1997

VideoText Database System

Haitao Jiang

Danilo Montesi

Ahmed K. Elmagarmid
Purdue University, ake@cs.purdue.edu

Report Number:
97-003

VIDEOTEXT DATABASE SYSTEMS

Haitao Jiang
Danilo Montesi
Ahmed K. Elmagarmid

CSD-TR 97-003
January 1997

VideoText Database Systems

HAITAO JIANG
Computer Science Department
Purdue University
West Lafayette, IN 47907 USA
jiang@cs.purdue.edu

DANILO MONTESI*
School of Information Systems
University of East Anglia
Norwich NR4 7TJ, UK
dm@sys.uea.ac.uk
Dipartimento di Scienze dell'Informazione
Universita' di Milano
Via Comelico 39/41
20135 Milano, Italy
montesi@dsi.unimi.it

AHMED K. ELMAGARMID
Computer Science Department
Purdue University
West Lafayette, IN 47907 USA
ake@cs.purdue.edu

December 15, 1996

Abstract

This paper introduces a new approach to realize video databases. It consists of a VideoText data model based on free text annotations associated with logical video segments and a corresponding query language. Traditional database techniques are inadequate for exploiting queries on unstructured data like video, supporting temporal queries, and ranking query results according to their relevance to the query. In this paper, we propose to use information retrieval techniques to provide these features and to extend the query language to accommodate interval queries that are useful due to the stream nature of video data. Algorithms are provided to show how the user queries are evaluated. Finally, a generic and modular video database architecture based on VideoText data model is described.

Keywords: video databases, data and query model, content-based retrieval, information retrieval, database architecture.

*This author has been partially supported by *NATO-CNR Senior Fellowship* visiting the Department of Computer Science, Purdue University.

1 Introduction

Video databases store a large number of video streams and provide content-based accesses to users. An increasingly important issue in creating video databases is the indexing and retrieval of video data. Currently, there are two main approaches to this problem which are based on image analysis and text annotations, respectively. The image analysis approach supports accesses based on the visual content of the video data; however, in many cases, semantic content-based accesses is more interesting and nature to users. The text annotation approach provides semantic content-based accesses since automatic semantic interpration of video data is not feasible given the state art of computer vision and machine intelligence.

A number of proposals use database techniques to annotate and query video [4, 11, 18, 25]; however, they do not exploit the feature of free text annotation and sophisticated Information Retrieval techniques (IR) in terms of temporal queries and ranking answers [8, 17, 24]. It has been shown that Boolean retrieval (mostly used in databases) may yield a rather poor retrieval quality in the video databases. On the other hand, systems based on term weights or probabilistic retrieval improve retrieval effectiveness by producing a ranked list of answers according to their relevance to the query rather than just a set of answers as in Boolean retrieval [24].

In this paper, we introduce a video data model called VideoText that can be used for semantic content-based query purposes. It is based on free text annotations rather than a fixed set of keywords to index video streams or video segments. The annotations for many video data resources are already available, e.g. the closed caption in TV broadcasts. The proposed approach supports incremental, dynamic, and multiple creation of annotations. It uses IR techniques to provide content-based queries to a video databases and rank the query results. The basic operators of IR are extended to consider interval queries that are characteristic of video databases. In this way we can find, for instance, overlapping video segments whose annotations contain specific terms. The resulting model allows the user to define different granularities for the answer. For instance the result of a query can be either whole video streams or logical video segments. In addition, queries can be recursively refined, i.e. a query can be given on the result of previous one by adding further conditions or changing the granularity of the answer. In the proposed data model, video and annotation are related but independent. This induces a modular architecture for the resulting VideoText database where the video storage system and the information retrieval system are two components of a video database. Existing information retrieval systems can be readily integrated (although to have the full query language some extensions are required). We do not consider, due the nature of the approach, queries by shape, color, and other visual information; these kind of queries have been extensively studied in the literarture [1, 12].

The paper is organized as follows. Section 2 compares other related works and Section 3 recalls the basic concepts of information retrieval systems. Section 4 describes the VideoText data

model and query language, and query evaluation algorithms are given in the Section 5. Some query examples are presented in Section 6 to illustrate the ideas presented in the previous sections. Section 7 describes the general system architecture implied by our data model. Finally, in Section 8, we draw some conclusions and sketch further research issues.

2 Related Work

Video data modeling deals with the problem of how to represent the video data to facilitate users' accesses. Two main categories of video data models are segmentation-based models and annotation-based models.

The basic idea of segmentation based models [10, 11, 22, 23, 28] relies heavily on the image processing techniques. For a given video stream, scene change detection algorithms [14] are usually used to parse and segment the video stream into a set of basic unit called *shots*. These shots are then matched or classified against a set of domain specific templates (patterns) to extract higher level semantics and structures contained in the data. A hierarchical representation of the video stream can be built. The main advantage of these models is that the indexing process can be fully automated. But they also have following disadvantages:

- lack of flexibility and scalability since video streams are pre-segmented by the scene change detection algorithms.
- similarity measure between two frame images is ill-defined and limited, making the template matching process an ill posed problem.
- lack of applicability for video streams that do not have well defined structure. For a video stream of a class lecture, there is no clear visual structure in terms of shots, segmentation using scene change detection algorithms is difficult.
- limited semantics can be derived from this matching process, and the templates are application specific.

Video annotations are often used to provide content-based access in the multimedia systems. For example, Little et al. [15] propose to use a video schema that has *movie*, *scene*, and *actor* relations with a fixed number of attributes in a VOD service system. The video features are manually extracted and used to instantiate the schema. So, the queries are only allowed on the attributes of relations (title, setting, name etc.), and query on the temporal ordering of the video segments are not allowed. Furthermore, their schema doesn't consider the temporal relationship between video annotations, and descriptions can not be assigned to overlapping video segments. In another paper by Little et al. [16], seven timing relationships of the multimedia data are discussed, which consist the basis of interval query operators presented in the our video query language in the

Section 4. However, their focus was on the time-dependent multimedia data layout and authoring, which is different from ours.

The basic idea of the annotation based models is to layer the content information on top of the video stream, rather than segment the video data into shots. Each annotation is associated with a logical video segment, i.e. a subset of a video stream which can be defined by the starting frame number and ending frame number. The annotation layering and the notion of logical video segment have certain advantages:

- variable level of granularities can be supported, i.e. annotations can be made on logical video segment of any length, from a single frame to the whole video stream.
- the annotation information can be easily handled by existing sophisticated information retrieval (IR) and database techniques.
- various annotations can be linked to same logical segment of video data, and they can be added and deleted independent of the underlying video streams. This supports dynamic and incremental creation and modification of the video annotation.
- supports semantic content-based video retrieval and queries.

One of the earliest annotation based model is the *stratification model* proposed by Davenport et al. [6, 21]. Several video data models [3, 9, 11, 25] based upon video annotation layering have since been developed.

The generic video data model developed by Hjelsvold et al. [11] allows free text annotation of video material, i.e. annotating of any arbitrary video frame sequence. This is done by establishing a *Annotates* relationship between a *FrameSequence* and an *Annotation*. The sharing and reuse of the video material is supported by the idea of logical *VideoStream*. However, only simple Boolean queries are supported on the annotations. The temporal relationship and management of the video annotations are not fully addressed.

Nested stratifications is allowed in the Algebraic Video model proposed by Weiss et al. [25], i.e. the logical video segments can be overlapped or nested. Multiple views on the same raw video segments can be assigned, and video algebraic operators are used for the recomposition of the video material. Four kinds of interval relations (*precede, follow, overlap, and equal*) are defined as attributes of a logical video segment. However, the following problems persist:

- video data is annotated by keywords, i.e. attribute/value pairs, thus quite limited.
- not all the interval relations between two logical video segments can be expressed using above four relations; for example, the *during* relation between two intervals [15].

- these interval relation attributes are only allowed one value for one logical video segment, but a given logical video segment can have the same interval relation with many other logical video segments.

- can not support user queries that are based on interval relations among logical video segments.

Although text annotation is allowed as an attribute, this model, like many previous works, is focused on the video editing, re-composing etc., and doesn't address the problem of how to manage video annotation using IR techniques. Thus, it only allows Boolean queries on certain attributes, i.e. free text queries are not allowed.

The main limitation of the annotation based models is the creation of the annotations. Video annotation can come from the following sources:

- closed captioning which can be captured from the video signal by using some specific product, e.g. TextGrabber (TM).

- text appears in the frame images. These frames can be detected by scene change detection algorithms [27, 14], and recognized by using OCR techniques.

- audio signals which can be transformed into text by using voice recognition techniques, as in the Informedia project of Carnegie Mellon University [20].

- manual annotation done by humans according to their knowledge and understanding of the video data.

Due to the limitations of current machine vision and image processing techniques, the full automation of the video annotation process will remain impossible for a long period of time. Thus, video annotation is usually a manual process which can be biased, limited and very time consuming. However, for large amounts of the educational video material and TV programs, annotation material can be automatically obtained from the first three sources mentioned above.

3 Information Retrieval

Information retrieval systems retrieve textual documents using a partial match between the user query and the documents. There are three fundamental issues in information retrieval: the choice of documents representation; query formulation; and the construction of a suitable ranking function which determines the extent to which a document is relevant to a query. Depending on how these problems are addressed, different categories of retrieval models have been developed, such as the Boolean, vector space, and probabilistic models [17, 24]. The Boolean retrieval considers only the presence of terms defined in the query. It yields a rather poor retrieval quality, either producing too many or too few documents and does not rank the output [17]. These drawbacks stem mainly

from the exact matching strategy and the naive document representation. A document is retrieved only if it logically satisfies a query. This approach is very close to that of database systems. The vector space model is able to rank documents by adopting an inexact matching strategy. That is, documents are ranked according to the values of a predefined similarity measure. These similarity values are assumed to reflect the degrees of relevance of individual documents with respect to a query [17]. The probabilistic model uses the knowledge of the distribution of the index terms for the probabilistic ranking. In addition, it can be seen as an adaptive model based on Bayesian decision theory. Instead of a query being formulated directly by the user, a ranking function representing the information request is constructed by an inductive learning process (i.e. relevance feedback). Several studies have been done to combine the strength of these three categories of retrieval models and to formulate a unified approach [5, 19].

For our purpose, we consider a generic information retrieval system providing ranked output and the ability to express logical and temporal queries. This system allows the user to define queries through variables, terms and a set of core operators like `AND`, `OR`, `NOT` and `ADJ` where the last operator stands for adjacency and used to impose an order between the left argument and the right argument (i.e. `information ADJ system`) means that `system` must follow `information`. The query `X ADJ system` returns any document where `system` follows `X` [17, 26]. For instance, consider a simple document database with three documents containing two terms with different relevance. The index can be conceptually defined as follow

$$\begin{aligned}&\texttt{indterm}(d_1, \texttt{video}, 0.7)\\ &\texttt{indterm}(d_1, \texttt{retrieval}, 0.8)\\ &\texttt{indterm}(d_2, \texttt{video}, 0.9)\\ &\texttt{indterm}(d_2, \texttt{retrieval}, 0.5)\\ &\texttt{indterm}(d_3, \texttt{retrieval}, 0.9)\end{aligned}$$

where the relevance of each term is computed according to the chosen model. For instance considering the frequency of a term in a document [17]. The query `video AND retrieval` will returns the following ranked answers:

$$\begin{aligned}&(d_1, 0.56)\\ &(d_2, 0.45)\end{aligned}$$

Note that the query evaluation process can be seen as a ranking function that takes a document database and produce an ordered document database defining the effect of a query.

4 VideoText Model

As we discussed in the Section 2, the current annotation based video data models have one or more of the following problems:

- Use simple keyword or Boolean matching which usually produces poor results.

- Do not support full text annotations; video annotation is only done on a fixed set of attributes.
- Do not support queries that relate to the interval relationships between video annotations.

In this section, we introduce a video data model called *VideoText* which attempts to address above problems. The query language based on the VideoText model is also presented, along with the database system architecture that can be used to implement the model.

4.1 Data Model

The basic idea of *VideoText* data model (VT) is quite simple and intuitive since it is designed in such a way that the video streams and video annotations are related but independent. The VideoText model can be defined as:

$$VT = (V, T, Map)$$

where V is a set of video streams $v^i \in V, i = 1, ..., n$, and is also a set of logical video segments; a *logical video segment* is a consecutive of frame sequence $[f^i_j, f^i_{j+1}, ..., f^i_k]$ that has a meaning by itself for indexing and query purposes. Thus, it can span from a single frame to the whole video, and they can also overlap each other. T is a set of video annotations which are text documents that describe the contents of the logical video segments. Map is a mapping relation between logical video segments and video annotations.

The first and last frame numbers are used to give a compact representation of a logical video segment. That is, $[f^i_j, f^i_{j+1}, ..., f^i_k]$ can also be denoted as $[v^i_{j-k}]$. If it is a single frame, then the beginning and ending frames are the same. For example, the single frame $[f^i_j]$ is denoted with $[v^i_{j-j}]$. i is the video identifier that uniquely identifies the raw video data stream $v^i \in V$ that a logical video segment belongs to.

A *video annotation* $t^i_l \in T$ is the content description provided for a logical video segments $[v^i_{j-k}] \in V$. The number of annotations can be less or more than the actual number of frames in a video stream, since the annotation is done on the logical video segments. Each annotation $t^i_l = text$, where *text* is a free text annotation. Note that the free text can be used to express other information, such as the unique identifier to denote the user or application responsible for the annotation.

As we discussed in Section 2, there are several sources of video annotation, and in this paper, we assume annotations are already available. A mapping relation Map defines the correspondence between annotations and logical video segments. For instance the mapping

$$Map([t^2_7], [v^2_{50-1436}])$$

define a one-to-one relationship between between $[t^2_7]$ and $[v^2_{50-1436}]$. We can also have a many-to-one mapping. For instance the mapping

$$Map([t_7^2, t_8^2], [v_{50-436}^2])$$

defines a many-to-one relationship between $[t_7^2, t_8^2]$ and $[v_{50-1436}^2]$. Similarly many-to-many and one-to-many relationships can be defined. For instance the following mappings

$$Map([t_1^2, t_7^2], [v_{1-35}^2, v_{15-75}^2])$$

$$Map([t_7^3], [v_{45-60}^3, v_{61-150}^3])$$

defines a many-to-many relationship between $[t_1^2, t_7^2]$ and $[v_{1-35}^2, v_{15-75}^2]$, and a one-to-many relationship between $[t_7^3]$ and $[v_{45-60}^3, v_{61-150}^3]$. The advantages of above mapping is that the same video data can be shared and annotated by different users for different purposes and can be easily reused in different applications.

Compared to segmentation based video data models, the VideoText model deals with logical video segments, i.e. the video stream itself is not physically segmented. Each annotation can be associated with one or more logical video segments defined by the starting and ending frame number and vice versa, which implicitly determines the temporal relationship between any given two annotations within the same video. This provides much more flexibility and eliminate the need of maintaining the temporal relations between video segments as in the video segmentation based models. The VideoText data model allows unlimited different annotations and interval relationship among them are embedded in the model itself. Thus, any user query which contains such relationship between two annotation (corresponding to logical video segments) is acceptable.

4.2 Query Language

We now turn our attention to the query language based on above VideoText data model. The emphasis is on the query language on free text video annotations, which users can use to retrieve logical video sequences in terms of their semantic content. Other types of queries that are based only on the video identifier, user identifier, starting frame number, and the ending frame number etc. are quite straightforward and will not be detailed here.

A VideoText database (VTDB) is defined as a collection of VideoText documents $\{vt^1, \ldots, vt^s\}$ where $vt^i \in VTDB, i = 1, ..., s$ is defined according to the above VideoText data model. Each VideoText document vt^i is a tuple:

$$(v_{j-k}^i, t_l^i)$$

where v_{j-k}^i and t_l^i are a logical video segment and one of its video annotations respectively. We also denote the set of all logical video segments in a VideoText database $VTDB$ as $VTDB.V$, and the set of all annotation as $VTDB.T$. The mapping relation between $VTDB.V$ and $VTDB.T$ is implicitly defined by the tuples in the $VTDB$.

A query Q on VideoText database can be defined as a triple:

$$(expr, scope, r)$$

where *expr* is an query expression formed through terms, and zero or more operators. The *scope* of a query defines the granularity of the answer; it can be either video streams (v) or logical segments (s). The scope is video streams means that the query target objects are video streams, and they are the units that are returned. If the scope is logical video segments, the query is done on logical video segments, and they are the units returned. Notice that a video streams is just a special case of its logical video segments. The reason we single it out is because interval operators are not defined among video streams, as we will see later on. The r denotes the maximum number of logical video segments that have to be retrieved; this restriction is due to the fact that the query on video data content often result multiple answers rather than an unique one.

The syntax of the query expression *expr* can be defined as

$$\begin{aligned}
expr &:= expr\ OP_i\ interval \\
&:= interval \\
interval &:= interval\ OP_b\ text \\
&:= \texttt{NOT}\ text \\
&:= string\ \texttt{ADJ}\ string \\
&:= text \\
text &:= string \\
&:= (\ expr\ ) \\
OP_i &:= \texttt{AFTER} \mid \texttt{BEFORE} \mid \texttt{OVERLAPS} \mid \texttt{OVERLAPS}^{-1} \mid \texttt{STARTS} \mid \texttt{ENDS} \\
&\quad \mid \texttt{MEETS} \mid \texttt{MEETS}^{-1} \mid \texttt{DURING} \mid \texttt{DURING}^{-1} \\
OP_b &:= \texttt{AND} \mid \texttt{OR} \\
string &:= [\texttt{A} - \texttt{Za} - \texttt{z0} - \texttt{9}]^*
\end{aligned}$$

The *string* is usually a character string (usually a word or a number) that the user want to find in an annotation. So, the simplest query can be just, for example, `(computer, s, 10)` qhich means to find at most 10 logical video segments whose annotation contains the word `computer`. The OP_b, OP_i, `ADJ` and `NOT` are the operators that defines the relationships between the terms or the relationships between the annotations that contains these terms. There are three classes of operators:

Boolean operators include `AND`, `OR` and `NOT` are the traditional Boolean operators. For instance the expression `information AND system` is looking for video data (either videos or

segments depending on the query scope) which has annotation that contains both `information` and `system`. If the scope is the video, the terms can be in different annotations of the logical video segments that belong to the same video stream; if instead, the scope is the logical video segment, both the terms must be in the same annotation of a logical video segment.

Temporal operator `ADJ` defines the ordering relationship between two terms. The expression like `information ADJ system` returns all the answers where `system` follows `information`. Thus VideoText documents containing the phrase `system information` are not returned. Again, if the scope is the video, the terms can be in different annotations of the logical video segments that belong to the same video stream; if instead, the scope is the logical video segment, both the terms must be in the same annotation of a logical video segment.

Interval operators deal with the interval relationships between logical video segments that belong to the same video stream, as well as their corresponding annotations because of the mapping between them. There are ten operators include `OVERLAPS`, `OVERLAPS`$^{-1}$, `STARTS`, `ENDS`, `AFTER`, `BEFORE`, `DURING`, `DURING`$^{-1}$, `MEETS`, `MEETS`$^{-1}$ which is based on the thirteen possible ways of relating any two given intervals discussed in [16]. These operators' scope is the logical video segment.

Operators `OVERLAPS` and `OVERLAPS`$^{-1}$ both return logical video segments that overlap each other, and whose corresponding annotations contain the given terms. They differ in the way of overlap. For instance, the expression `man OVERLAPS smoking` is looking for overlapping logical video segments whose annotations contain `man` and `smoking` respectively, and the logical video segement with `man` starts earlier. The operator `OVERLAPS`$^{-1}$ is the complementer of the operator `OVERLAPS`.

The operator `STARTS` returns logical video segments which start at the same time, and whose annotation contains the corresponding terms. For instance the expression `man STARTS smoking` is looking for logical segments annotated with `man`, which start at the same time as at least one logical video segment annotated with `smoking`. The `ENDS` operator is the complementer.

The operator `BEFORE` retrieves logical video segments which end just before some logical video segment starts, and their annotations must contain the given terms respectively. For example, the expression `man BEFORE smoking` is looking for logical video segments with `man` in their annotation, and which end before at least one logical video segment with the annotation term `smoking` starts. The operator `AFTER` is the complementer.

The operator `DURING` retrieves logical video segments which start after some logical video segment starts, and end before the given logical video segment ends, and their annotations must contain the given terms respectively. For example, the expression `man BEFORE smoking` is looking for logical video segments with `man` in their annotation, and each of them is completely contained

in at least one logical video segment with the annotation term `smoking`. The operator $\texttt{DURING}^{-1}$ is the complementer.

The operator `MEETS` retrieves logical video segments which ends as soon as some logical video segments starts, and their annotations must contain the given terms respectively. The operator $\texttt{MEETS}^{-1}$ is the complementer.

The above sets of operators allow users to define all the possible queries on the VideoText database. Notice that current IR systems are capable of dealing with Boolean and temporal operators, but not the interval operators. The query processing algorithms are discussed in Section 5.

The effect of a query Q is a function from a given set of VideoText documents $VTDB$ into a (ranked) subset ($VDBT'$) of the given VideoText documents ($VTDB$). Thus, the system allows users to recursively refine their queries, which at the same time, the query evaluation process itself is a recursive procedure.

$$Q : VTDB_n \rightarrow VTDB_{n+1}.$$

Consider a small VideoText databases $VTDB_0 = \{\texttt{vt}^1, ..., \texttt{vt}^{800}\}$. The query

$$Q_1 = (\texttt{information AND system}, \texttt{v}, 50)$$

returns a set of (ranked) videos (notice the scope of the query is the video stream) that are ranked according to the relevance of that video to the query as discussed in Section 3.

$$VTDB_1 = \{ \begin{array}{ll} 0.98 & \texttt{vt}^5 \\ 0.98 & \texttt{vt}^3 \\ \vdots & \vdots \\ 0.56 & \texttt{vt}^{34} \end{array} \}$$

Note that the relevance is computed once the query is provided and thus the input databases can be also seen as a set of equally relevant videos (all with relevance 1). At this point we can provide a query on the result of Q_1. For instance with the query $Q_2 = (\texttt{information ADJ system}, \texttt{v}, 20)$. Then we can have

$$VTDB_2 = \{ \begin{array}{ll} 0.98 & \texttt{vt}^5 \\ 0.16 & \texttt{vt}^{18} \\ \vdots & \vdots \\ 0.06 & \texttt{vt}^{78} \end{array} \}$$

Now we may want to find the logical segments of the video $\texttt{vt}^5$ using the query $Q_3 = (\texttt{information ADJ system}, \texttt{s}, 20)$. This correspond to have as input database $VTDB_2' = \{\texttt{vt}^5\}$. Thus the query returns

$$VTDB_3 = \{ \begin{array}{ll} 0.46 & [\texttt{v}^5_{15-500}] \\ 0.11 & [\texttt{v}^5_{3-3}] \end{array} \}$$

Note that both a whole video and a single video frame are special cases of VideoText document. The approach has the advantage to provide a framework where several queries can be give in cascade and the output of a query is the input of the next one. For instance, in the above query, by default the whole $VTDB_2$ is considered as the input of Q_3. This allows the user to refine a query, i.e. when we retrieve a set of videos or segments, they can be selectively played or another query can be give on the whole database or on the answer set. For example, the above query Q_2 can be given over the set of 50 videos that is the result of the previous query or over a single video if this is selected first and then the query is given. We will discuss more details of the query evaluation in the Section 5.

5 Query Evaluation

In this section, we present the query evaluation algorithms. A user query expression is evaluated by recursively decompositing it into sub-expressions and processing them based on the syntax given in Section 4.

Algorithm QUERYEVALUATION
Input: *A VideoText database VTDB and a query Q;*
Output:*A new VideoText database* $VTDB' \subset VTDB$;
begin
case Q
$(expr\ OP_i\ interval, scope, r)$:
V_l = QUERYEVALUATION$(VTDB, (expr, scope, r))$
V_r = QUERYEVALUATION$(VTDB, (interval, scope, r))$
$VTDB'$:= INTERVALEVALUATION(V_l, V_r, OP_i)
$(expr\ OP_b\ text, scope, r)$:
V_1 = QUERYEVALUATION$(VTDB, (expr, scope, r))$,
V_2 = QUERYEVALUATION$((VTDB, (expr, scope, r))$,
$VTDB'$ = BOOLEANEVALUATION(V_1, V_2, OP_b)
$(string$ ADJ $string, scope, r)$:
$VTDB'$ = IREVALUATION$(VTDB, (string$ ADJ $string, scope, r)$
(NOT $text, scope, r)$:
$VTDB' = VTDB -$ QUERYQUERY$(VTDB, (text, scope, r))$,
$(string, scope, r)$:
$VTDB'$ = IREVALUATION$(VTDB, (string, scope, r))$,
end.

The IREVALUATION algorithm first identifies those video annotations that contain the given term in the query expression by using IR techniques (TEXTQUERY), and then it looks for the

associated videos or logical video segments based on the mapping between logical video segments and annotations (VIDEORETRIVAL). Although simple, it indicates how the video database and the IR sub-system are integrated and interact with each other in our data model and query language.

Algorithm IREVALUATION
Input: *A VideoText database $VTDB$ and a query $Q = (string, scope, r)$;*
Output:*A new VideoText database $VTDB'$;*
begin
$VTDB'.T :=$ TEXTQUERY$(VTDB.T, Q)$;
$VTDB'.V :=$ VIDEORETRIVAL$(VTDB'.T)$;
$VTDB' := (VTDB'.V, VTDB'.T)$
end.

The algorithm TEXTQUERY is essentially the query processor of the information retrieval system for a given string. Please note that here we only let IR system deal with a string for the convenience of description. It would be more efficient to let IR system to handle the handle the query expression involving strings with only Boolean and temporal operators.

Algorithm TEXTQUERY
Input: *A set of annotations $VTDB.T$ and a query $Q = (string, scope, r)$;*
Output: *A new set of annotations $VTDB'.T \subseteq VTDB.T$;*
begin
$VTDB'.T := Q(VTDB.T)$
end.

The algorithm VIDEORETRIEVAL retrieves a set of videos or logical video segments that corresponds to a set of text annotations based on the mapping between two sets defined by VideoText documents.

Algorithm VIDEORETRIEVAL
Input: *A set of text annotations $VTDB'.T$ and the query Q;*
Output:*A set of video or logical video segments $VTDB'.V$;*
begin
$\forall t_l^i \in VTDB'.T$
if $(\exists v_{j-k}^i$ **AND** $(v_{j-k}^i, t_l^i) \in VTDB)$
if $Q.scope = s$
then $VTDB'.V = VTDB'.V \cup \{v_{j-k}^i\}$
else $VTDB'.V = VTDB'.V \cup \{v^i\}$
end.

The algorithm INTERVALQUERY is used to evaluate the interval relationship between two logical video segments from a *target* VideoText database and a *reference* VideoText database, respectively. The *target* VideoText database means that logical video segments of it are the possible result of the query, and logical video segments from the *reference* VideoText database are used as a reference interval with regard to the relationship given by the interval operators. For example, the query sub-expression `raining BEFORE sunshine` is to find a set of logical video segments (target) whose annotations all contain the term `raining`, and they must followed by at least one logical video segment (reference) whose annotation contains the term **sunshine**. These target logical video segments' annotations, however, are not required to contain the term **sunshine**. Obviously, only two logical video segments from the same video stream has valid interval relationships.

Algorithm INTERVALQUERY
Input: *A target VideoText database $VTDB_t$, a reference VideoText database $VTDB_r$, and an interval operator OP_i;*
Output: *A new target VideoText database $VTDB'$;*
begin
    case (OP_i)
        AFTER: AFTERQUERY($VTDB_t, VTDB_r$)
        BEFORE :BEFOREQUERY($VTDB_t, VTDB_r$)
        OVERLAPS :OVERLAPSQUERY($VTDB_t, VTDB_r$)
        OVERLAPS^{-1}:OVERLAPS^{-1}QUERY($VTDB_t, VTDB_r$)
        DURING :DURINGQUERY($VTDB_t, VTDB_r$)
        DURING^{-1} :DURING^{-1}QUERY($VTDB_t, VTDB_r$)
        STARTS :STARTSQUERY($VTDB_t, VTDB_r$)
        ENDS :ENDSQUERY($VTDB_t, VTDB_r$)
        MEETS :MEETSQUERY($VTDB_t, VTDB_r$)
        MEETS^{-1} :MEETS^{-1}QUERY($VTDB_t, VTDB_r$)
        end
end.

Following is the algorithm to implement the OVERLAPSQUERY. Interested readers can find other interval operator algorithms in the Appendix A.

Algorithm OVERLAPSQUERY
Input: *A target VideoText database $VTDB_t$ and a reference VideoText database $VTDB_r$;*
Output: *A new target VideoText database $VTDB'$;*
begin
    $VTDB' = \phi$

foreach $(v^i_{j-k}, t^i_l) \in VTDB_t$

begin

if $((\exists(v^i_{m-n}, t^i_o) \in VTDB_r)$ **AND** $(j < m < k < n))$

then $VTDB' = VTDB' \cup \{(v^i_{j-k}, t^i_l)\}$

end

end.

Algorithm BOOLEANEVALUATION

Input: *Two VideoText database:* $VTDB_1$ *and* $VTDB_2$*, and a Boolean operator* OP_b*;*

Output: *A new VideoText database* $VTDB'$*;*

begin

case (OP_b)

AND :$VTDB' = VTDB_1 \cap VTDB_2$

OR :$VTDB' = VTDB_1 \cup VTDB_2$

end.

6 Examples

In this section, two simple examples will be used to illustrate ideas in the previous sections. Queries, their step-by-step evaluation processes, and the corresponding results are given to help readers to understand the concept. Let's assume there is a video database which consists of three VideoText documents as shown in Figure 1. For the purpose of convenience, the annotation corresponding to the logical video segment v^i_{j-k} is denoted as t^i_{j-k} and the VideoText document (v^i_{j-k}, t^i_{j-k}) is simply represented by its logical video segment v^i_{j-k} in the following two examples.

6.1 Query Example One

In this example. we are going to show a query that contains Boolean, temporal as well as the interval operators which is expressed as following.

Q_1 = (((((George AND chair) AND (chair ADJ window)) OVERLAPS raining) BEFORE Chicago), s, 10)

Notice that in the above query, the granularity is the logical video segment because of interval operators in the query expression. The maximum number of query results is 10. Here is how the above query is processed step by step:

Step 1 the sub-expression (George AND chair) is looking for a logical video segment whose annotation contains both terms George and chair, the order between these two terms, however, is not important. The results are: v^1_{50-150} and $v^2_{115-235}$.

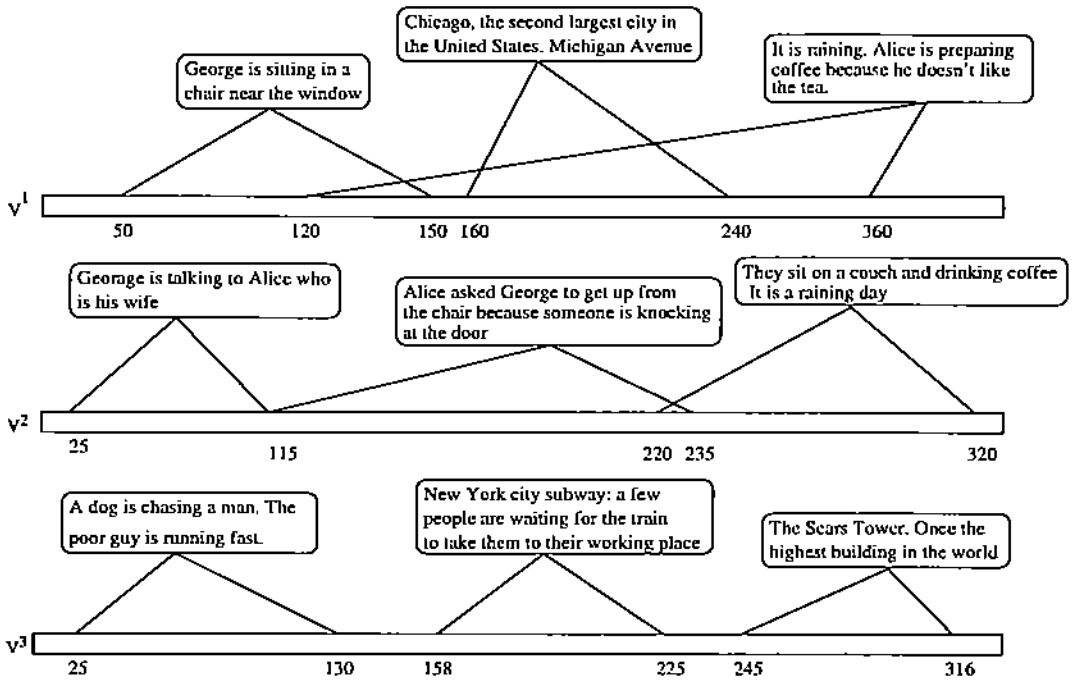

Figure 1: Annotations, mapping and videos

Step 2 the sub-expression (`chair ADJ window`) deals with the temporal ordering of the terms in the same annotation; the only logical video segment that has the term `window` after the term `chair` is v^1_{50-150}.

Step 3 the sub-expression ((`George AND chair`) `AND` (`chair ADJ window`)) applies `AND` Boolean operator to the results of Step 1 and Step 2 which returns the intersection of two results: v^1_{50-150}.

Step 4 the sub-expression (((`George AND chair`) `AND` (`chair ADJ window`) `OVERLAPS raining`) applies the interval operator `OVERLAPS` to the result of previous step. It results the logical video segment v^1_{50-150} (target)) since v^1_{50-150} is overlapping with the logical video segment $v^1_{120-360}$ (reference) which has the term `raining`.

Step 5 the logical video segment (target) v^1_{50-150} ends before the logical video segment $v^1_{160-240}$ (reference) which has the term `Chicago` in its annotation. The final result of the query is the logical video segment v^1_{50-150}, and the user can choose to play the video stream v^1 from frame 50 to 150 to view the result.

6.2 Query Example Two

As we discussed in 4, only Boolean and temporal operators can be used if the granularity of the queries is the video rather than the logical video segment. One example of such query can be as

following:

$$Q_2 = (((\texttt{George OR city}) \texttt{ AND chair}) \texttt{ AND } (\texttt{drinking ADJ coffee}), \texttt{v}, 3)$$

Since the granularity of the queries is the video, its annotation contains the annotations of all the logical video segments that belongs to it. The above query can be evaluated as follows:

Step 1 the sub-expression (`George OR city`) returns v^1, v^2 and v^3 since the annotation of all three video streams contains either `George` ($t^1_{50-150}, t^2_{25-115}, t^2_{115-235}$) or `city` ($t^1_{160-240}, t^3_{158-225}$).

Step 2 the sub-expression (`George OR city`) `AND chair`) eliminates v^3 from above intermediate result since its annotation doesn't have the term `chair`.

Step 3 now v^1 and v^2 need to be check against the sub-expression (`drinking ADJ coffee`) because of the `AND` Boolean operator. The video v^2 satisfies the condition since its annotation has both `drinking` and `coffee` terms, and term `coffee` is after `drinking` ($t^2_{220-320}$). The video v^1 is not selected although its annotation contains both terms, because the term `coffee` is not after the term `drinking` ($t^1_{120-360}$).

Step 4 the result of the query is the video stream v^2.

7 VideoText Video Database System Architecture

The VideoText model defined so far introduce a modular and general architecture for video databases. Although video annotations and logical segments are related in the data model, they can be managed by different components of the video database based on the VideoText model. These components of the VideoText database are:

- The *Video Storage System (VSS)* stores the video streams (using some video compression technique, like MPEG), and returns a set of videos or logical video segments in response to requests from the INT sub-system. The basic functionality of VSS is to store and retrieve the video data. It basically implements the VIDEORETRIEVAL function of the IREVALUATION algorithm in Section 5. The VSS can be just a file system (e. g. UNIX file system) or a continuous real-time media server with specific CPU and disk scheduling algorithms, depending on the requirements of the applications. The video streams are not even required to be reside on the same host or location; they can be distributed across the network, such as on the World Wide Web.

- The *Information Retrieval System* (IRS) stores indexes and video annotations, and allows the user to retrieve video annotations by specifying strings and their relationships (`AND`, `OR`, `NOT`, and `ADJ`) in the queries. The basic function it implements in the query processing is the

TEXTRETRIEVAL in the IREvaluation algorithm, which many existing IR systems, either commercial or free, can perform. One such example is the *Harvest* information discover and access system [2].

- The cooperation and integration of the two above sub-systems is made possible through a third component we called *Integrator (INT)* which provides following functionalities:
  - Acts as the bridge between the VSS and IRS sub-systems and controls their operations. For example, it sends query expressions with only strings and their relationships (`AND`, `OR`, `NOT`, and `ADJ`) to the IRS to obtain annotations that contain given terms. Similarly, it sends instructions to VSS to retrieve certain video streams or video segments.
  - Stores the mappings among logical video segments and their free text annotations. Thus it can find all the video annotations that correspond to a given logical video segment and vice versa.
  - Processes queries. As we discussed in Section 5, each user query is first parsed and decomposed, and sub-query involving strings and their relationships (Boolean and temporal) are sent to IRS sub-system and processed using IR techniques (IREVALUATION). The returned video annotation documents are used to retrieve the videos or logical video segments from VSS, and then interval operations in the query are performed by INT sub-system to obtain the final query results. It is interesting to note that if the granularity of the query is video stream, then the query processing is completely handled by the IRS. The INT sub-system only needs to retrieve the corresponding video streams from VSS and present them to the user as in Example 2 of Section 6.
  - User interface. INT is the sub-system that allows users to interact directly with the video database system. The interactions include query specification, video and annotation display, browsing and updating etc. Another interesting function that the INT can implement is to act as an information filter, i.e. it can filter out certain video segments (for example, violence) from the query results which the present user is not authorized to see.

Figure 2 shows the integration of above three sub-systems. One advantage of this system architecture is its flexibility, i.e. each sub-system is relatively independent of others. For example, one can always apply the state of the art of IR techniques to the IRS without too much changes in the VSS and INT sub-systems. Another advantage is that the video databases based on above architecture are much easier to be implemented since VSS and IRS sub-system can be based on existing technologies and systems, with only the INT sub-system requiring new implementation.

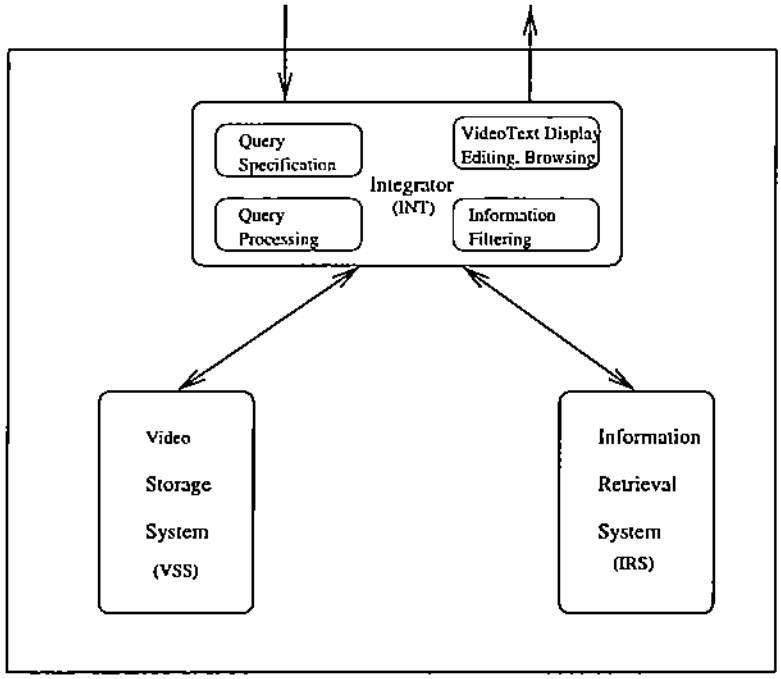

Figure 2: The General Architecture of a VideoText Database System

8 Conclusions and Future Work

We have presented a new approach to video databases based on a new video data model called VideoText. The model allows free text video annotation on the logical video segments, as well as querying based on the temporal and interval relationships between annotated logical video segments. The query processing is based on information retrieval techniques, and results can be ranked according to their relevance to the semantic content of the video data. The model also introduces a modular system architecture for implementing the video databases, where the video storage system (VSS) and the information retrieval system (IRS) can be integrated together by a third system (INT) to provide semantic content based queries and retrieval to the video data.

The VideoText model is rather general, and it can be further extended using structured text annotation and by adding some application specific elements when used with certain applications. For example, to implement a World Wide Web based video database system, the video annotations do not have to be plain text documents, they can be HTML (Hyper Text Markup Language) documents which have the mappings to the logical video segments embedded as hyperlinked URLs. We believe that this approach is also promising for multi-language indexing and query, and suitable as a component of video applications like Video-on-Demand services.

Some ideas presented in the paper have been used in the development of an animal behavior video database system [13], in which users can query and retrieve the video data based on the description of certain activities animals are engaged in. In the near future, we hope to further

explore the relationships between logical video segments, e.g. the inclusion inheritance. This has significant impact on the efficiency and consistency of the system. We also plan to implement a prototype system using VideoText model for a video database which consists of educational material for veterinary medicine.

A Implementation Algorithms for Interval Query Operators

Algorithm OVERLAPS^{-1}QUERY

Input: *A target VideoText database* $VTDB_t$ *and a reference VideoText database* $VTDB_r$;

Output: *A new target VideoText database* $VTDB'$;

begin

$VTDB' = \phi$

foreach $(v^i_{j-k}, t^i_l) \in VTDB_t$

begin

if $((\exists(v^i_{m-n}, t^i_o) \in VTDB_r)$ **AND** $(m < j < n < k))$

then $VTDB' = VTDB' \cup \{(v^i_{j-k}, t^i_l)\}$

end

end.

Algorithm STARTSQUERY

Input: *A target VideoText database* $VTDB_t$ *and a reference VideoText database* $VTDB_r$;

Output: *A new target VideoText database* $VTDB'$;

begin

$VTDB' = \phi$

foreach $(v^i_{j-k}, t^i_l) \in VTDB_t$

begin

if $((\exists(v^i_{m-n}, t^i_o) \in VTDB_r)$ **AND** $(j = m))$

then $VTDB' = VTDB' \cup \{(v^i_{j-k}, t^i_l)\}$

end

end.

Algorithm ENDSQUERY

Input: *A target VideoText database* $VTDB_t$ *and a reference VideoText database* $VTDB_r$;

Output: *A new target VideoText database* $VTDB'$;

begin

$VTDB' = \phi$

foreach $(v^i_{j-k}, t^i_l) \in VTDB_t$

begin

if $((\exists(v^i_{m-n}, t^i_o) \in VTDB_r)$ **AND** $(k = n))$

then $VTDB' = VTDB' \cup \{(v^i_{j-k}, t^i_l)\}$

end

end.

Algorithm AFTERQUERY

Input: *A target VideoText database* $VTDB_t$ *and a reference VideoText database* $VTDB_r$;

Output: *A new target VideoText database* $VTDB'.V$;

begin

$VTDB' = \phi$

foreach $(v^i_{j-k}, t^i_l) \in VTDB.V_t$

begin

if$((\exists(v^i_{m-n}, t^i_o) \in VTDB.V_r)$ **AND** $(n < j))$

then $VTDB'.V = VTDB'.V \cup \{(f^i_{j-k}, a^i_{j-k})\}$

end

end.

Algorithm BEFOREQUERY

Input: *A target VideoText database* $VTDB_t$ *and a reference VideoText database* $VTDB_r$;

Output: *A new target VideoText database* $VTDB'$;

begin

$VTDB' = \phi$

foreach $(v^i_{j-k}, t^i_l) \in VTDB_t$

begin

if $((\exists(v^i_{m-n}, t^i_o) \in VTDB_r)$ **AND** $(k < m))$

then $VTDB' = VTDB' \cup \{(v^i_{j-k}, t^i_l)\}$

end

end.

Algorithm DURINGQUERY

Input: *A target VideoText database* $VTDB_t$ *and a reference VideoText database* $VTDB_r$;

Output: *A new target VideoText database* $VTDB'$;

begin

foreach $(v^i_{j-k}, t^i_l) \in VTDB_t$

begin

if $((\exists(v^i_{m-n}, t^i_o) \in VTDB_r)$ **AND** $(m < j < k < n))$

then $VTDB' = VTDB' \cup \{(v^i_{j-k}, t^i_l)\}$

end

end.

Algorithm DURING^{-1}QUERY

Input: *A target VideoText database $VTDB_t$ and a reference VideoText database $VTDB_r$;*

Output: *A new target VideoText database $VTDB'$;*

begin

foreach $(v^i_{j-k}, t^i_l) \in VTDB_t$

begin

if $((\exists(v^i_{m-n}, t^i_o) \in VTDB_r)$ **AND** $(j < m < n < k))$

then $VTDB' = VTDB' \cup \{(v^i_{j-k}, t^i_l)\}$

end

end.

Algorithm MEETSQUERY

Input: *A target VideoText database $VTDB_t$ and a reference VideoText database $VTDB_r$;*

Output: *A new target VideoText database $VTDB'$;*

begin

foreach $(v^i_{j-k}, t^i_l) \in VTDB_t$

begin

if $((\exists(v^i_{m-n}, t^i_o) \in VTDB_r)$ **AND** $(k = m))$

then $VTDB' = VTDB' \cup \{(v^i_{j-k}, t^i_l)\}$

end

end.

Algorithm MEETS^{-1}QUERY

Input: *A target VideoText database $VTDB_t$ and a reference VideoText database $VTDB_r$;*

Output: *A new target VideoText database $VTDB'$;*

begin

foreach $(v^i_{j-k}, t^i_l) \in VTDB_t$

begin

if $((\exists(v^i_{m-n}, t^i_o) \in VTDB_r)$ **AND** $(j = n))$

then $VTDB' = VTDB' \cup \{(v^i_{j-k}, t^i_l)\}$

end

end.

References

[1] Gulrukh Ahanger, Dan Benson, and T. D. C. Little. Video query formulation. In *Storage*

and Retrieval for Image and Video Database II, IS&T/SPIE Symposium on Electronic Image Science & Technology, San Jose, CA, February 1995.

[2] C. Mic Bowman, Peter B. Danzig, and Darren R. Hardy. Harvest: A scalable, customizable discovery and access system. Technical Report CU-CS-732-94, Department of Computer Science, University of Colorado-Boulder, 1995.

[3] C. W. Chang, K. F. Lin, and S. Y. Lee. The characteristics of digital video and considerations of designing video databases. In *Proceedings of the 4th International Conference on Information and Knowledge Management*, pages 370–377, Baltimore, Maryland, USA, November 1995.

[4] T. Chua and L. Ruan. A video retrieval and sequencing system. *ACM Transaction on Information Systems*, 13(4):373–407, October 1995.

[5] W. B. Croft and D. J. Harper. Using probabilistic models of documents retrieval without relevance feedback. *J. Doc.*, 35:285–295, 1979.

[6] Gloriana Davenport, Thomas G. Aguierre Smith, and Natalio Pincever. Cinematic primitives for multimedia. *IEEE Computer Graphics & Applications*, pages 67–74, July 1991.

[7] Ahmed K. Elmagarmid, Haitao Jiang, and et al. *Video Database System: Issues, Products and Applications*. Kluwer Academic Publishers, 1996. In Print.

[8] N. Fuhr. Models for integrating retrieval and database systems. *IEEE Data Engineering Bulletin*, 19, 1996.

[9] Arun Hampapur. *Design Video Data Management Systems*. PhD thesis, The University of Michigan, 1995.

[10] Arun Hampapur, Ramesh Jain, and Terry Weymouth. Digital video indexing in multimedia systems. In *Proceedings of the Workshop on Indexing and Reuse in Multimedia Systems*, August 1994.

[11] Rune Hjelsvold and Roger Midtstraum. Modeling and querying video data. In *Proceedings of the 20th International Conference on Very Large Data Bases*, September 1994.

[12] Mikihiro Ioka and Masato Kurokawa. Estimation of notion vectors and their application to scene retrieval. Technical Report 1623-14, IBM Research, Tokyo Research Laboratory, Shimotsuruma, Yamato-shi, Kanagawa-ken 242, Japan, 1993.

[13] Haitao Jiang and Jeffery W. Dailey. Video database system for study animal behavior. In C.-C. Jay Kuo, editor, *Proceedings of SPIE Multimedia Storage and Archiving Systems*, volume 2916, pages 162–173, Boston, MA, November 1996. SPIE.

[14] Haitao Jiang, Abdelsalam Helal, Ahmed K. Elmagarmid, and Anupam Joshi. Scene change detection techniques for video database systems. *ACM Multimedia Systems*, 1996. Accepted.

[15] 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 *Proceedings of First ACM International Conference on Multimedia*, pages 427–436, Anaheim, CA, August 1993.

[16] T. D. C. Little and A. Ghafoor. Interval-based conceptual model for time-dependent multimedia data. *IEEE Transaction on Knowledge and Data Engineering*, 5(4):551–563, August 1993.

[17] M. J. McGill and G. Salton. *Introduction to Modern Information Retrieval.* MacGraw-Hill, 1983.

[18] Kok Meng Pua. Prototyping the VISION digital video library system. Master's thesis, University of Kansas, 1993.

[19] G. Salton and et al. Extended boolean information retrieval. *Communications of the ACM*, 26:1022–1036, 1983.

[20] Michael A. Smith and Alexander Hauptmann. Text, speech, and vision for video segmentation: The informedia project. In *AAAI Fall 1995 Symposium on Computational Models for Integrating Language and Vision*, 1995.

[21] Thomas G. Aguierre Smith and Glorianna Davenport. The stratification system: A design environment for random access video. In *Workshop on Networking and operating System Support for Digital Audio and Video*, 1992. Association of Computing Machinery.

[22] Deborah Swanberg, Chiao-Fe Shu, and Ramesh Jain. Architecture of multimedia information system for content-based retrieval. In *Audio Video Workshop*, San Diego, CA, November 1992.

[23] Deborah Swanberg, Chiao-Fe Shu, and Ramesh Jain. Knowledge guided parsing in video database. In *Electronic Imaging: Science and Technology, San Jose, California*, February 1993. IST/SPIE.

[24] C. J. van Rijsbergen. *Information Retrieval.* Butterworths, 1979. Second Edition.

[25] Ron Weiss, Andrzej Duda, and David Gifford. Content-based access to algebraic video. In *IEEE International Conference Multimedia Computing and Systems*, Los Alamitos, CA, 1994.

[26] S. K. M. Wong and Y. Y. Yao. On modeling information retrieval with probabilistic inference. *ACM Transaction on Information Systems*, 13(1):38–68, January 1995.

[27] Boon-Lock Yeo. *Efficient Processing of Compressed Images and Video.* PhD thesis, Princeton University, January 1996.

[28] Hong Jiang Zhang, Yihong Gong, Stephen W. Smoliar, and Shuang Yeo Tan. Automatic parsing of news video. In *Proceedings of IEEE Conference on Multimedia Computing Systems*, May 1994.