Multimedia Search and Retrieval: New Concepts, System Implementation, and Application

Qian Huang, Atul Puri, Member, IEEE, and Zhu Liu

Abstract—We first present new concepts applicable to the design of multimedia search and retrieval schemes in general, and to MPEG-7 in particular, the multimedia description standard in progress. Raw multimedia data is assumed to exist in the form of programs that typically consist of a combination of media types such as visual, audio, and text. We partition each such media stream into smaller units based on actual physical events. These physical events within each media stream can then be effectively indexed for retrieval. The concept of logical events is introduced next; we define logical events as those that can provide different “views” of the content as may be desired by a user. Such events usually result from either the correlation of events that cross different media types, or by merging recursively chosen events from a lower level within each media type.

We then address the related issue of how to develop a practical multimedia information retrieval system that exploits the aforementioned concepts of physical and logical events as well as other aspects such as storage, representation and indexing to enable efficient search, retrieval, and browsing. Finally, we implement the proposed concepts and solutions within a multimedia system that addresses a real application, effective browsing of broadcast news, and evaluate its performance.

Index Terms—Description schemes, multimedia browser, multimedia search, MPEG-7, search and retrieval, search system.

I. INTRODUCTION

WITH THE continuing explosion of multimedia information in today’s society, searching for information of interest is becoming increasingly difficult. Typically, much of this information needs to be accessed at a time later than when it is generated, by people other than who generate it, and with effort much less than that required by a fully manual search. Further, this information must often be accessed in a form that supports ease of editing or modification for standalone presentation or incorporation in bigger applications. While efficient compression of multimedia, as it exploits the inherent redundancies in the data, reduces the storage requirements at the expense of selectively introducing imperceptible artifacts, the resulting compressed data is usually more fragile. Of particular relevance to our discussion are MPEG standards. For instance, the recent MPEG-4 version 1 [1] standard addresses the representation of multimedia content as objects for the purpose of coding and presentation. The continuing work on MPEG-4 (version 2) further addresses presentation and management of content, flexible control of playback, a file format for storage, as well as simple descriptors to describe the content. The next logical step would seem to be development of techniques for flexible organization, search and access of these coded objects so that databases of such content can be created; this is being addressed by MPEG-7.

Next, we will discuss the motivation leading to MPEG-7, followed by a brief introduction to MPEG-7.

Most of the search engines still only allow search of textual documents [2], although search and retrieval with images and visual features has been known for quite a while [3]. Furthermore, these engines use proprietary and nonstandardized descriptors, requiring users to have expertise in the different syntax used by each engine. This typically results in difficulty in specification of a complex search and the result is usually unsatisfactory. A goal of MPEG-7 [4] is to enable search for multimedia and to improve the current situation related to proprietary solutions by standardizing an interface for describing multimedia content. MPEG-7 expects to standardize both descriptors and description schemes that may be associated with the content itself to facilitate fast and efficient search. Thus, audio-visual content with associated MPEG-7 metadata may be easily indexed and searched for.

MPEG-7 aims to address not only finding content of interest in “pull” applications, such as that of database retrieval, but also in “push” applications, such as selection/filtering to extract content of interest within broadcast channels. MPEG-7, however, does not aim to standardize algorithms and techniques for extraction of features or descriptions, or for that matter, searching and filtering using these descriptions. It is expected that MPEG-7 will work with not only MPEG coded content but also with content coded by other methods.

While the MPEG-7 work has only begun recently, other effective concepts and systems for image and video search already exist in the literature [5]–[10]. Some attempts [11]–[13] to utilize the characteristics of the related media have also been undertaken, mainly for very specific applications. Further, an attempt has also been made at utilizing the higher level semantics and structure [8], [14]–[17]. While extracting multimedia content is an extremely important problem, key issues related to the system design are often not addressed in the literature. This work differs from the earlier work in that: 1) it exploits cues from different media in content based indexing and 2) it addresses search and retrieval of multimedia data not mainly from content-extraction perspective, but from system-design perspective. The focus of this paper is thus on system-related issues; for details of automatic multimedia content extraction, see [8], [14].

We assume that the raw multimedia data is in the form of physical programs typically consisting of a combination of one or more of the following media types: visual, audio and text. In these media, events can be extracted (manually, automatically,
or semi-automatically) at different levels of abstraction. Such events may or may not exist physically in the data stream. For example, say a user wants to access news clips that report on two stories, one related to a tax bill on tobacco before the congress, and the other to the war in Bosnia, from a number of sources over a period of time. Two separate virtual events can then be formed over time by integrating portions of individual daily reports from a variety of sources. Notice that these events do not physically exist, furthermore, they are quite different in nature and may be composed probably quite differently. Recognizing that such needs are realistic, a multimedia system should not only provide flexible tools to allow users to edit and to compose any virtual event, but also accordingly support the storage as well as search and retrieval of both physical and nonphysical (logical) multimedia content in a seamless way. In this paper, we address these issues and propose solutions for a coherent system design that facilitates more sophisticated search and retrieval needs. Our proposed solutions cover system architecture, data storage and representation, database and its remote access, and human-machine interfaces. We then design, develop, and implement a system for broadcast news retrieval that utilizes the aforementioned principles. It is also hoped that the proposed system can offer a concrete example of capabilities required from the MPEG-7 system and serve as a useful tool in the MPEG-7 experimentation model [18] and working draft development.

The organization of the rest of the paper is as follows. In Section II, we formulate the main problem addressed. Section III discusses the unique characteristics of each media types considered. Section IV defines the new concepts and their properties. Section V illustrates the overall system architecture. Section VI discusses several specific issues in implementing the proposed system architecture. Section VII presents the system configuration for the specific application of broadcast news browsing. Finally, in Section VIII, we summarize the findings of the paper.

II. CONTENT-BASED MULTIMEDIA SEARCH AND RETRIEVAL

There are many traditional, as well as upcoming, applications that can significantly benefit from the content based search and retrieval capabilities. Examples include areas such as education, business, entertainment, and sports. In general, search and retrieval is increasingly becoming a basic functionality needed as part of a bigger multimedia application or service. It can either directly be an enabler of a desired functionality to the user, or indirectly allow customization of available functionalities (such as, filtering) for the user. Thus, it is not very accurate to list applications of search and retrieval, but rather applications that employ search and retrieval. Further, search and retrieval is typically associated with pull applications, in which the user wants to locate and extract a piece of information from a database. However, within the context of new upcoming applications, the term “search and retrieval” is also applicable for push applications, in which the user wants portions of broadcast information to be selectively presented among many choices. Furthermore, search and retrieval should be possible within real-time as well as non real-time applications, the real-time being the ones where content indexing information is generated in real-time and associated with it alongside the content.

The focus of this paper is first to develop the primary concepts needed for object-based multimedia search and retrieval, then to develop a systems architecture that can effectively use these concepts, and finally, customize the system within context of a real-world application and demonstrate its effectiveness. We assume that the content is multimedia, i.e., containing different media components correlated by the context of the application data. The problem we are addressing is important because, as mentioned before, current search and retrieval solutions are mostly proprietary and emphasize mainly textual document retrieval with very little or no support for multimedia documents. Even though some existing solutions claim to employ media information other than text, they do not produce satisfactory results because the multimedia information is only partially exploited. We aim to integrate features and characteristics across different media types and develop system solutions that facilitate not only flexible multimedia content indexing, but also efficient content management. We believe that both these aspects are quite relevant to MPEG-7.

III. MEDIA TYPES AND THEIR PROPERTIES

Depending on the context of an application, either different elementary media types (e.g., audio, video, and text) or their combinations may be relevant. While each media type may have common features such as associated timing information, the media types may also be quite different due to their unique characteristics, e.g., the size of their independent access units. Traditionally, in literature, different media are typically dealt with completely separately, including how they are coded or how the content is described, etc. For transmission or data access at a lower level, it may in fact be reasonable and even efficient to deal with each media type individually. The main reason is that succinct description of data at that level requires exploitation of the distinct characteristics of the signals. Therefore, utilizing the specific properties of different media in designing appropriate description schemes leads to much more efficient representations. In other words, proper description of data at lower levels is usually derived in a bits centric mode. However, while considering multimedia information at a more conceptual level, the requirement of access becomes more semantic or content centric. Thus, the bits-oriented description strategy may not only be inappropriate, but simply inadequate for indexing and retrieval at the semantic levels. We now examine in more detail such differences in properties of the individual media.

A. Audio

Audio data may be comprised of speech, music, or a combination of the two. Since the audio signal is inherently one-dimensional with the notion of time associated with it, the conventional search and retrieval on audio data is strictly linear. To enable content-based retrieval, the audio stream has to be segmented and indexed first. Audio events can be defined as continuous audio segments that have some distinct acoustic characteristics. For example, such events can be single-speaker speech, music, multi-speaker conference speech, or speech over background music and can be analyzed and classified in a proper feature space [11]. The classification process can produce the
required partitioning of the audio stream. Once an audio stream is partitioned, each event may be stored and described individually by their most unique features, indices to these events can be established, and content-based search and retrieval can then be accomplished using the indices.

Although indexing audio events via segmentation improves the retrieval capability, there are other retrieval requirements, such as conceptual events, that are more logical rather than physical data chunks. Such requirements demand not only the development of more advanced technologies than the ones that simply physically partition the data stream, but also the system support to manage audio data aggregation. The example in Section I (stories on tax bill, and Bosnia war) illustrates such needs. It is obvious that logical events are related to how users view the data, as opposed to how the data exists physically, and that they are usually a composite of lower level events that can be physical or logical. An important feature of any high-level event is that it most likely to be application dependent.

B. Visual

Visual data may comprise of still images, natural video, or synthetic animations. In context of applications of our interest, we focus mainly on video data. Video conveys visual information that includes both 2-D space and time. The video data can be acquired as a stream of frames at some lines of resolution per frame and an associated frame rate. As is obvious, video frames since they capture a sense of motion in the scene have a notion of time associated with them. After digitization, one can associate a spatial resolution in the horizontal and the vertical direction with each frame. Different events may be extracted from digitized video, either in 2-D space or in 2-D space and time. For example, a static logo in a video scene may be considered as a 2-D event, while a moving object may be considered as a 3-D event. Similar to audio, such events could be defined as physically continuous data segments each with distinct visual characteristics. Such physical event partitions video frame(s) as either belonging to the event or not. Although indices to different visual events offer improvement over linear search, the search and retrieval capability is still limited to what physical data offers.

To enable users the flexibility of defining logical events specific to their own applications (e.g., a collection of family members videos taken in different cities, a virtual family reunion video can be created by merging the indices of these clips to generate a new event which can be retrieved at later times just as if it were taken physically), a multimedia information system has to provide: 1) the means for users to create, edit, describe, and manipulate conceptual events and 2) the system support to store, search, and retrieve not only physical, but also logical, events.

C. Text

Although more than one standard already exists for representation of text, within the context of multimedia, new standards may still be needed to facilitate search and retrieval. This is so because of the need to capture the correlation between text and other media. Text, say in ASCII, can be thought of as a collection of characters (including letters and special symbols). By semantically combining continuous segments of these characters, we generate meaningful units such as words, sentences, paragraphs, etc., for the purpose of conveying information. Depending on the application, a text transcript as an adjunct to audio and/or video may or may not be available, or required. In some cases, it may be possible to generate text transcript either semi-automatically or automatically, for instance by recognition of the semantic units in the speech of a speaker. Furthermore, in the manner similar to audio and video, for synchronization purpose text may also have a notion of time associated with it.

Thus, segments of text can be thought of as physically partitioning the text stream. Similar to audio and video, from a set of lower level physical events, many different types of logical events of text can be formed. Examples of textual events include dialogues, stories, story categories, and news summaries.

IV. MULTIMEDIA DATA ABSTRACTION AND RETRIEVAL

Within the context of multimedia, recognizing the difference between data access at the lower and higher levels, we introduce the concepts of physical and logical layers. A physical layer addresses the media events that are time continuous and better described using the specific characteristics of a media signal. A logical layer addresses multimedia events that present integrated behavior of other events across media, across the event hierarchy within each media, or a combination of both. The necessity to make such a distinction exists mainly because by identifying the nature of each layer, efficient representation scheme for that layer can be developed (see Fig. 1). This preserves the unique features of each layer and enables a transparent interface for users to seamlessly access information of all levels.

To meet a wide range of application needs, advanced multimedia systems have to also provide users the flexibility to build higher level events when necessary. To meet such requirements, we also introduce concepts such as physical program, physical event, logical event, and content structures. In later sections, we then address the problem of providing adequate system support for these heterogeneous types of events and layers, all in one coherent architecture.

A. Physical Program and Physical Event

A physical program is a continuous stream of multimedia data that begins and ends due to the physical scene edits and cuts decided by human operators. Examples of physical programs include things such as a two-hour movie, a half-an-hour news broadcast, a video of a 1999 reunion of a family, etc.

A physical event is a continuous segment of multimedia data, representing a coherent and meaningful event within a physical program. A physical event may have counterparts across the various media types. For example, a news story on a particular topic will have corresponding parts in both the audio and the video streams, with each possibly having different boundaries in time.

Physical events can form an event hierarchy, where higher level events are aggregations of lower level events and the events at the lowest level as primitives. All physical events and hierarchies reside at the physical layer. As a summary, a physical event is: 1) continuous in time; 2) specified by a single continuous time range (between a beginning and an ending time; 3) uniform in some proper feature space; and 4) more likely to be
Fig. 1. Relationship among different event layers.

efficiently represented when a common underlying characteristic is exploited.

B. Logical Event

A logical event is a virtual view of the multimedia data constructed from a set of physical programs or events, or recursively from other physical or logical events. It is application dependent and satisfies a goal for the user not satisfied by a physical event. Thus, a logical event is not a primitive. Logical events can also form hierarchies of their own, but they reside on top of the physical layer. Therefore, a logical event is a collection of events that are semantically coherent; it contains possibly disconnected component events. A logical event is thus neither specified by a single continuous unit of time (enclosed by a beginning and an ending time) nor by a uniform feature space. Thus, its aggregation has to be explicitly described, so that the event can be unambiguously rendered when it is retrieved.

C. Semantic Content Structure

We define a specific type of logical event referred to as a semantic content structure. This type of event is also conceptual and completely virtual. Such events provide a short description or quick summarization of the structure of the underlying multimedia data. An example of a semantic content structure is a Table of Content (ToC). Such an event can be thought of as metadata, generated based on the interpretation of structures in the actual content of one or more related media. Such events are essentially artificial, but represent a mechanism to present hierarchy information embedded in structured data for the purpose of facilitating user interaction. Note that it is in these purely artificial events, the distinction among all the events at lower levels, physical and logical, disappears, enabling a seamless interface to the end users.

Although we have distinguished events to be physical or logical so that they can be efficiently represented, stored, and represented, such a distinction is mainly for system design reasons and should not affect the end users. In order to provide a seamless search and retrieval experience, a system that represents and manages different types of media information is needed. Thus, a coherent design is necessary to bridge the gap. We next address a number of additional important issues relevant to design of a practical system for media information retrieval and browsing. These issues include but are not limited to layered representation of media data, access to remote databases, and a unique interface.

V. SYSTEM ARCHITECTURE

A high-level architecture of our multimedia information search and retrieval system is shown in Fig. 2. While the primary components of our system are similar to that of other such systems and consist of a server-end and a client-end, there are however, some difference in details. On a conceptual basis we envisage two servers for each server site: a multimedia data server and a web server. Although both such servers can be implemented on one physical machine, it may be more desirable to run them on separate machines due to both performance and system independence reasons.

In our system, the raw multimedia data is first digitized, compressed, and stored. The actual compression technique used depends on the media as well as requirements and features of an application. Furthermore, we attempt to retain only the representative key information likely to play a significant role in search and retrieval depending on the media itself. For example, our system may retain only compressed key frames as the raw data for video by applying scene cut (in real-time) to the original video data stream. Advanced segmentation techniques can then be applied using either the manual editing tools or the automatic tools to analyze the media data to extract and index meaningful events at different layers which can subsequently be used to construct the content structure event, serving as a compact yet effective index table or a roadmap for the user. This structurally organized media content is then stored on the media server. In our system, upon request from the client side, this structured content will be accessed, via use of Java database connectivity (JDBC)
Fig. 2. Architecture of our multimedia search and retrieval system.

and remote method invocation (RMI), and delivered through the network to the remote client.

VI. SYSTEM IMPLEMENTATION

A. Server Implementation

To support the functionalities at the server, a number of technical problems need to be addressed, including how to extract the content, how to represent the retrievable units, how to populate the content and its indices in the database (in an accessible easy to share, and efficient manner), as well as how to facilitate the query and retrieval between the server and the client. We next address these issues.

1) Content Hierarchy Extraction: Analyzing and segmenting multimedia data manually can usually be very time- and effort-consuming. Thus, semiautomatic or fully automatic methods are preferable, although they are likely to be error prone. Next, the segmented content data needs to be indexed. Here, it is important that the index be reflective of and be able to capture the significant transitions in the data, but at the same time result in a condensed view of the data so that effective search and retrieval becomes possible on this condensed index table. Automatically extracting meaningful content has been one of most widely addressed topics in multimedia research. Existing automated tools can provide capabilities in extracting physical level events (e.g., scene cuts) and higher level logical events (e.g., story integrated over time). Many such tools perform a bottom-up processing, extracting content based purely on signal properties. Hopefully the MPEG-7 standardization work will also generate a need for tools in the future that can allow content extraction just by analyzing the meta information from the content provider; this meta information may be delivered together with the media data.

It is important to recognize that the high level-information embedded in the content is very application dependent. Thus, the design of content-extraction techniques on the server end should consider this. On the other hand, at the level of primitive physical events, algorithms can be automated in an application-independent way. Further, at the level of physical event layer, besides the automated solutions, the system should also provide flexible and semi-automatic editing tools so that users can identify retrievable units of their own interests. At the level of logical events, description schemes should be offered that allow users to form their virtual view of the data from available lower level events. Fully automated solution can only be designed and applied when there is a well-defined application domain.

An application that we present in Section VII uses fully automated techniques on the server end that extract a set of predetermined events such as physical, logical, and structural events for broadcast news browsing. Although organizing multimedia content in an efficient and suitable manner for search and retrieval is a very important issue, we do not delve into these details due to the different focus of this paper. Thus, the rest of the paper concentrates mainly on system design and application development issues. For a detailed technical description of the segmentation algorithm, we refer readers to [14] and [8].

2) Layered Content Storage: We consider two alternative forms of information storage techniques: file systems and databases. Typically, a file system format and its management is often proprietary. While its data creation and access is more direct and thus efficient due to small overhead, the maintenance of the system can be tedious, time consuming, and error prone. On the other hand, databases offer a standard and systematic technology for information management and provides adequate and flexible retrieval functionalities, such as keyword searching, sorting based on a number of criteria. Although the management overhead of databases is higher, the maintenance is easier and transparent. In addition, standard database systems support backward compatibility so that it is more suitable for data porting and upgrading.

In a multimedia information search and retrieval system, because of the hierarchical nature of the content and the large amount of information involved, it is desirable to store the data in different layers. The lower the levels are, the larger
the amount of information is and further away from the applications. At the same time, the higher the levels are, much of its content contains mostly meta data (significantly smaller amount) yet closer to the applications. Hence, to achieve the balance between data access efficiency and portability, we propose a layered content storage solution: store lower level information in a file system and higher level information in standard database systems.

With synchronized audio/video/text information, the feature that holds different media together is the time. When raw data is stored in a file system, different events in the data stream can be accessed using time based indexing. Thus, for a given media, timetables with respect to other media can be generated according to the partition for that media, such as by the physical events at the primitive level. Timetables can also be stored in the same file system as the raw data. Hence, a description of events, both physical and logical, can be stored in databases. Since a logical event is usually an assembly of other events, it does not need to directly index into the stream data but to indirectly rely on its component events to do the job.

Fig. 3 presents the layered configuration of the proposed content storage. At the bottom, the synchronized audio/video/text is stored in the file system. Three timetables corresponding to different media are stored in the file system above the raw data. Each table records the partition information (segments in time) at the lowest physical event level. Each segment in the partition has an entrance in one of the event tables in the database. For example, if a segment in the audio stream is the speech of an anchorperson in news scenario, there will be an enclosed time frame in the audio timetable as well as an entry in the database table that describes different audio events, where the descriptors for this anchorperson speech event will have a link pointing to the counter entry in the audio timetable. This is illustrated in Fig. 4. Section VII introduces our implementation of the layered storage for broadcast news content.

3) Database Content Access: As mentioned earlier, we employ JDBC to provide connectivity to our database for query and update. JDBC is built around the well known and robust concepts such as structured query language (SQL) and call level interface (CLI). Since JDBC is based on these standard concepts, it can be implemented on top of existing content databases. Further, programs developed with Java and JDBC are both platform and vendor independent.

Fig. 5 illustrates how JDBC communicates with a database. The JDBC interfaces are shown as shaded areas. It consists of two layers. The topmost layer is the JDBC API, which communicates with the JDBC driver manager, sending various SQL commands. There are two ways for the JDBC driver manager to interact with the database. First, it can communicate with different third-party drivers that actually connect to the database and then either return the information being queried or perform the action requested by the query. Alternatively, it can communicate with the ODBC driver through JDBC/ODBC bridge. Although more efficient, the obvious shortcoming of the first is that it requires matching the JDBC driver with the specific database being used and such drivers are not always available. In our current implementation, we chose Microsoft Access 97 as the database for managing our multimedia content because it comes with an ODBC driver and the JDBC/ODBC bridge is available from Sun Microsystems in conjunction with the Java development kit (JDK).

B. Client Browser Implementation

Since multimedia data is stored on the server and the amount of data can be quite significant, effective techniques for interactive query, multimedia streaming/playback, and human machine interface are essential.

We chose to implement a Java applet-based interface on the client end because of its platform independence and its support for a variety of coded media. Another alternative would have been to use a web-browser plug-in approach, which obviously has the shortcoming of platform dependence. In addition, a Java applet gets downloaded automatically at the run time, a completely transparent process which adds only a minimum burden
to the client. For security reasons, however, it is undesirable to let an applet on the client side directly access the database on the server side. A solution to this dilemma is provided by RMI, which can be used for the communication between Java objects running on Java virtual machines, one on the client side and the other on the server side. On the server end, once the Java server is activated initially, it will keep alive indefinitely, waiting for clients to connect to it. When the client generates a query, the Java applet will search the database, retrieve the target content, render the visual presentation for the selected content, and display the presentation on the browser interface. When needed, Java media framework (JMF) is activated to stream and play back the multimedia data.

1) Remote Database Query: As noted earlier, Java offers facilities that extend the Java object model beyond the address space of a single virtual machine. This enables Java objects running on different virtual machines to communicate with each other regardless of whether or not they reside on the same machine. This is called RMI. The Java RMI is essentially an evolution of procedural remote procedure call (RPC) adapted to an object-oriented paradigm.

Fig. 6 illustrates the mechanism of RMI. When a client application invokes the method of a remote object, it calls a regular Java method encapsulated in a surrogate object, called a “stub,” that resides on the client (client stub). This stub builds up an information packet consisting of an identifier of the remote object, a number describing the method to be invoked, and the parameters to be marshaled, and then sends this information packet to the server. On the server side, a skeleton object (server stub) interprets the information contained in the packet, passes the interpretation to the actual object, calls the desired method of the object, captures the return values or exception from it, and then sends a packet consisting of the values back to the client stub. The client stub unmarshals the information and hands it to the application as the return value from the remote method.

In a client/server-based information retrieval application, every retrievable unit can be identified by a distinct key. When the unit is requested, the key is uniquely determined at the client side and then delivered to the RMI mechanism to invoke all related remote methods that retrieve the desired content associated with this key from different database tables on the server. The returned content contains not only the data itself, but also the meta information that describes how the content should be rendered for playback or presentation purposes.

2) Browsing Interface: When the amount of multimedia content stored on the server is large, it is important to provide the users on client side access to information about what can be retrieved or what is available as well as the semantics of the content retrieval process. The first is related to the structure of the content data, while the second is related to the semantics of the content. Both are important aspects in developing an effective human-machine interface.

Further, a good presentation is critical to achieve high user satisfaction in interactive retrieval process. Presentation addresses how to effectively present the content to the user which differs from representation, which mainly addresses what information should be stored and how. The two are however related because what is being presented is based on what and
how it is represented. We propose two forms of presentation. The first, for describing the content structure, is ToC, and the second, for presenting the content semantics, is TimeLine.

*Table of Content: A Conceptual Content Structure:* A Table of Content (ToC) is a familiar structure to humans due to the use of such a form for providing quick information about contents and structure of a book. It also serves as an index so that through a quick glance at it, one can get a general sense of the content and its organization. Decisions as to what to look further can then be made based on such initial understanding. The familiar concept of a ToC can also be applied to organization of digital multimedia content.

In a multimedia system, a higher level view of the content structure helps users to get a rough idea what is available. Users can then use this information to decide what to retrieve. For example, the ToC for a broadcast news program may indicate that there are two categories of content, one is the headline stories and the other is commercials (see Fig. 7). Based on the information given in such a presentation, a user who is interested in only news stories can avoid choosing any commercials.

To equip a multimedia information retrieval system with such capabilities, the content structure of a physical program has to be represented explicitly from which the browser can then render a ToC. For example, if the aforementioned program contains six headline stories (assuming they have been extracted), the browser should render such meta information explicitly on the interface (see Fig. 7).

*TimeLine: A Content Structure in Time:* While a ToC presents a conceptual summarization of the content organization, it does not offer any information about how the content is distributed in time. Since multimedia content is typically time varying and often there is need to index with finer granularity than that supported by the ToC, we propose, in conjunction with ToC, another form of presentation of structure of the content called the TimeLine. It simultaneously expresses both the structure of the content as well as the distribution of the involved events in time. The ToC and the TimeLine presentations are designed to complement each other. When a specific event is chosen, the TimeLine will simultaneously indicate where in the program this event resides. In Section VII, within the context of broadcast news application, we illustrate the implementation of both the proposed presentation forms.

*Event Presentation:* We further discuss the issues of how to present the semantics of the events. While the ToC and Time-
Line are designed to visualize the structure of the content data, the most important criterion in designing the presentation for an event is to visualize its semantics by selectively displaying pieces of information from the event in an extremely compact form.

To achieve this goal, we first make a distinction between presentation and playback. In the context of multimedia search and retrieval, we refer to presentation as a static description of some event generated to reveal the essence of its content. Playback is the process in which the multimedia event under query is reconstructed and played, in full length, under user control. Thus, with respect to retrieval, the former occurs before the retrieval and the latter occurs after the retrieval. The two serve very different purposes: the former facilitates a user in making a retrieval decision, and the latter simply recreates the natural flow of the multimedia content. Clearly, the requirement for presentation to be semantically very manner is much more intense.

Using our terminology, since audio is used mainly only in playback, video and text are the only media used in generating a presentation. Appropriate techniques can thus be developed to automatically extract the visual and textual information of an underlying event based on a predefined criteria. For text, it can be as simple as identifying significant words. Automatically extracting most relevant visual content information for the construction of a presentation is a much more difficult task. Algorithms available in the literature to perform such tasks are rather limited [16], [14], [8]. We discuss elsewhere our solutions for automatically generating a small set of keyframes for visual presentation [8], [14]. Applications of such techniques to various types of events in the context of broadcast news data are shown in Section VII.

3) Content Playback with a JMF Player: A JMF player is an object that processes a time-based media stream, reading data from the source and rendering it at a precise time. Client applications can create and control JMF players for a number of standard media types. JMF can also be extended to support additional media formats and protocols. JMF players are often used synonymously for players written to comply with Java media player API- and JMF-based players can be used as either stand-alone applications or applets that facilitate incorporation of multimedia into web-based applications or services. Currently, the Java media player API supports many media content types including MPEG-1, H.263, QuickTime, AVI, WAV, AU, and MIDI. Using JMF, time-based media from diverse sources can be synchronized and presented.

The JMF player also allows controlling of the behavior of the playback. For example, one can playback either the entire media stream or a subset of the media stream. One can also group discontinuous media clips which share the same semantic meaning into one chunk for playback. The Java media player API also provides an event model for asynchronous communication between JMF players and also with other applications.

VII. AN APPLICATION: MULTIMEDIA BROADCAST NEWS

We apply the system design discussed in the previous section to application of browsing of broadcast news. We now discuss how the different aspects of the system are implemented for this application.

A. Server End

1) Content Hierarchy Extraction: Since broadcast news data is well structured, we designed automated algorithms to extract various events. At lower levels, algorithms are designed specific to each individual media to extract primitive physical events in each stream. At higher semantic level, algorithms are designed to take advantage of integrated cues that merge different events into semantically meaningful logical events, such as headline stories and news summaries.

In the audio stream, three types of physical events are identified: anchor speech, detailed news report, and commercials. In video stream, two types of physical events are extracted: scene cut and anchor appearance. In text stream, three types of physical events are constructed: word, sentence, and paragraph. Necessary alignment between different media is performed prior to event extraction. Cues from different media then are integrated to improve the segmentation quality. The extracted lower level physical events partition their corresponding streams, providing the content structure at the lowest level.

Logical events are predefined and constructed from physical events across media. For example, a common semantic unit in broadcast is headline stories. Another useful unit is news summary given by the anchor. Fig. 8 shows the predefined content hierarchy for broadcast news. Due to space limitation and the different focus of this paper, for details of content hierarchy and its automatic construction using signal processing techniques we refer the readers to [8], [14].

2) Content Storage and Access: With the content hierarchy formed, the events and their descriptions are stored in both file and database systems. In the file system, we store, for each physical broadcast news program, the timetables for audio, video, and text, individually (see Fig. 3). Each media may have multiple timetables depending on at what granularity the stream in partitioned. For example, the timetables for text can be as fine as for each word or can be as coarse as for each paragraph. In general, for each partition, there is a timetable in the file system. Our file system design and the timetables for different media are discussed in detail elsewhere [19]. The tables in the database system will index into these timetables for the actual retrieval and data access.

Physical events in each media are recorded in separate database tables at the physical event layer (see Fig. 3). Each entry in database tables records all the attributes about the event, including both its descriptors as well as the indices to the media data through the timetables. For logical events, the recorded attributes mainly relate to the meta information about how the event is formed, including what are the component events, in what media, and how they are synchronized in time and space. A logical event retrieved using such information can thus be completely rendered.

For each event, whether physical or logical a description provides the presentation information based on which the browser can render the event. The simplest case is that of a scene cut. Since each cut can be presented by a key frame, the meta information can contain simply the pointer to the key frame. A more complicated case is that of stories or news summaries. For each story, the meta data about its presentation includes the location of the keyframes
used to construct the visual presentation of the story and a set of keywords that are chosen to reveal the content of the story.

Fig. 9 shows the relationship between a headline and its component events (different speakers) in individual media at physical layer. Based on the relationships among all events, we designed a relational database schema shown in Fig. 10. Links among tables are made through primary keys defined for each table. The right-most column in this figure is the tables at physical event layer (each table is for separate media streams). The middle column is the tables at logical event layer. The left column is the content structure layer.

B. Client End

As we discussed in Section VI, there are also a number of system issues related to the client side. Next, we discuss how each of these issues is addressed in our system implementation.

1) Remote Database Query: For each broadcast news program, the physical program ID is uniquely determined by the channel and the date of the program. Once a particular program is requested, everything under it, including physical and logical events as well as content structures, are made immediately retrievable through remote database query. To enable simultaneous retrieval of different types of content under one program, five remote methods on the server end are implemented. They are watchProgram (returns program object), watchStory (returns a list of headline stories), watchAudio (returns audio physical events and their descriptors), watchStorySegment (returns individual stories and their descriptions), and watchStoryImage (returns the meta data on story presentation). Via remote database query mechanism, any retrieval request initiated on the client side is always sent to all five methods. Using returned information, the client application reconstructs whatever is needed at the browsing interface.

2) Content Presentation: We developed automated solutions to generate the presentations for different semantics. As discussed earlier, there are two levels of presentation, one is at content structure level (ToC or timeline) and the other at content level (headline stories and news summaries). For the details of these algorithms, see [8]. We now illustrate our presentations at these two levels.

   Content Structure Presentation: As discussed earlier, we have proposed two complementary forms for presenting the content structure of a news program. The first, a ToC, and the second, a TimeLine presentation. Differing from the presentation for individual events where the emphasis is to convey the content in a most visually relevant way, the presentation for content structure concentrates on providing an overall view of the semantic structure of the underlying data. Through this presentation, a user can: 1) see conceptually how many semantic units can be retrieved under the program; 2) understand how these semantic units aligned in time; and 3) retrieve any semantic units, physical or logical, through a seamless visual query interface.

   Fig. 7 illustrates the content structure presentation for a news program. The left portion is the ToC and the right is the TimeLine presentation. The ToC provides a familiar form to summarize the content, in which different events, physical and logical, are conceptually categorized each event is visually indexed, using proper color codes, into the time stamped events shown in the TimeLine presentation on the right. Within the TimeLine presentation, the time line runs from left to right and top to bottom and there are two layers of categorization at any time instance. The first layer is for physical events (e.g., anchor speech) and the second layer is for logical events (e.g., stories or news summary). Each distinct section in this representation is marked by a different color and the overall color codes correspond to the color codes used in the ToC. While the construction of each logical event is transparent in ToC (also to users), one can see how a logical event is formed from the TimeLine presentation by examining the color coding.

   As far as retrieval is concerned, there is no distinction between physical and logical events in both presentations. To re-
retrieve a particular event, a user can simply click on either the segment on TimeLine presentation or the corresponding button in ToC. There are two modes of retrieval: static (single click) and playback (double click). The former refers to retrieving
the static presentation (discussed below) of the event, through which user can further examine the content via a very concise description of the event. In this case, the raw data is not streamed over from the server. The latter refers to the ultimate retrieval where all the synchronized streams of the event is delivered to the client and played back. Another feature in Timeline is that when the retrieved unit is played back, the corresponding segment in Timeline flickers indicating where the active event is within the program. In the case of playback a logical event where a set of disjoint events will be seamlessly played back (e.g., news summary consists of disconnected anchor speeches), the Timeline will automatically flicker each component event when it is being played. Therefore, these two forms of presentation together describe the content structure from different perspectives, satisfying variety of needs from users.

**Story Presentation:** A story presentation is a static description of a news story. It is invoked when users choose to examine the content in a static view before deciding whether to play back. Fig. 11 shows an example of the presentation for a headline story, corresponding to the right portion in the figure (the ToC remains at all time during any session so that users can point to a different event at any time). On the right-hand side, the event currently chosen from ToC is displayed. As we can see, there are several components in the static presentation for a story. The upper left corner lists a set of keywords automatically selected to reflect the content of the story. On the right side of the presentation, story text is displayed. In the middle, it is the visual component of the story presentation. Images displayed are also selected automatically from the video clips within the story boundary. The criterion used is that these images carry maximum amount of information since they are synchronized to text with respect to the story keywords [8], [14]. In our experiments, keyframes chosen this way are visually informative in terms of the content. For example, the story displayed in Fig. 11 is about the El Nino in California. It is clear that the presentation in the figure is very relevant to the content.

From this presentation, users can further grasp the content without retrieving the raw data (which is usually large amount). This gives them more opportunity to evaluate and decide their retrieval options.

**News Summary Presentation:** The static presentation for news summary (a logical event) serves somewhat similar purpose as ToC except that it does not contain any information about commercials. In this presentation, all headline stories are mentioned in an extremely compact fashion. Fig. 12 gives an example of this presentation. From the presentation, one can tell immediately that there are six headline stories on that day. Each headline story is described by one keyframe with a set of keywords flicked every second underneath the image. The purpose of deriving this logical event and make it accessible in browsing is to provide a quick glance about today's news so that a user can choose which story is interesting to pursue. This is similar to the function of the first page of news papers where all the deadline news starts.

3) **Streaming Playback:** As indicated in Section VI, we use JMF to control the playback. Once a certain event is chosen
to be retrieved, a window shown in Fig. 13 appears indicating streaming playback. In this window, the upper portion shows the video and the lower portion shows text synchronized with the video and audio. Currently, we display only the key frames in the video window. The text scrolls up with time. The timing information with respect to the starting point of the program is also provided. As mentioned earlier, while the event is played back, the corresponding segment in TimeLine presentation flickers, signaling that it is active.

VIII. SUMMARY

In this paper, we have introduced a new layered approach to multimedia content representation and storage for search and retrieval. The media types supported are visual, audio and text. Each media type may be segmented into different events at different layers, which can be further aggregated into high-level events, either physical or logical, according to different application requirements. A user, depending on the need or the preference, can choose from a number of logical views each formed by a logical collection of events.

The concepts are implemented in a client/server system on a PC platform, designed for broadcast news search and retrieval. The system employs the Java programming environment as the basis and specifically uses JDBC for database access, RMI for remote object communication, and JMF for media presentation. The query allowed by (the current version of) the system is that supported by the commercially available relational database used in this system. However, the system design due to extensibility can employ new forms of query, such as those that may be supported by the multimedia description schemes of MPEG-7. In our initial tests, the system performs well for the chosen application. Further evaluation for other applications, as well as extension of the system to incorporate MPEG-7 description schemes, is expected to be conducted in the future.