing the same operation infinite number of times on the same interval set. Formally, $X \bigoplus X = X \bigoplus X \bigoplus X = X \bigoplus X \bigoplus \ldots \bigoplus X$. This also means $X \bigoplus X$ will yield all the intervals that extended union can yield when applied on one or more elements in X. We can also prove that both $(X \bigoplus X \bigoplus Y)$ and $(X \bigoplus Y \bigoplus Y)$ are subsets of $(X \bigoplus X \bigoplus Y \bigoplus Y)$. Therefore, the operation $(X \bigoplus X) \bigoplus (Y \bigoplus Y)$ produces the same set of intervals as would be produced by $X \bigotimes Y$.

In Fig.7, the operations $(X \bigoplus X)$ and $(Y \bigoplus Y)$ produce two sets consisting six and three intervals respectively. Further set extended union operation on these two sets yields eight intervals $I_1, \ldots, I_8$ which is the same set of results that is obtained using its original definition (See Fig.6).

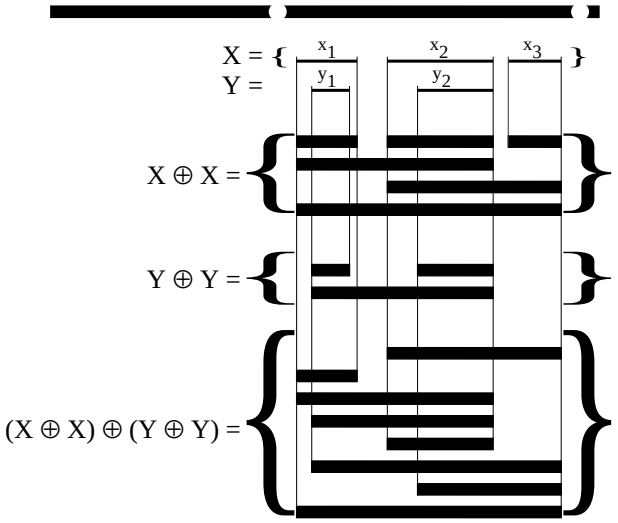

Fig. 7. Property of Powerset Extended Union

B.5 Interval Intersection

Given two video intervals I_1 and I_2 over $I(F)$, the two video intervals I_1 and I_2 are said to be intersecting iff $\exists f.f \in I_1 \wedge f \in I_2$, that is, $\max(start(I_1), start(I_2)) \leq \min(end(I_1), end(I_2))$. The operation *interval intersection* $(\odot)$ on any two arbitrary video intervals yields a single video interval I as follows:

$$I_1 \odot I_2 = \begin{cases} I[f_s, f_e] & \text{if } I_1 \text{ and } I_2 \text{ are intersecting} \\ \text{undefined} & \textit{otherwise} \end{cases}$$

where $f_s = \max(start(I_1), start(I_2))$ and $f_e = \min(end(I_1), end(I_2))$. Thus supposing there are two video intervals $I_1[10, 20]$ and $I_2[15, 40]$ then $I_1 \odot I_2 = I[15, 20]$. However, for $I_1[10, 20]$ and $I_2[25, 40]$, $I_1 \odot I_2$ does not produce any interval.

B.6 Interval Set Intersection

This is the set-variant of interval intersection operation. The *interval set intersection* $(\bigodot)$ operation on two sets of video intervals X and Y returns a set of video intervals constituting of the pairwise interval intersection $(\odot)$ between the elements of the two input sets.

$$X \bigodot Y = \{x \odot y \mid x \in X, y \in Y, x \text{ and } y \text{ intersect}\}$$

B.7 Intersect Test

The *intersect test* operation $(\ominus)$ on two sets of video intervals X and Y returns a subset of X including only intervals in X that intersect with at least one interval in Y.

$$X \ominus Y = \{x \in X \mid \{x\} \bigodot Y \neq \emptyset\}$$

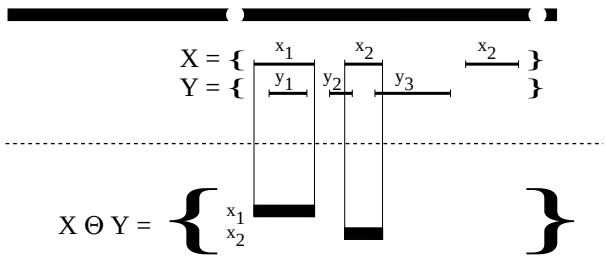

Fig. 8. Intersect Test between X and Y

For example, in Fig. 8, both X and Y has three elements $\{x_1, x_2, x_3\}$ and $\{y_1, y_2, y_3\}$ respectively. However, $X \ominus Y$ will include only two intervals x_1 and x_2, because x_3 does not intersect with any of the intervals y_1, y_2 or y_3 in Y.

C. Query Semantics

Having defined a set of video operations, now we formally define the operational semantics of the queries. Intuitively (but not strictly), the semantics of a query is defined on the basis of the following policies. The answer to a query includes those intervals such that:

1. they satisfy the condition specified in the query, and
2. they are as short as possible, that is, no other subintervals contain the same number of indexed video units associated with the keywords given in the query.

Next we define the semantics of queries step by step starting from a simple query, and finally showing the semantics of queries in general form.

C.1 Single Term Queries

Single term queries are those that contain only a single term with a single quantifier. As explained before, there are two kinds of simple queries.

1. $(k + \ldots + k)^{\forall}$
2. $(k\bullet \ldots \bullet k)^{\exists}$

The answer to simple queries are defined as follows. In the following definitions, $[\![q]\!]$ denotes the answer to a query q.

$$\begin{aligned} [\![(k)^{\forall}]\!] &= \{I | \exists (k, I) \in \mathcal{VD}\} \\ [\![(k_1 + \ldots + k_n)^{\forall}]\!] &= [\![k_1]\!] \uplus \ldots \uplus [\![k_n]\!] \end{aligned}$$

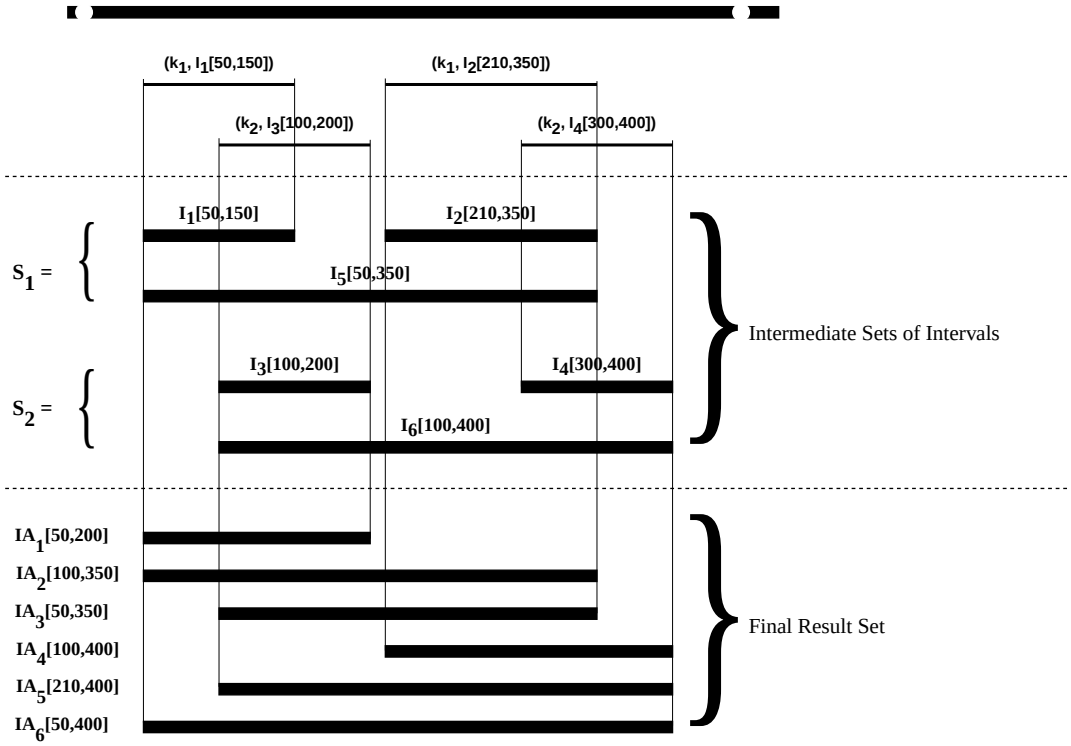

Fig. 9. An example of a conjunctive query

$$\begin{aligned}\llbracket(k)^{\exists}\rrbracket &= \{I|\exists(k,I)\in\mathcal{VD}\}\\ \llbracket(k_1\bullet\ldots\bullet k_n)^{\exists}\rrbracket &= \llbracket k_1\rrbracket\bigodot\ldots\bigodot\llbracket k_n\rrbracket\end{aligned}$$

In the definition above, we simply write $\llbracket k\rrbracket$ instead of $\llbracket(k)^{\forall}\rrbracket$ or $\llbracket(k)^{\exists}\rrbracket$ whenever no distinction is necessary between them, i.e. $\llbracket(k)^{\forall}\rrbracket = \llbracket(k)^{\exists}\rrbracket$.

C.2 Conjunctive Queries

Conjunctive queries are the ones which contain multiple terms listed with conjunctions. First we define the operational semantics of conjunctive queries including only either existential quantifiers or universal quantifiers.

$$\begin{aligned}\llbracket(k_1^1+\ldots+k_1^{m_1})^{\forall}\wedge\ldots\\ \wedge(k_n^1+\ldots+k_n^{m_n})^{\forall}\rrbracket &= \llbracket(k_1^1+\ldots+k_1^{m_1})^{\forall}\rrbracket\bigodot\ldots\\ &\quad\bigodot\llbracket(k_n^1+\ldots+k_n^{m_n})^{\forall}\rrbracket\\ \llbracket(k_1^1\bullet\ldots\bullet k_1^{m_1})^{\exists}\wedge\ldots\\ \wedge(k_n^1\bullet\ldots\bullet k_n^{m_n})^{\exists}\rrbracket &= \llbracket(k_1^1\bullet\ldots\bullet k_1^{m_1})^{\exists}\rrbracket\bigotimes\ldots\\ &\quad\bigotimes\llbracket(k_n^1\bullet\ldots\bullet k_n^{m_n})^{\exists}\rrbracket\end{aligned}$$

The following two examples will clarify the semantics of operations performed on video intervals to synthesize answers.

Example 1: Let us take an illustrative example. Suppose we have the following information stored in the database.

$$\begin{aligned}&(k_1, I_1[50,150])\\ &(k_1, I_2[210,350])\\ &(k_2, I_3[100,200])\\ &(k_2, I_4[300,400])\end{aligned}$$

Intervals $I_1,\ldots,I_4$ are shown at the top of Fig. 9. Let us consider the query "*find me video intervals in which both* k_1 *and* k_2 *appear somewhere*". The query can be formulated as:

$$Q = (k_1)^{\exists}\wedge(k_2)^{\exists}.$$

Then $\llbracket(k_1)^{\exists}\rrbracket$ and $\llbracket(k_2)^{\exists}\rrbracket$ can be given as follows:

$$\begin{aligned}\llbracket(k_1)^{\exists}\rrbracket &= \{I_1[50,150]), I_2[210,350]\}\\ \llbracket(k_2)^{\exists}\rrbracket &= \{I_3[100,200]), I_4[300,400]\}\end{aligned}$$

Next we need to perform powerset extended union ($\bigotimes$) between the two sets $\llbracket(k_1)^{\exists}\rrbracket$ and $\llbracket(k_2)^{\exists}\rrbracket$.

As explained before, it can be computed by using set extended union operations as below:

$$\begin{aligned}&\llbracket(k_1)^{\exists}\rrbracket\bigotimes\llbracket(k_2)^{\exists}\rrbracket =\\ &\quad(\llbracket(k_1)^{\exists}\rrbracket\bigoplus\llbracket(k_1)^{\exists}\rrbracket)\bigoplus(\llbracket(k_2)^{\exists}\rrbracket\bigoplus\llbracket(k_2)^{\exists}\rrbracket)\end{aligned}$$

The set extended union operation ($\bigoplus$) on each of these sets with itself will produce the following two sets of intervals.

$$\begin{aligned}S_1 &= \llbracket(k_1)^{\exists}\rrbracket\bigoplus\llbracket(k_1)^{\exists}\rrbracket\\ &= \{I_1[50,150], I_2[210,350], I_5[50,350]\}\\ S_2 &= \llbracket(k_2)^{\exists}\rrbracket\bigoplus\llbracket(k_2)^{\exists}\rrbracket\\ &= \{I_3[100,200], I_4[300,400], I_6[100,400]\}\end{aligned}$$

S_1 and S_2 are shown in the middle of Fig. 9. Performing the set extended union operation further on these two resulting sets will produce the following set of interval answers.

$$\begin{aligned}[\![Q]\!] &= S_1 \bigoplus S_2 \\ &= \{IA_1[50,200], IA_2[100,350], IA_3[50,350], \\ &\quad IA_4[100,400], IA_5[210,400], IA_6[50,400]\}\end{aligned}$$

These intervals are shown at the bottom of Fig. 9.

Example 2: Let us take a more realistic example. Suppose we have the following information stored in the database.

$$(\texttt{cyborg}, I_1)$$
$$(\texttt{cyborg}, I_2)$$
$$(\texttt{monster}, I_3)$$
$$(\texttt{shoot}, I_4)$$

$I_1, \ldots, I_4$ are intervals as shown in Fig. 10. Let us consider the query "*find me video intervals in which* `cyborg` *and* `monster` *together appear somewhere and somewhere in the scene* `cyborg` *is seen* `shoot`*ing*". The query can be formulated as:

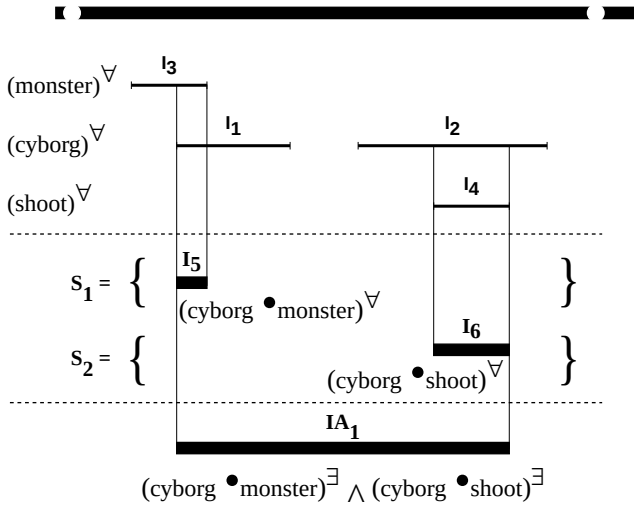

Fig. 10. Conjunctive query consisting of a pair of simultaneous query terms

$$Q = (\texttt{cyborg} \bullet \texttt{monster})^{\exists} \wedge (\texttt{cyborg} \bullet \texttt{shoot})^{\exists}.$$

The operations given below are obvious.

$$\begin{aligned}[\![(\texttt{cyborg} \bullet \texttt{monster})^{\exists}]\!] &= \{I_5\} \\ [\![(\texttt{cyborg} \bullet \texttt{shoot})^{\exists}]\!] &= \{I_6\}\end{aligned}$$

$$\begin{aligned}S_1 &= [\![(\texttt{cyborg} \bullet \texttt{monster})^{\exists}]\!] \bigoplus [\![(\texttt{cyborg} \bullet \texttt{monster})^{\exists}]\!] \\ &= \{I_5\} \\ S_2 &= [\![(\texttt{cyborg} \bullet \texttt{shoot})^{\exists}]\!] \bigoplus [\![(\texttt{cyborg} \bullet \texttt{shoot})^{\exists}]\!] \\ &= \{I_6\}\end{aligned}$$

where I_5 and I_6 are the intervals shown in the middle of Fig. 10. Finally, the following set of intervals are returned as the answer.

$$[\![Q]\!] = S_1 \bigoplus S_2 = \{IA_1\}$$

IA_1 is shown at the bottom of Fig. 10.

The semantics of general form of conjunctive queries including both existential quantifiers and universal quantifiers are defined as below:

$$[\![(q_1)^{\forall} \wedge \ldots \wedge (q_m)^{\forall} \wedge (q_{m+1})^{\exists} \wedge \ldots \wedge (q_n)^{\exists}]\!] = \begin{cases} [\![(q_1)^{\forall}]\!] \bigodot \ldots \bigodot [\![(q_m)^{\forall}]\!] \ominus \\ \quad [\![(q_{m+1})^{\exists}]\!] \ominus \ldots \ominus [\![(q_n)^{\exists}]\!] & \text{if } m \neq 0 \\ [\![(q_{m+1})^{\exists}]\!] \bigotimes \ldots \bigotimes [\![(q_n)^{\exists}]\!] & \text{if } m = 0 \end{cases}$$

Next we show an example of a complex query consisting of both kinds of quantifiers. Suppose intervals I_1 and I_2 associated with keyword k_1 and k_2 respectively, and intervals I_3 and I_4 associated with keyword k_3 as illustrated at the top of the Fig. 11. Then for a user wanting to find out an interval where there appears k_3 throughout the interval and each k_1 and k_2 appears somewhere in the interval, the query is formulated as below:

$$Q = (k_1)^{\exists} \wedge (k_2)^{\exists} \wedge (k_3)^{\forall}.$$

For this query, the computation defined above will return I_8 (shown at the bottom of the Fig. 11) as an answer but not I_7 because the operation $\ominus[\![k_1]\!]$ eliminates I_7.

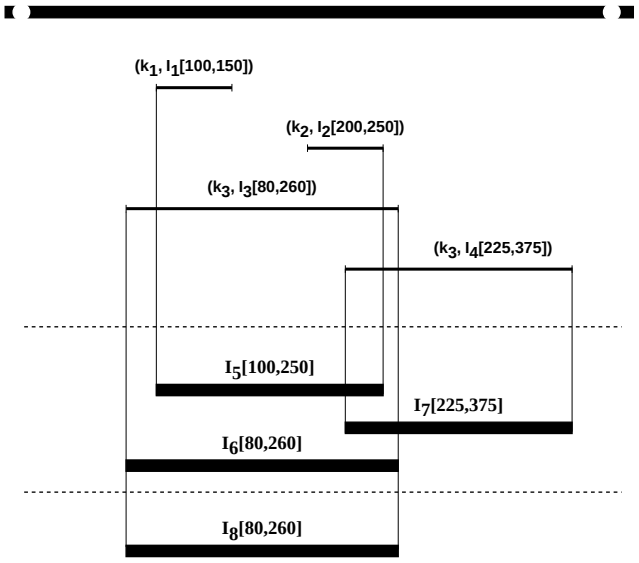

Fig. 11. A complex conjunctive query

C.3 Disjunctive Queries

Finally, the semantics of disjunctive queries is simply defined by the usual set union.

$$[\![q_1 \vee \ldots \vee q_n]\!] = [\![q_1]\!] \cup \ldots \cup [\![q_1]\!]$$

VI. Degree of Relevance and Ranking

As stated before, because of their relaxed semantics, certain types of queries when submitted against a video database, may produce answer intervals having unequal degree of relevance. It should be noted that a conjunctive query with existential quantifiers of type $(q_1)^{\exists} \wedge (q_2)^{\exists} \wedge \ldots \wedge (q_n)^{\exists}$, in particular, may yield an answer set that may contain a large number of elements. It is because of the way the answer intervals are computed by the powerset extended union operation applied for such queries. Not all of those elements may be equally relevant to a user who submits the query. It is important that the system be able to present the answers in the best possible manner to a user. The most common way to present a large number of unequal answers is to rank them in order.

Ranking strategy is important in any information retrieval systems [16], [22]. Its importance is even greater, especially in video databases in which query answer units are not predefined in the database but rather computed dynamically using the available information. There are many kinds of video resources such as TV News, movies, animations etc. and these resources are widely used in different video applications such as Video On Demand, News On Demand, education, video documentation, research-oriented programme etc. [3], [8]. Having such a wide variety of data and numerous kinds of applications, it is very difficult to make definite unique choices on how to compute relevance since relevance largely depends on the nature of the applications and the nature of the query itself.

However, we believe a relevant answer to any video data query, in general, should be an interval which intuitively,

1. should satisfy all the query conditions,
2. should be short enough in such a way that it does not contain too much unnecessarily lengthy scenes, and
3. should be long enough in such a way that it doesn't break out of the context.

We propose two different ways to compute the degree of relevance of a video interval against a given query.

A. Weighted Terms

Supposing there are n terms in the query $q = (k_1)^{\exists} \wedge (k_2)^{\exists} \wedge \ldots \wedge (k_n)^{\exists}$, and suppose an answer I has L number of frames. Then the answer I's degree of relevance to the query q is computed as $\sum l_i/L$, where l_i is the length of frames in which the term k_i emerge. This strategy is chosen under the assumption that the keywords relevant to a query statement are clustered within a certain range of video interval. It is also assumed that two contextually related keywords in a meaningful video interval are, in general, close to each other. The basic idea is that the more the video units in which the query terms appear to overlap, the more significance the resulting interval will have.

Consider an example with four intervals $I_1, \ldots, I_4$ shown at the top of the Fig. 12. Then supposing for a query $q = (k_1)^{\exists} \wedge (k_2)^{\exists}$, four answer intervals $IA_1, \ldots, IA_4$ are computed as shown at the bottom of the Fig. 12. The average length of frames, which is computed by the formula $\sum l_i/L$ stated above, in which the query terms are clustered determines the relevance of an answer. For example, for the answer interval IA_1, it is $((440-350)+(440-340))/(440-340) = 190/100 = 1.900$. The numbers shown at the bottom right side in Fig. 12 indicate each answer interval's respective degree of relevance.

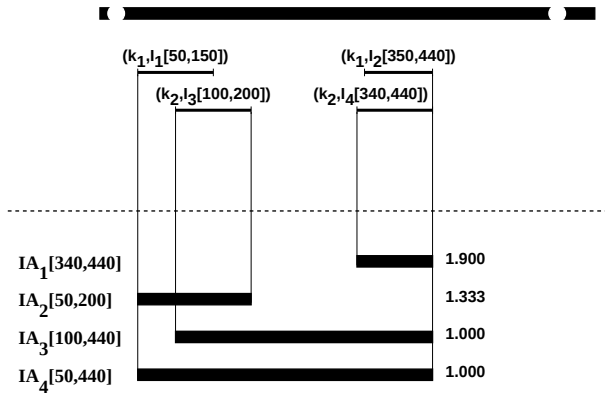

Fig. 12. Ranking based on weighted terms

B. Maximal Length of Allowable Noise

Given a query $q = (k_1)^{\exists} \wedge (k_2)^{\exists} \wedge \ldots \wedge (k_n)^{\exists}$, *noise* is said to exist in a video interval if and only if there exists a frame in which none of the term present in the query would emerge. Otherwise, the interval is called noise-free for the query q.

The ranking strategy described above cannot be justified when two keywords are separated too far. It is highly unlikely that a gap of an hour between two keywords can still preserve the context in which they appear. In such cases, whether a noise is allowable or not depends largely on the absolute length of that noise and not on the relative length.

In most TV dramas and movies, we often see that two persons appear for some time, then other things are shown for a very short moment, and those two persons appear again. In such cases, it can be assumed that the entire interval including the noise (the short interval of time when both of the persons do not appear at all) composes a single scene, because such short intervals do not break the contextual flow of the scene. On the other hand, if the length of such noise is five minutes in an interval with similar contents, then its possibility of being just a sequence of frames breaking two independent scenes is quite high. Therefore, in such cases, a different strategy must be taken into consideration.

The second method considers the maximum length of contiguous noise frames that may be irrelevant to a given query. The basic idea is the farther the keywords are separated, the less likely the interval's relevance for the given query will be. Consider again an example, with four intervals $I_1, \ldots, I_4$ shown at the top of the Fig. 13. Then supposing for a query $q = (k_1)^{\exists} \wedge (k_2)^{\exists}$, four answer intervals $IA_1, \ldots, IA_4$ are computed as shown at the bottom of the Fig. 13. In the figure, white rectangles indicate the noise

portions and the negative numbers shown at the right side are the measures of the respective maximum noise present in each answer interval.

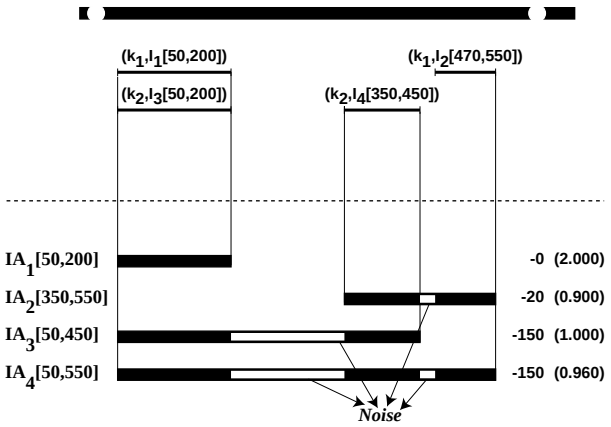

Fig. 13. Ranking based on the absolute length of *noise*

The answer interval IA_2 ranked higher than IA_3 because IA_2 includes shorter length of noise than IA_3 or IA_4 and therefore can be considered more likely to be an independent scene. It is important to note that when the same set of intervals are ranked according to the *weighted terms* strategy, IA_2 is ranked last, that is after IA_3 and IA_4. In the figure, this different ranking order is denoted inside the parenthesis shown at the rightmost side of each answer interval.

VII. Query Approximation

Video data may not have been described in the way a user may have expected. It is difficult to imagine that the expressions used by end-users will always conform with the annotations associated with video units [20]. As a result, sometimes no video intervals may be exactly matched with the users' specification. We provide a query evaluation based on approximation for dealing with empty answers in our query mechanism.

Approximative answers have been studied by many researchers in the past. Approximation is necessary usually when a database system has only partial information [12], [16] about the real world. In [11], several constructs have been defined for various kinds of partial information. In [2], it is argued that even when two systems have complete information about two separate worlds, the information that they possess may be insufficient to answer certain queries. They have shown how answers can be approximated for dealing with such queries. [25] proposes a query processor that makes approximate answers available when there is not enough time to produce an exact answer. As many of these researchers have pointed out there are some advantages to use approximation-based query evaluation. An approximate answer is quicker to evaluate and could be an intermediate answer so that user interaction could be allowed during the query processing. If there is no precise answer to a query, an approximate answer, even though it is not exact, will be returned.

Our query evaluation based on approximation can be summarized as follows. First we define two different kinds of orderings among video intervals. The first ordering is defined on the basis of how well two different annotations can describe the same video interval. The second ordering is defined between a video interval and its subintervals according to the goodness of the information contained in them. This will let us define ordering on a set of queries according to how specific or how general the conditions specified in the queries are. We then describe some strategies for re-formulating the original query in such a way that intervals that can be approximated to the original query are returned as answers.

A. Ordering based on Annotations

There are many different ways to interpret a video interval. All 'Bill Clinton', 'American President', 'Politician' can be keyword interpretations of a video interval showing 'Bill Clinton'. Those keywords, however, are not equal to each other. Some of them are more specific than others.

In order to define ordering among the annotations associated with a video interval, we first define a value generalization hierarchy which is a *partial order* where values are ordered on the basis of a *more-specific* ($\preceq$) relationship. We assume this relationship to be reflexive, transitive and anti-symmetric. An expression such as $k \preceq k'$ denotes that a value k' is *more-specific* than another value k. `politician` $\preceq$ `Bill Clinton` thus indicates that the value `Bill Clinton` is *more-specific* than the value `politician`.

Then we also define the relation $\preceq$ for interval annotation in a natural way as follows:

$$(k, I) \preceq (k', I) \text{ iff } k \preceq k'$$

[19] has presented a more ambitious annotation model in which descriptive keywords supplemented by quantifiers such as *somewhere*, *everywhere* and *simultaneous* can be provided to arbitrarily chosen video intervals. The detailed explanation is beyond the scope of this paper. For such an ideally described video data, the ordering on annotations can be extended in such a way that

$$((k)^{\exists}, I) \preceq ((k)^{\forall}, I)$$

and

$$((k_1)^{\exists} \wedge (k_2)^{\exists}, I) \preceq ((k_1 \bullet k_2)^{\exists}, I)$$

Furthermore, a set of annotations containing more descriptive keywords is more-informative than one containing a lesser number of keywords.

$$((k_1)^{\exists} \wedge (k_2)^{\exists}, I) \preceq ((k_1)^{\exists} \wedge (k_2)^{\exists} \wedge (k_3)^{\exists}, I)$$

B. Ordering based on Interval-Subinterval Relationship

Ordering can be also defined between a video interval and its subintervals according to the goodness of the information contained in them. For two intervals I and I' having a relationship of $I' \subset I$, interval I' is said to be a more-informative interval than I, if I' contains at least as much

information as I plus some extra information. Intuitively, if I and I' both contains a keyword k partially (that is *somewhere* within the interval) and I' is a subinterval of I, then I' is considered more-informative. This judgment is made on the basis that compared to I, I' has less number of frames where k does not appear. In other words, I' has relatively less *partial* information and hence provides us richer information about the whole data in general. Again we use $I \preceq I'$ to denote I' is more-informative than I.

More formally,

$$((k)^{\exists}, I) \preceq ((k)^{\exists}, I') \text{ for each } I' \subset I$$

and

$$((k_1 \bullet k_2)^{\exists}, I) \preceq ((k_1 \bullet k_2)^{\exists}, I') \text{ for each } I' \subset I.$$

However it should be noted that

$$((k)^{\forall}, I') \preceq ((k)^{\forall}, I) \text{ for each } I' \subset I.$$

C. Query Reformulation

Given a query Q', if no answer intervals are found, then the query can be reformulated into Q such that $Q \preceq Q'$. Here, the original query Q' is more specific than the reformulated query Q. It should be noted here that if $[\![(Q')]\!]$ and $[\![(Q)]\!]$ represents the set of answers for the query Q' and Q respectively, then $[\![(Q')]\!] \subset [\![(Q)]\!]$.

The following strategies explain the query reformulation in general.

C.1 Universal Approximation

If a query $(q)^{\forall}$ is issued but there is no interval that satisfies the condition specified by q throughout its play, then the query can be reformulated to $(q)^{\exists}$.

C.2 Simultaneity Approximation

If a query $(q_1 \bullet q_2)^{\exists}$ is issued but there is no interval in which q_1 and q_2 appear simultaneously, then the query can be reformulated to $(q_1)^{\exists} \wedge (q_2)^{\exists}$, i.e. a query retrieving intervals in which both q_1 and q_2 appear but at different places within the intervals.

C.3 Generalization Approximation

If a query $(k)^{\exists}$ is issued but there is no interval in which k emerges, then the query can be reformulated to $(k')^{\exists}$ with some keyword k' that is less-specific than k.

It should be noted that a single query statement Q may have many approximations. The query should be reformulated in such a way that the closest answers to the one that would have been found by the original query should be returned. Thus, if $Q_2 \preceq Q_1 \preceq Q$, then Q_1 should be the immediate choice as the reformulated query for Q. However, there may not be just one such closest candidate because there are several ways to make a query less specific as explained above. When there are multiple closest candidates, we need to select only one of them on the basis of some criteria. However, this issue will not be discussed further in this paper.

VIII. Conclusion and Future Works

In any video database system, end-users often have difficulty in retrieving the intervals that they desire to see. This is because the intervals they are hoping to find may not have been defined as answer units in the database. Whether indexing units are extracted automatically or identified manually, descriptive information about video data can be associated with only fragmentary video units. As a result, the problem will definitely arise when users issue queries to look for undefined intervals. In such cases, intervals need to be computed dynamically from the indexed units that currently exist in the database. We have presented a new video query model which attempts to find a set of possible interval answers from such a video database. The model has not been designed with one particular indexing scheme in mind.

In order to compute interval answers, we defined a new set of operations for queries that are most likely to be submitted on a video database. We also presented a framework in which users can specify their queries declaratively by simply listing keywords with quantifiers. The semantics of these query specifications are obvious, which make query interpretation easy. The answers to those declarative queries are computed by transforming those queries into algebraic query expressions and by evaluating them. These expressions contain specified keywords and respective operations. To present the answers in the best possible manner, we discussed some ranking strategies that are relevant to video databases. We also discussed how query approximation can be realized in case no answers to a query can be computed.

Algebraic operations such as union, intersection, concatenation does not always produce the desired answers since these operations do not consider the presence of noise in a meaningful unit, which is highly natural and common in video data. The main contribution of this paper is the definitions of interval extended union operation and its set variants. Given a query, these operations let us compute all the possible boundaries for interval answers from a set of stored video units. In other words, meaningful units can be computed from a source of fragmentary units which may or may not be meaningful in their original forms. It should be noted here that these operations can be extended to any continuous data whose semantic boundaries for meaningful units can be defined in numerous ways. As a matter of fact, the idea of synthesizing answer intervals from indexed video units can even be implemented in a hybrid system such as QBIC [5] [10].

There are many open issues that need additional investigation. The ranking strategies presented here may not be the best ones and therefore need further study. In [17], various methodologies for benchmarking multimedia databases are proposed. They have classified retrieval techniques into four categories according to the nature of data and query. Precise vs imprecise data and well-formulated vs ill-formulated queries are discussed. In our case, although the data in itself may be precise, it may be incomplete to answer even precise queries. Hence, there is a pos-

sibility of imprecise answers being computed and returned. A new methodology needs to be considered to benchmark the retrieval of continuous media such as video.

The extended union operation produces plenty of intervals most of which might be irrelevant to the user. In real applications, however, such answer overloading is inefficient and undesirable. As we discussed in Section VI also, there exists certain temporal constraints in a video interval to make it a single meaningful unit. In [28], time-constrained policy is used as an aid for clustering video shots. This notion can be applied to evaluate whether the similar keywords that emerge within a time interval form a single meaningful unit or not. In order to reduce answer overloading, other explicitly specified temporal constraints need to be considered as well. We have been working on these issues and so far the results have been promising.

Acknowledgements

This work is partly supported by the Japanese Ministry of Education under Grant-in-Aid for Scientific Research on Priority Area: "Advanced Databases", No. 08244103. This work is also partly supported by Research for the Future Program of Japan Society for the Promotion of Science under the Project "Researches on Advanced Multimedia Contents Processing". We would like to thank all anonymous referees for their invaluable comments which helped us improve the overall contents of our manuscript.

References

[1] J.F. Allen. "Maintaining Knowledge about Temporal Intervals". *Communications of the ACM*, 26(11):832–843, 1983.

[2] P. Buneman, S. Davidson, and A. Watters. "A Semantics for Complex Objects and Approximate Queries". *Journal of Computer and System Sciences*, 43(1):170–218, August 1991.

[3] A.K. Elmagarmid, H. Jiang, A.A. Helal, A. Joshi, and M. Ahmed. "Video Database Systems: Issues and Products and Applications". Kluwer Academic Publishers, 1997.

[4] C. Faloutsos and K. Lin. "FastMap: A Fast Algorithm for Indexing and Data-Mining and Visualization of Traditional and Multimedia Datasets". *SIGMOD Record*, 24(2):163–174, 1995.

[5] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. "Query by Image and Video Content: The QBIC System". *IEEE Computer*, 28(9):23–32, September 1995.

[6] A. Hauptmann and M.A. Smith. "Text and Speech and Vision for Video Segmentation: The Informedia Project". In *AAAI Fall 1995 Symposium on Computational Models for Integrating Language and Vision (http://informedia.cs.cmu.edu/)*, 1995.

[7] R. Hjelsvold, R. Midtstraum, and O. Sandst. "A Temporal Foundation of Video Databases". Recent Advances in Temporal Databases. Proceedings of the International Workshop on Temporal Databases and Zurich and Switzerland, 1995.

[8] R. Hjelsvold, R. Midtstraum, and O. Sandst. "Searching and Browsing a Shared Video Database". Multimedia Database Systems. Design and Implementation Strategies. chapter 4. Kluwer Academic Publishers, 1996.

[9] E.J. Hwang and V.S. Subrahmanian. "Querying Video Libraries". *Journal of Visual Communications and Image Representation*, 7(1):44–60, March 1996.

[10] W. Li, K. Selçuk Candan, K. Hirata, and Y. Hara. "A Hybrid Approach to Multimedia Database Systems through Integration of Semantics and Media-based Search". *Lecture Notes in Computer Science – Worldwide Computing and Its Applications*, 1274:182–197, August 1997.

[11] L. Libkin. "Aspects of Partial Information in Databases". PhD thesis, University of Pennsylvania, 1994.

[12] W. Lipski. "On Semantic Issues Connected with Incomplete Information Databases". *ACM Trans. Database Systems*, 4(3):262–296, September 1979.

[13] T.D.C. Little, G. Ahanger, R.J. Folz, J.F. Gibbon, F.W. Reeve, D.H. Schelleng, and D. Venkatesh. "A Digital On-Demand Video Service Supporting Content-Based Queries". In *Proc. of ACM Multimedia 93 and California and USA*, pages 427–436, 1993.

[14] T.D.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] N.A. Lorentzos and Y.G. Mitsopoulos. "SQL Extension for Intervals Data". *IEEE Transactions on Knowledge and Data Engineering*, 9(3):480–499, May/June 1997.

[16] J.M. Morrissey. "Imprecise Information and Uncertainty in Information Systems". *ACM Transactions on Information Systems*, 8(2):159–180, 1990.

[17] A.D. Narasimhalu, M.S. Kankanhalli, and J. Wu. "Benchmarking Multimedia Databases". *Multimedia Tools and Applications*, 4(3):333–356, May 1997.

[18] 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.

[19] S. Pradhan, K. Tajima, and K. Tanaka. "Using Prototype Objects and Powerdomains to Support Public Authoring of Video Databases". In *Proc. of Int'l Symposium on Cooperative Database Systems for Advanced Applications*, pages 210–217. World Scientific, 1996.

[20] S. Pradhan, K. Tajima, and K. Tanaka. "Querying Video Databases based on Description Substantiality and Approximations". In *Proc. of the IPSJ Int'l Symposium on Information Systems and Technologies for Network Society*, pages 183–190. World Scientific, 1997.

[21] S. Pradhan, K. Tajima, and K. Tanaka. "A New Algebraic Approach to Retrieve Meaningful Video Intervals from Fragmentarily Indexed Video Shots". In *Advances in Visual Information Management: Visual Database Systems 5 (VDB5)*, pages 11–30. Kluwer Academic Publisher, May 2000.

[22] B. Simonnot and M. Smail. "Model for Interactive Retrieval of Videos and Still Images". In *Multimedia Database Systems. Design and Implementation Strategies*, chapter 10. Kluwer Academic Publishers, 1996.

[23] T.G. Aguierre Smith and G. Davenport. "The Stratification System: A Design Environment for Random Access Video". In *Proc. 3rd Int'l Workshop on Network and Operating System Support for Digital Audio and Video*, pages 250–261, 1992.

[24] D. Swanberg, C. Shu, and R. Jain. "Architecture of Multimedia Information System for Content-based Retrieval". In *Audio Video Workshop and San Diego and CA and USA*, November and 1992.

[25] S.V. Vrbsky and J.W.S. Liu. "An Object-Oriented Query Processor that Produces Monotonically Improving Approximate Answers". In *Proceedings of the 7th IEEE Conf. on Data Engineering and Kobe and Japan*, pages 472–81, April 1991.

[26] H.D. Wactlar, T. Kanade, M.A. Smith, and S.M. Stevens. "Intelligent Access to Digital Video: Informedia Project". *IEEE Computer*, 29(5):46–52, May 1996.

[27] R. Weiss, A. Duda, and D. Gifford. "Composition and Search with a Video Algebra". *IEEE MultiMedia*, 2(1):12–25, Spring 1995.

[28] M. Yeung, B.L. Yeo, and B. Liu. "Extracting Story Units from Long Programs for Video Browsing and Navigation". In *International Conference on Multimedia Computing and Systems*, pages 296–305, June 1996.

[29] H.J. Zhang, C.Y. Low, S.W. Smoliar, and J.H. Wu. "Video Parsing and Retrieval and Browsing: An Integrated and Content-based Solution". In *Multimedia 95 Proceedings*, pages 15–24, November 1995.

Sujeet Pradhan received a BE in Mechanical Engineering from the University of Rajasthan, India in 1988, an MS in Instrumentation Engineering in 1995 and a Ph.D. in Intelligence Science in 1999 from Kobe University, Japan. Since 1999 May, he is a lecturer of the Department of Computer Science and Mathematics at Kurashiki University of Science and the Arts, Japan. He was a Colombo Plan scholar during 1984–1988 and a Mombusho scholar during 1995–1997. A JSPS (Japan Society for the Promotion of Science) Research Fellow during the period between 1997 and 1999, his research interests include video databases, multimedia authoring, prototype-based languages and semi-structured databases. Dr. Pradhan is a member of IEEE Computer Society and Information Processing Society of Japan.

Keishi Tajima received the B.Sc., M.Sc., and D.Sc. degrees in information science from the University of Tokyo, in 1991, 1993, and 1996 respectively. In 1996, he joined the Department of Computer and Systems Engineering at Kobe University, as a research associate. His research interests include data model for non-traditional database applications, database programming languages, and security in database systems.

Katsumi Tanaka received the B.S., M.S., and Ph.D degrees in information science from Kyoto University, in 1974, 1976, and 1981 respectively. In 1986, he joined the Department of Instrumentation Engineering at Kobe University, as an associate professor. Since 1994, he is a professor of the Department of Computer and Systems Engineering and since 1997, he is a professor of Division of Information and Media Sciences of Graduate School of Science and Technology at Kobe University. His research interests include object-oriented databases, video databases, historical database models, and hypermedia systems. Dr. Tanaka is a member of the ACM, IEEE Computer Society and the Information Processing Society of Japan.