**RC21444 (96156) 18 November 1998** **Computer Science**

# IBM Research Report

## Multi-Search of Video Segments Indexed by Time-Aligned Annotations of Video Content

**Anni Coden, Norman Haas, Robert Mack**
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

**Research Division**
**Almaden - Austin - Beijing - Haifa - T. J. Watson - Tokyo - Zurich**

# Multi-Search of Video Segments Indexed by Time-Aligned Annotations of Video Content

Anni Coden, Norman Haas, Robert Mack
IBM Thomas J. Watson Research Center
anni, nhaas, rlamck@us.ibm.com
Hawthorne, NY USA

**Abstract:**

This paper describes an approach to the design and implementation of a video content management system that enables the efficient indexing and storing, and searching and browsing of video segments defined by annotation of video content attributes. The annotation values represent the durations of independently indexed and potentially overlapping attributes of the video content, such as camera motion, shot distance and face count, and associated information, such as spoken words in the soundtrack. These values define segments of video of any granularity from entire programs down to arbitrarily short segments. Boolean searches on these multiple indexed criteria can be performed, and temporal intervals corresponding to video segments can be computed that reflect the union and intersection of search results meeting these Boolean conditions. We propose a relational database-based model for storing indexed attributes which is flexible and because it does not index or store any fixed temporal segmentation of the video. Rather annotation values can be stored in any way appropriate for an open-ended set of annotation methods representing attributes of video content. The system is part of a project to exploit a set of automatic annotation methods for the visual attributes of video content, and to develop a component of a complete studio for broadcasting digital HDTV (High Definition Television).

**Keywords** : Media Content Management, Video Database, Query System

## Introduction

Digital video asset management systems are becoming more pervasive as analog video tape archives are being converted to digital format, and new videos are produced in digital format. Key video management tasks are ingest, annotation or indexing, database modeling, search and browsing of stored content. Commercial systems are beginning to develop technological solutions for these video management tasks. A key challenge is annotation: developing search indices of descriptors for video content.. Aguierre-Smith and Davenport put the challenge this way: "[The organization of a] video database system is a challenging information management problem because the way that a particular sequence of images is described effects [how] a maker will retrieve it and incorporate it into a movie" [1].

However, the nature/type of metadata being stored is changing rapidly. Historically, "card catalog" metadata were associated with a video, such as title and production date. Such metadata attributes are descriptions of the video as a whole. Video asset management systems being developed in research projects and in some commercial systems (examples include ISLIP [2], based on InforMedia research prototype [3] AVID [4], VIRAGE [5], EXCALIBUR [6]) are beginning to go beyond limited sets of metadata attributes and index video content on a video segment granularity (time unit) considerably less than traditional program clips. With the advance of automatic indexing of advanced content-related features in videos programs (e.g., detecting faces or other objects, the motion of these objects, etc.), it is becoming increasingly necessary to index metadata of different types and relate them to varying

granularities in the database: to "index into the clip" [7]. The types of annotations can be open-ended, they can overlap (e.g., multiple visual events can occur within the same segment of video), and the duration of the annotation features can vary widely depending on the content of the video. This requirement adds complexity to the database storing the metadata and to the corresponding processes of querying and browsing.
What does it mean to retrieve a segment of video based on a query specification that refers to multiple annotation criteria? How can we enable users to browse video content in a way that closely relates to the annotation criteria specified in a query specification? We will address these issues in the rest of the paper.

In the system we describe here, video indexing is done by storing start and stop times of video segments corresponding to the value of a particular annotation feature (e.g., number of faces in a segment of video). A given segment of video can contain overlapping segments corresponding to overlapping annotation values. In our system, operations of indexing, searching and browsing can be defined over these segments in flexible ways: e.g., indexing may store segment specifications for a set of annotation features. A query specification may return a set of computed segments corresponding to some Boolean combination of annotation values. Browsing can consist of playback of these computed result segments, but also allows expanding the limits of playback of the larger video segments containing the result segments, or contracting the limits to playback smaller segments contained in the result segment. Our system stores and operates on segment specifications, and as we will show, provides great flexibility in searching and browsing video.

**Annotating Video Content**

The full spectrum of subject matter and format of videos that might occur in an unconstrained collection of video is quite diverse. Consider what is typically presented each day on television broadcast networks: news and weather reports, dramas, sports, sitcoms, concerts, wildlife documentaries and movies. The content of these programs vary widely in the amount of visual action, the importance of the spoken material to the visual image, the abundance or absence of humans, the presence of superimposed text, etc. In addition to program content, which typically runs for tens of minutes, a collection would include commercials and station promos, which might be at most one minute in length, and as short as ten seconds or less. The length of shots would vary widely over this material. Also, an unconstrained collection would include produced, edited and raw footage. These types of video differ greatly, in that raw footage tends to have very much longer shots, to contain segments in which there is no action or no meaningful action, and does not have closed-caption text. The challenge is to develop an approach to annotation that is open-ended, increasingly content-driven, and takes into account the temporal duration in which annotation criteria apply within the video.

Some promising work has been done on how to build systems that do represent content-driven segmentation and annotation of videos.

The stratification system of Aguierre-Smith and Davenport [1] uses a set of predefined keywords and free text to describe parts of a video. The notion is that the keywords provide a context for the free text. For example, a keyword may say “house” or “garden”, whereas the free text associated with a frame may specify “someone eating”. Searching for the word “eating” will find two separate segments, the keywords will give the context - eating in the house or in the garden. The free text is associated with a single frame, whereas a set of frames are associated with each keyword. Multiple keywords can be associated with a set of frames. All the data is stored in ASCII files and annotations are purely text-based.

Such a system, however, does not lend itself to the inclusion of other types of function-valued metadata, in particular, a value cannot be associated with a metadata attribute. For instance, it is not clear how to extend the system to store the information that there are ‘three’ faces visible in a particular sequence of frames. The system allows for only one free text description per frame and

hence, as the author points out, "simple annotations depend on the linearity of the medium and random access does not give contextual information to the search results." A content managment system has to allow for all types of data to be stored and searched and for the results to be viewed in a context as appropriate for the application.

Davis [8x] makes another case for video stream-based annotations. In particular, he points out that "different segmentations are necessary as video content/features continue across a given segmentation or exist within a segmentation boundary. Stream based annotations generate new segmentations by virtue of their unions, intersections, overlaps, etc. In short, in addressing the challenges of representing video for large archives what we need are representations which make clips, not representations of clips." After laying this ground work, his system starts with a fixed segmentation based on grouping sets of frames into shots represented as icons which are used as building blocks for a visual language which is geared towards understanding the semantics and syntax of video. Similarly to the stratification system, a video in this system can be is described by a fixed (but potentially large) set of words or icons. A drawback of it is that adding new annotations implies extending the language and all the functions operating on it.

Commercial systems ([2], [4], [5], [6]) similarly are based on storing annotations implying a fixed segmentation criteria for the video, and retrieving this fixed segmentation regardless of the annotation criteria specified in the query specification. Furthermore, the time segments are defined at ingest time and cannot be changed later on in the process.

**Multisearch and Browse Prototype**

Our prototype system does not require fixed segmentation and can accommodate different of annotation values: Annotations asserting the values of metadata attributes can be made using any scalar or vector data type as appropriate - text, number, or graph to name the most common ones. For instance, the following attributes are presently associated with a video: keywords, free text, motion direction, motion magnitude, face count, face size, shot distance, OCR text, and speech transcription. For each annotation, the exact interval of time over which an metadata attribute and its value is true is recorded and the value itself is associated with this time interval. Clearly, the time intervals recorded are of varying length. Hence, multiple annotations can overlap within any given segment of video, and Figure 1 illustrates this capability, in schematic form.

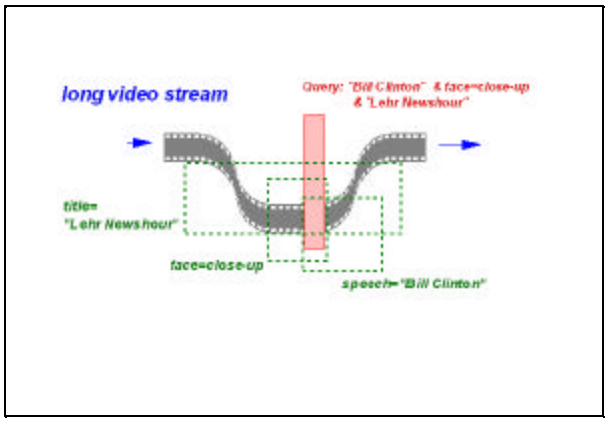


**Figure 1 Multiple overlapping annotations time-aligned with video.**

If appropriate, spatial data can also be stored in conjunction with an annotation. For instance, the rectangle in which three faces appear may be known and could be recorded. One can easily impose an x-y coordinate on a video where the x coordinate coincides with the time line and hence, only the y-coordinate needs to be stored.

A simple query involves a single attribute of the video, such as one or more free text terms (in associated text) or a value of motion magnitude. For example: find all the video segments where the word 'impeachment' is mentioned or find all the video segments where the camera motion is 'fast'. A query can also be composed of one or more attribute (attribute values) in any well-formed Boolean combination, for instance: "Find the video segments where ("impeachment" is spoken AND you see one face in close-up) OR there are two

people moving fast". The result of such a "multisearch query" is the set of time segments in which the query is satisfied. The start and stop times of these segments must be determined by computation.

Our approach to content annotation is to use modular "annotation operators", which are independent of one another (although the time intervals where their values are true might overlap). At present, the descriptors we can detect and index are: camera motion (pan, zoom), face count (0 - 6, many), shot length (close-up, long shot and several intermediate lengths, not to be confused with the shot's temporal duration), spoken text (speech recognition), closed caption text (human transcription, carried external to the image), and open captioning (text encoded as pixels in the image; usually superimposed titling, such as the score in a sporting match). In principle, there could be any number of such descriptors.

The prototype does not predetermine the unit of segmentation (granularity), other than the inherent limit of a single frame. Rather, segmentation is determined automatically, based on the video content, and as a consequence of applying the annotation operators. When a new video is annotated, annotation descriptors scan the video, and "measure" a value for each frame. This value might be a scalar, a character string or a vector, - (such as a histogram). When the value changes for some frame, the annotation operator emits a

"keyword = value (in point, out point)"

assertion, where the keyword is unique to that operator, and "in point" and "out point" are the first frame and last frame of the interval,- over which that value held. For example, the "zoom" annotation descriptor can automatically detect within a video when the camera is performing a zoom operation, and extract for each zoom operation the start and end frames and its value (e.g., 'zoom in' or 'zoom out'). Thus segmentation boundaries are a function of the value of the annotation feature in the video content itself.

The search and browse prototype also annotates text associated with video, either as caption text, or recognized speech, or manually entered descriptions. Caption text associated with video provides topic-related "new story" segments, whose duration in the video can be determined by extracting story delimiters embedded in the closed-captioning, using a caption text annotation operator. An example is a typical newscast of television evening news which is comprised of "stories." The closed caption text contains special markers ">>>" which denote news story breaks (these are manually inserted by professional transcribers). The prototype can also annotate the duration of recognized speech text associated with the video. In this case, there are no markers delimiting topic or "story" sgementations. Several options exist for recording the time at which a word is spoken. One option is to "time-align" indexed terms: i.e., track the actual time each word is spoken. However, for purposes of indexing the text content of recognized speech it is convenient to have the notion of a "document" collection. Hence, we record sequences of words (typically in the order of a 100), and index each 100 word chunk as a "document". [9] The video segment corresponding to the document is defined to be the interval from when the first word of the document was uttered until the last word is uttered. For a variety of reasons (speed, reliability of ranking, especially when the search key is multiple words), we group words into blocks (typically, order of 100), and search them first, then search the ten best matching blocks for occurrences of the individual words. In addition to indexing these "documents", we also record the time interval for each word, to provide time-aligned browsing of video by text terms. For example, suppose we are searching a sports program aired after "September 1, 1997" where the announcer shouts "goal", where the camera zooms in when this word is uttered, and where three faces are visible. After performing the search, the user interface can show the sports event by playing the video, display the announcer's comments in text format with the word "goal" highlighted, and allow the user to jump from one "goal" to another in the text and see a close-up of the event in the video at the same time.

This example indicates the "multi-search" capability of our system. This means that a user can pose Boolean combinations of different types of indexed criteria, and the system will combine component results to return a single result set. Furthermore, the system will compute the exact time intervals in which the events occured. In the next section, we will explain multi-search in a more formal way. Multi-search shows quite clearly how the flexible granularity of recording the content-related metadata comes into play. In our previous example, the knowledge of the exact time when "goal" was shouted enables us to highlight the corresponding text and play the corresponding video. However, the time interval where the camera was zooming could differ (e.g., it might be a longer or shorter relative to the segment where "goal" was shouted).

Figure 2 shows an example of a Boolean query, the component result segments, and a computed intersection.

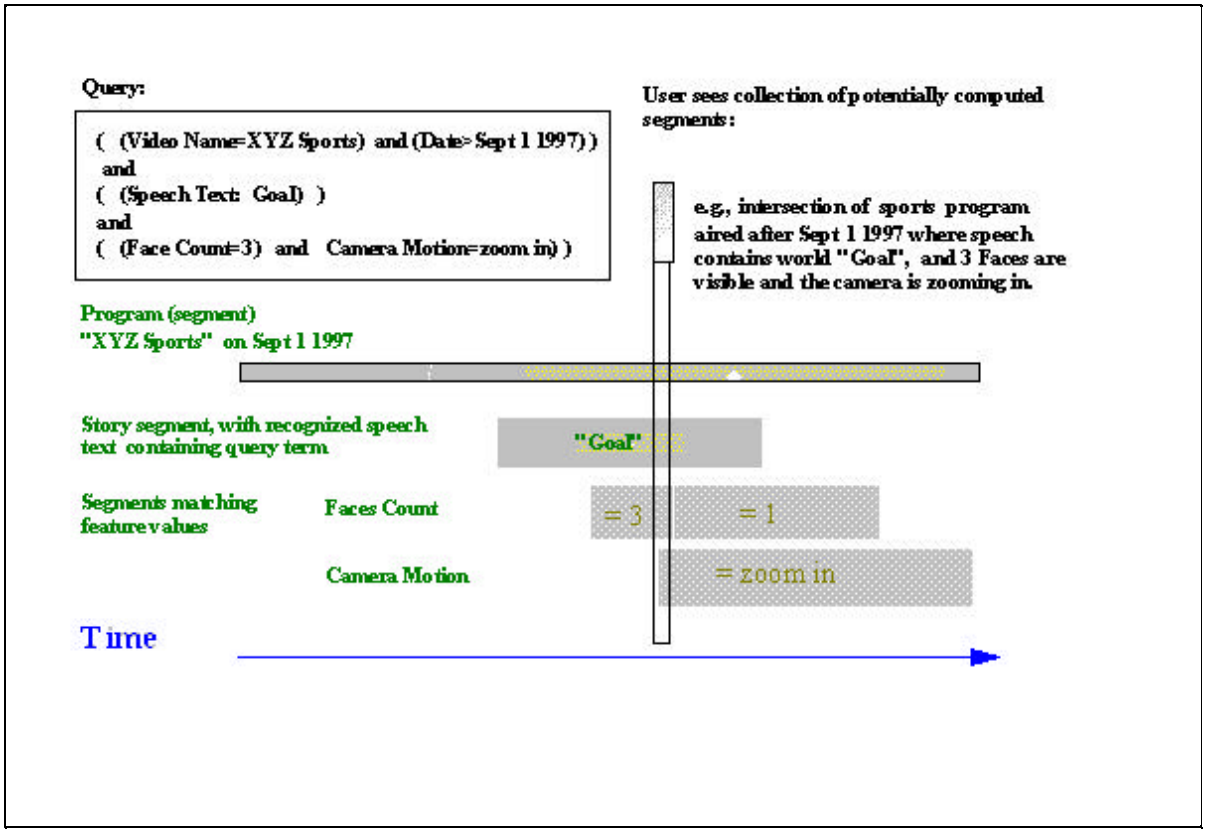


**Figure 2 Multisearch on segments: Combining results from Boolean combinations of search criteria.**

The system can compute the exact time interval in which all the specified events happen ("goal" is shouted, camera is zooming in, and three faces are shown). Clearly, the data does not to be loaded in a specific way to be able to carry out this precise calculation. Furthermore, our system is not only able to play the relevant video segment, users can also chose to view only the frames where the word "goal" is shouted, or view the whole video segment which contains the retrieved segment where zooming occurs or view only the video segments where both events occur. This enables flexible nonlinear browsing of by end users (the user interface is discussed below). The "granularity" of the browsing can be different from the granularity of the "querying", a typical desire of a user to change the specification as the application shifts from querying mode to browsing mode.

**User Interface for Search and Browse**

It is worth describing an end-user's view of these annotation methods, and the search and browsing capabilities they enable. Figure 3 shows a screen shot of a query input form for a prototype graphical user interface (GUI) which enables a user to pose queries in using multiple search criteria, including caption text and speech terms, camera motion attributes, face attributes, and program level meta data (e.g., title and play date). Using a fixed form to build query specifications limits the type of Boolean queries that can be made. Consequently, we also implemented a query language where (expert) users can specify a Boolean query of arbitrary complexity using parentheses and a query syntax (an example of a query using this langauge is shown in Figure 2).

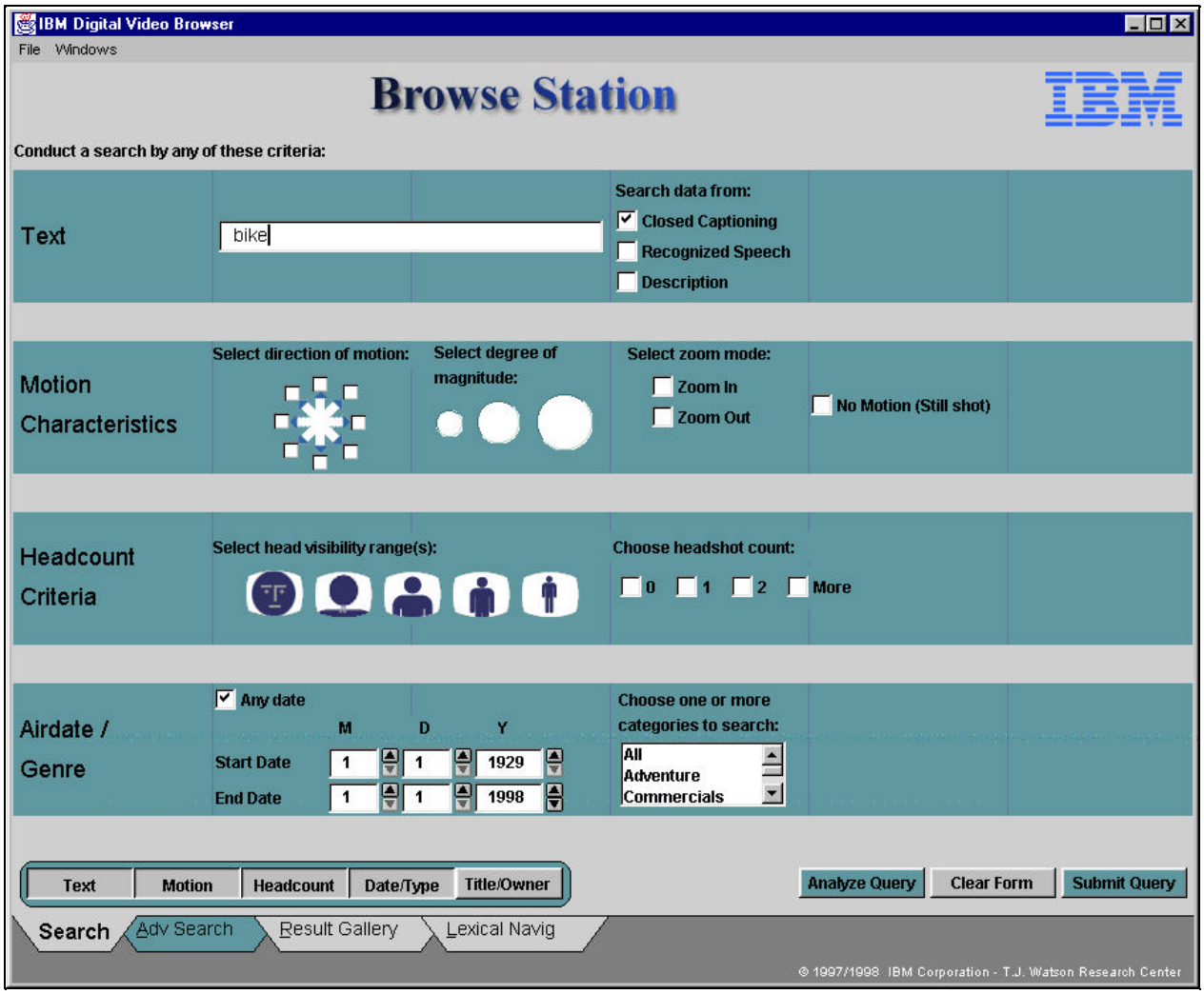


**Figure 3 Query input form for text terms, motion direction and magnitude, and program level attributes.**

The results of a search are shown in Figure 4. For each result, the relevant text and some key frames for the first few shots comprising the result segment are shown. The system also allows for "drilling down" into the content of a selected search result segment, viewing all the information associated with the segment, as shown in Figure 5 For each result segment, end users can view closed caption and recognized speech text, video content, shot story board with key frames, etc. In addition the various VCR style controls allow users to navigate through the video program containing the result segment, according to various segmentation criteria that may not have been part of the original query specification. Examples include shot boundaries, story boundaries (based on caption text markers), frames, seconds, etc. This implies that all the index feature values for the result segment are available to the browser, even if these values were not specified in the original search specification. In effect, end users are able to browse the results in terms of any indexed value expressed in the segment. The display panel of a selected search result segment becomes a starting point for flexible and broad-based browsing functions.

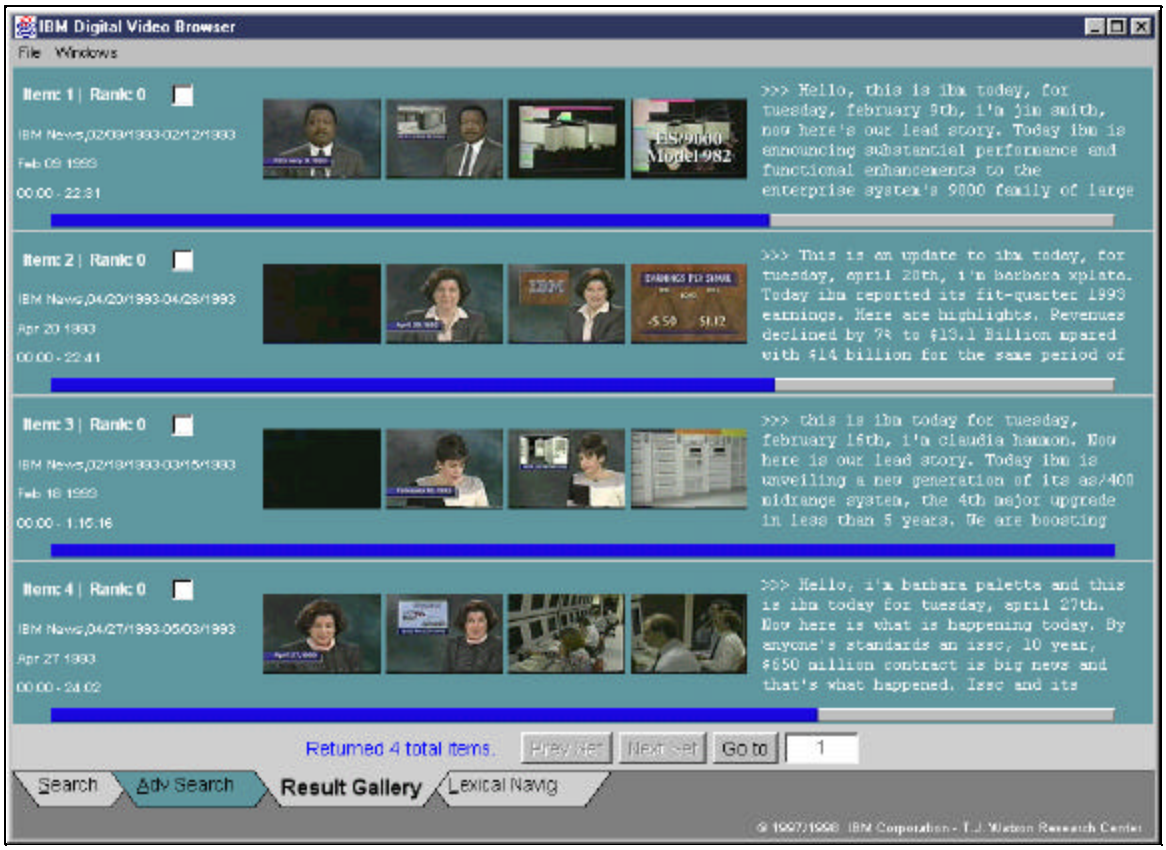


**Figure 4 Search result page, showing storyboard keyframes for video result segments and caption text excerpts.**

## Query Model and Processing

Underlying the query and browse application in Figures 2, 3, 4 and 5 is a query model, including query processing methods, designed to handle the content-oriented, and time-based annotations we have described. A query can be an arbitrary Boolean combination, including parentheses and negation of constraints, and more advanced operations related to combining results, each of which we discuss in turn.

Queries in our prototype are conjunctions of inequality constraints (including equalities) of the form "*keyword <relational operator> value*", where *<relational operator>* could include "=" and annotation operator-specific vector distance metrics along the lines of "distance metric N(value, V0) < k", for some constant vector value V0 and scalar value k. In addition, a "like" operator is supported (in SQL this corresponds to a substring match operator). Values can be specified exactly, but wild cards are also supported. For text, the constraints are of the form "contains lexical unit 1 , ... lexical unit N", where a lexical unit is a single word or a multi-word (proper name, phrase, etc.). Furthermore, a text query can have the form "contains a Boolean expression of words and/or phrases". These constraints can be augmented with a set of linguistic attributes such as language, morphology and synonyms to mention a few.

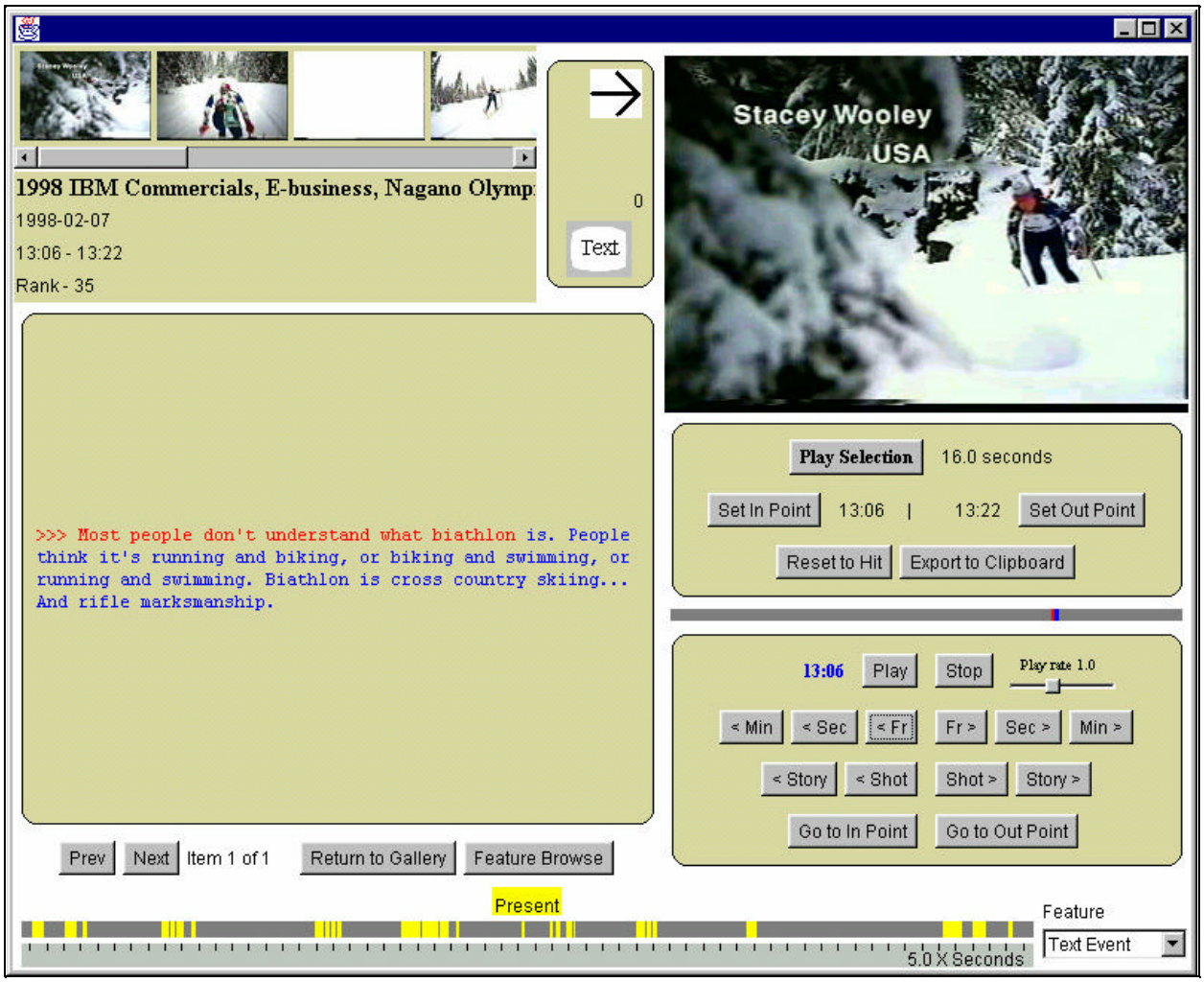


**Figure 5 Detailed content view of a specific result video segment, with video playback controls.**

For example, a query could be of the form: determine the time intervals in videos where the genre is sports, the announcer talks about goals or scores, the score is greater than three, and the camera zooms in.

The search result set produced by each constraint is a list of (interval, ranking) pairs, where each (temporal) interval is a (video ID, in point, out point) triple. The answer set for a complete query is obtained by computing, for each video, the intersections/unions of the intervals. The following pair wise interval relationships occur [10]:

- two intervals may be identical
- one may lie entirely within the other
- they partially overlap, or
- they are completely disjoint.

Finally, the intersection/union of the surviving answer set intervals is computed, to return the answer. Recall that Figure 2 showed an example of a Boolean query, the component result segments, and a computed intersection. The query can be an arbitrary Boolean expression including parentheses

of the above mentioned constraints, and hence the intersection or union is defined according to the query specification. Furthermore, other time-based operations can be performed which we discuss in the next section.

In the case of unions, arising from disjunction of constraints, the following pair wise interval relationships occur [10]:

- two intervals may be identical
- one may lie entirely within the other
- they partially overlap, or
- they abut one another.

then, the two intervals can be combined into one. Finally, the union of the surviving answer intervals is returned as the query result.

The value of this approach is the ability to handle a diversity of source material, where any attempt to define a segmentation that was roughly based around a generic time constant or a fraction of the total length of the video would not generally be appropriate. Units of segmentation that would be well-suited to some queries would almost certainly be inappropriate for others. Also, the in- and out-points for different annotation operators are, in general, completely different from each other, and consequently would generate a sparse set of annotations. Our database schema allows for very efficient storage of such sparse sets.

Also, the in- and out- points in the result set of a particular query is a function of the query itself. Full flexibility is available for applications to apply whatever granularity of segmentation is appropriate for user queries or for the content indexing needed. For instance, some commercially available systems use shot transitions as their segmentation of video and results are returned at the granularity of a scene. Our system automatically indexes scene transitions, and an application can easily map the results into "scenes" if so desired. This feature, called *rounding up* is discussed in more detail in the next section.

**Additional Operations on Time-Based Segment Specifications**

Our system contains special features which are useful in manipulating search results for purposes of helping users find relevant results, or relating video segments to a larger video context from which the results are derived.

**Combining Multiple Rankings**. Searching video segments by text associated with the video (e.g., caption text) returns results when a full text index is searched. Multiple search engines may return additional rank ordered result sets, e.g., search on recognized speech text (which can be indexed separately from caption text). In these cases, the system needs to compute a combined rank for a combined result set. Different considerations have to be taken into account in this combination process, such as the different types of content, user preferences or the reliability of the search. For example, the results of text search against the written transcript are 100% accurate whereas a search against recognized speech transcript does not in general achieve the same accuracy. Hence when combining these two ranks, more weight should be given to the rank returned from the search against the written transcript.

**Segment Padding.** In some cases, the constraint-satisfying interval in a search result is very short, e.g., corresponding to utterances of individual words. We provide a user-settable "padding factor" time constant, which is used to extend both the start and the end of each final answer set interval. Padding can be done at different times: one can pad the result sets of each sub query (e.g., the results from a text search, the result from a parametric search); or one can pad the final result set after combining the result sets of the sub queries. Our system allows for all these methods of padding. Padding intervals may be application specific, and values may be defined by default settings or exposed to end users who choose values based on their search requirements.

**Rounding Up.** A related feature is *rounding up*. In some cases, such as news stories, users may wish to resort to predefined semantic partitioning of the videos which would be captured as an annotation feature. They might want, for example, to view a complete story containing a result segment. Our scheme allows for overriding result

segment boundaries computed for Boolean queries along the following lines: Take each time interval T in the result set and determine the smallest predefined time interval T1 which contains T in its entirety. (Note that T1 could be a combination of several adjoining predefined time intervals.) This feature exposes the strength of our flexible segmentation. Any fixed segmentation (like the one defined by higher level segment criteria such as new story boundaries) can be imposed after the result set for a query has been defined, and hence the user is not restricted to seeing the results in terms of the segmentations produced by the original indexing methods.

## Conclusions

In this paper we presented a video content management system which is designed for storing and retrieving digital videos and metadata associated with them. Our system focuses on representing and storing the temporal duration of the metadata, using methods that result in efficient storage, efficient retrieval, and flexible searching and browsing. Each metadata annotation determines the segment granularity at which it is to be represented in the database index. Additional time-based operations allow an application (user) to determine the granularity at which the video data is presented as a search result and when result sets are browsed and played back.

## Acknowledgments

We would particularly like to acknowledge Michael Concannon's programming contributions to making the end user interface prototype a reality. We also thank the other members of the project team who developed the annotation methods referred to in this paper: Chitra Dorai and Jonathan Connell (and their IBM Research manager Ruud Bolle).

Our system is being developed as part of a cooperative, multicompany effort to design and implement an architecture for a complete studio for producing and broadcasting high-definition television (HDTV) using MPEG-2 compressed bitstreams. The specification, known as "Digital Studio Command and Control" (DS-CC) is a distributed objects model, implemented in CORBA. It has been submitted to the Society of Motion Picture and Television Engineers (SMPTE) as a recommendation as a broadcasting industry standard. The prototype system described here and the underlying server support are implemented in Java as DS-CC objects.

The work was funded in part by the National Institute of Standards and Technology, The Advanced Technology Program (NIST/ATP) under contract No. 70NANB5H1174. We are also grateful for the support of the NIST program, in particular the NIST Technical Program Manager, David Hermreck, and for the IBM Research Principal Investigator, James Janniello, who leads the studio infrastructure aspects of the NIST ATP project.

## References

[1] Aguierre Smith, T.G. and Davenport, G., "The Stratification System, A Design Environment for Random Access Video", Network and Operating Systems Support for Digital Audio and Video. Third International Workshop Proceedings 1993, p. 250-61, La Jolla, CA, USA 12-13 Nov. 1992. Springer Verlag

[2] ISLIP Media, Inc. "www.islip.com", Pittsburgh, PA 15312, USA.

[3] Christel, M., Stevens, S., Wactlar, H., "Informedia Digital Video Library," In Proceedings of the ACM Multimedia '94 Conference. New York, ACM. pp. 480-481, October, 1994.

[4]"Avid's Open Media Management Initiative," Content Watch Vol 1, No 5. 1998.

[5] Virage Inc., "www.virage.com", San Mateo, CA, USA.

[6] Excalibur Technologies Corporation. "www.excalibur.com", Vienna, VA 22182, USA.

[7] R. M. Bolle, B.-L. Yeo, M. M. Yeung, "Videoquery: Research Directions", IBM Journal of Research and Development, Volume 42, Number 2, March 1998

[8] M. Davis, " Media Streams: An Iconic Viusal Language for Video Representation", Proceedings of the 1993 Symposium on Visual Languages, pp. 196-202.

[9] S. Dharanipragada, and S. Roukos, ``Experimental results in Audio-Indexing", Proceedings of 1997 Speech Recognition Workshop,Chantilly,Virginia, February 1997.

[10] James F. Allen "Maintaining Knowledge about Temporal Intervals," Communications of the ACM, Vol 26, No 11, 1983

[11] Juliano, M., and Pendyala, K. Use of Media asset management systems in broadcast news environments. Whitepaper in: NAB Multimedia World Journal, 1997.